International Review of Research in Emerging Markets and the Global Economy (IRREM)
***An Online International Research Journal* (ISSN: 2311-3200)**
2016 Vol: 2 Issue: 1

Software Project Management Challenges

Pradip Peter Dey,
National University,
School of Engineering and Computing, USA.
E-mail: pdey@nu.edu

Muzibul Khan,
Kyocera Communications, Inc,
9520 Towne Center Drive, San Diego, CA 92121.

Mohammad Amin,
Bhaskar Raj Sinha,
Hassan Badkoobehi,
National University,
School of Engineering and Computing, USA.

Abstract

Managing complex software projects is enormously challenging; however, the challenges can be overcome by integrating appropriate management strategies with a strong team of professionals in an iterative software development process that promotes robust communication. Some iterative models of software development start with risk analysis in each iteration. This paper addresses software project management challenges by integrating software aspects with an iterative model of development using risk analysis and effective management strategies. There are many risk categories including communication risks, project planning risks, technical risks, budget risks, scheduling risks, legal risks, ethical risks, operational risks, security risks, and personnel risks that require timely attention. Potential risks should be identified, analyzed, evaluated, and appropriate strategies should be developed for managing imminent risks in a timely manner. This paper advocates a strategy that suggests a special role to communication risk because it interacts with other risks in a way that may allow its coupling with most other risks. In our framework, interacting risks are monitored and controlled with special attention to communication risk because it is considered a super-risk. The current state of knowledge indicates that a reasonable solution is to obtain early warnings from a team of risk analysts about potential risks, which are then further studied in interaction contexts in order to develop plans for risk mitigation strategies with software process phases.

Key Words: *Communication, iterative models, risk analysis, software development, volatility.*
JEL Classification : *C 19, G13, G 14*

1. Introduction

Alternative approaches to management developed over the years have been extremely important to modern software project management. However, new approaches are necessary to deal with some of the toughest challenges. According to experts, "the popular interest in management as a discipline and a field of study is fairly recent. But management, both as a practice and as a field of study, has a respectable history, in many different countries, going back almost two centuries" (Drucker 2008, p.12). Modern approaches to management developed in the U.S.A. after World War II are currently being taught at colleges and universities. The four classic management functions one may learn in school include: "planning, organizing, leading, and controlling" (Nelson & Economy 2005, p. 5). More functions need to be added to the list of skills for acquiring in the process of learning about project management aspects. Experts such as McFarland (1970) emphasized the role of managers in the project process. Management is the "process by which managers create, direct, maintain and operate purposive, organizations though systematic, coordinated, cooperative human effort" (McFarland 1970, p. 5). "Because projects are transient, their delivery follows a development process, from germination of the idea, through initiation, design and delivery, to commissioning, handover to the client and closeout of the work." (Turner 2014, p. 5). There is a great deal of interest in alternative processes in software development projects and the ideas of traditional process control are typically executed in software projects. Henry Gantt, who was a pioneer in planning and control techniques, proposed a chart, popularly known as the Gantt chart, as a project management tool (Gantt 1919). Software projects have often failed even under standard supervision of process elements and with standard management techniques. That is, in addition to traditional project management functions, there are additional challenges for software project management, which play significant roles in modern software industry.

Large software projects are difficult to measure and challenging to understand and manage. "The Standish Group research shows a staggering 31% of projects will be cancelled before they ever get completed. Further results indicate 52.7% of projects will cost 189% of their original estimates." (Leffingwell & Widrig 2000, p.6). Many software projects start in a hurry with a partial understanding of the problem to be solved with an assumption that the problem can be solved when a better understanding is achieved after further analysis of the problem with appropriate resources. However, volatility of the project elements starts appearing with considerable ambiguity with further analysis. Elements of a complex software system interact in a way that is both difficult to understand and to manage. In addition, various aspects of the project also interact with everything else. Various risk factors may become clear to some team members or specialists in the process of detailed analysis.

However, communication of such risk analysis may pose another level of risk. This risk is known as a communication risk, which is a super-risk that may negatively affect other risks both due to the difficulties involved in the process of communication and more importantly, because it permeates all aspects of software development. For example, if the problem has un-decidable elements that require technical communication at mathematical levels then the communication difficulties may prevent a general understanding of the problem among the team members and stakeholders. Other interactions among risks need to be studied in a new innovative method in order to make improvements in examining consequences of the risks and in developing strategies for their management either by proactive or reactive steps. Interacting risks should be studied using a comprehensive model of interactions among risks, process phases, software components etc., and properly integrated into the project management framework. The goal of this paper is to review the current practices of software project management and address software project management challenges by integrating software aspects with agile and iterative models of development using risk analysis and management strategies. There are many risk categories including communication risks, technical risks, budget risks, scheduling risks, legal risks, ethical risks, operational risks, security risks, and personnel risks that require timely attention often from specialists. Potential risks should be identified, modeled, and evaluated and then appropriate strategies should be developed for managing the imminent risks in a timely manner. This paper advocates a better-shared understanding of software complexity through studies of interacting risks together with software component interactions and interactions among software process phases. A team of risk analysts may provide early warnings about potential risks, which are then further studied, for possible interactions in order to develop appropriate management strategies. By integrating risk management into a software development process, it coordinates all activities in a comprehensive manner. It places special emphasis on interacting risks, which are becoming prominent in commercial and government systems (Bannerman 2008).

2. Literature Review

Traditional management strategies have not worked well for software projects. Software development projects are prone to special challenges. Several researchers have paid attention to these challenges and made progress towards a better understanding of them (Barros, Werner & Travassos 2004; Boehm 1991; Bannerman 2008; Charette 2005; Chemuturi & Cagley Jr. 2010; Jones 1998; Gulla 2012; Keil, Cule, Lyytinen & Schmidt 1998; Taylor 2006; Xia & Lee 2004). Some of the studies assessed the problem as follows ". . . software has long been one of the most troublesome technologies of the 20th century, and one that has long been resistant to executive control. One of the main reasons that software is difficult to

control is because it has been difficult to estimate software projects and to measure software quality and productivity" in a scientific way (Jones 1998). Risk management practices often differ from what have been suggested by experts (Ropponen & Lyytinen 1997; Taylor 2006). In September 2015, the U.S. Environmental Protection Agency (EPA) disclosed that Volkswagen could face fines of as much as $18 billion for allegedly manipulating exhaust-emissions tests. The EPA has initiated a criminal investigation against the company. "Volkswagen acknowledged that it installed software in some diesel-powered cars to make it appear that the cars met tough U.S. anti-smog rules, the EPA said." (The Wall Street Journal, Sept. 22, 2015, page B1). Michael Horn, Volkswagen's U.S. chief executive told a House subcommittee hearing on October 8, 2015 that a deceptive piece of software was put into as many as 11 million Volkswagen vehicles worldwide in order to fool emission-testing equipment (Los Angeles Times, Oct 9, 2015). The Volkswagen software was designed to pollute the air in the normal operating conditions of a car and reduce emission in the test conditions. This complex case is yet to be resolved; but it has implications for involved software engineers and project managers.

Promoters of software project risk management suggest that by taking adequate steps against risks, the incidence of problems in software could be reduced (Boehm 1991; Charette 1989; Jones 1994, Charette 2005). An apparently good approach is to identify the risks before they become major threats, and suggest appropriate remedies through planning. According to a study by Keil, Cule, Lyytinen and Schmidt (1998), three independent panels of experienced software project managers, "selected a common set of 11 risk factors as being among the more important items" (Keil, Cule, Lyytinen & Schmidt 1998: p. 78) for risk based software project management. The study emphasized identification of risks and their relative importance as perceived by managers. This is necessary but not sufficient for averting project failures. What is required is a set of effective steps against the risks so that their cumulative effect can be stopped in a timely manner to save a vulnerable software project. Lack of top management commitment to the project is often cited as an important risk factor, although it was not clear as to whether robust communication channels among the stakeholders were present in the environment. Risks may proliferate rapidly in an interactive spiraling manner with harmful consequences when communications among stakeholders degrade.

In a review, Gulla (2012) observes that by properly classifying documented causes of IT project failure, 54 percent are attributed to poor project management - the most cited factor, whereas only 3 percent are attributed to technical challenges. Gulla (2012) identifies seven key factors that are primarily responsible for project failures: (1) poor project planning and direction, (2) insufficient communication, (3) ineffective management, (4) failure to align with constituents and stakeholders, (5) ineffective involvement of executive management, (6)

lack of soft skills or the ability to adapt, (7) poor or missing methodology and tools. It is to be noted that poor project planning and other risks may result from insufficient communication.

Complex software projects are often studied to identify the key factors so that remedial steps can be suggested. Often subjective questions are asked to project managers or interested stakeholders and responses are statistically analyzed in order to come up with a perceived account of the problem. Xia, and Lee (2004) tried to relate their account of information system development project complexity with project performance based on responses from subjective questions. There are other studies (such as Keil, Cule, Lyytinen, & Schmidt 1998) that attempt to account for risk analysis based on surveys with subjective questions. Considerable progress has been made with these studies; however, a better understanding of software development projects is needed in order to make additional improvements. Software development is different from other traditional product development. "Rather than a product in the traditional sense, software is better viewed as a container for the real product. What the customer buys and the user employs is the executable knowledge contained in the software" (Armour 2015, p.32). The true nature of software imposes an overarching requirement on the study of the interactions of risk factors with the preeminent role of communication risk. Software development has often been considered one of the most challenging processes of modern technology. A complex software system with highly interactive elements is difficult for dynamic modeling and allows multiple interpretations and development perspectives (Pressman & Maxim 2014; Sommerville 2015). Some approach it from a scientific perspective while others treat it in an artistically creative manner. Over the decades, a multitude of approaches to software development have been proposed. Donald Knuth (1969) initially indicated that software writing is an art (Knuth 1969). David Gries argued it to be a scientific endeavor (Gries 1981). Watts Humphrey invested a considerable amount of time and effort in the study of software development problems and proposed software development primarily as a process (Humphrey 1989). In recent years, many practitioners have studied the software development from a different point of view and have come to realize that software is engineered (Pressman & Maxim 2014; Sommerville 2015; Braude & Bernstein 2011). Because of the adoption of engineering methods along with agile or iterative processes, there is an opportunity to address software project management challenges in a new way in the engineering process. In each iteration, risk factors can be reviewed and evaluated for proper management. In addition, managers can be proactive in developing quick response capabilities for handling realized threats (Bannerman 2008). An analytical account of communication risk and its interactions with other risk factors along with interactions among software components and process phases may reveal certain aspects of software projects and their contributions to the project management. That is, an analytical account will set the

stage for further experimental studies needed in order to make progress in software project management.

3. Risk Analysis

Based on the review of past studies presented in the preceding section, an analytical account of software project risk factors and their interactions is needed in order to make further progress. The communication risk and its interactions with other risk factors will be specially emphasized because of the complexity created by these interactions. Armour's view of software (Armour 2015) reveals some important aspects of software that pose a serious challenge to the traditional concepts held by most software project managers. Software project managers should be prepared to make appropriate conditions for collaborative knowledge development where structured and unstructured communication among the developers play the most important role. Teams of software professionals would be motivated to develop a shared understanding of the problem aspects through communication and consultations (Li, Ko, & Zhu, 2015). In order to achieve the shared understanding, professional software developers often depend on intuitive extensions of the meanings of words, phrases, terms, gestures, diagrams, artifacts etc. in semi-structured or unstructured communication environments. Limited terminologies, diagrams, languages and artifacts that are available today are found in popular software engineering textbooks (Pressman & Maxim 2014; Sommerville 2015; Braude & Bernstein 2011). Additional artifacts are found in IEEE and ISO standards, journals, monographs and reference manuals such as Rumbaugh, Jacobson and Booch (2005). However, despite all these, software engineers struggle in finding contextually meaningful terms, phrases, languages, tools and artifacts in order to succeed in communication. Recent studies have often identified inadequate or insufficient communication as one of the most important factors for project failures or setbacks (Gulla 2012). Poor project planning and direction is at the top of the list of factors and an important part of planning is to assign the right people to the right task and make clear assignments to team members, with defined goals and responsibilities (Gulla 2012). It is reasonable to argue that communication factors interact with project planning and resulting dynamics evade analysis if their combined effects are not adequately studied. Suppose the project under consideration has considerable security risk and two team members need to be assigned security related responsibilities with resulting dynamics of three way interactions; the lack of adequate terms and artifacts makes it difficult to communicate many aspects of security while the general problem of security is proven to be un-decidable (Harrison, Ruzzo & Ullman 1976). The main argument is that the communication problem in software projects interacts with other problems in such a way that many software projects suffer substantial loss resulting from the interacting risks. That is, communication risk is a super-risk that may

negatively impact other risks for a considerable period of time because of the difficulties involved in the process of communication. In the case of Volkswagen's recent problem mentioned above, many suspected risk factors including technical risks, legal risks, ethical risks, operational risks, and personnel risks etc. may have been drastically impacted by the communication risk leading to an unprecedented debacle. Although the Volkswagen emissions cheating scandal was an ethical failure, it could only happen in an environment of inadequate corporate governance, bad corporate management, and bad software project management. Failure in corporate oversight by definition generally boils down to a failure in communication, as does failure in management. Proper communication promotes a shared understanding of risk factors among managers and other stakeholders, and a shared sense of purpose in problem solving.

Certain agile and iterative software development models have incorporated frequent structured and unstructured communication events that are often effective in the face of the challenges resulting from the interacting risks. In these development models, the lack of proper terms, artifacts etc. are often overcome eventually by means of informal intuitive and extended communication among team members who work closely with each other in a process model that frequently put them on communication in a collaborative environment. If a software project manager is an active participant in such a team, then communication risk and other interacting risks may be mitigated effectively in a timely manner. Since software projects deal with creation of executable knowledge (Armour 2015), there is an extra demand on most of the active participants to rise up to a level of intellectual maturity where communication can take many forms using a wide range of formal and informal artifacts, tools, terms, languages etc. If the teams are separated geographically and culturally, the communication challenges may get harder to manage, and establishing a feeling of trust and belonging is also difficult in distributed teams (Carmel & Agarwal 2001; Holmström, Conchúir, Ågerfalk, & Fitzgerald 2006; Espinosa, Slaughter, Herbsleb & Kraut 2007). Careful monitoring of risks and periodic review of the impacts of interacting risks is essential. From the preceding logical analysis, it is clear that (1) interactions among risk factors, (2) interactions among distributed team members, (3) lack of adequate tools, languages and artifacts of communication, (4) interactions among software components, and (5) interactions among software development phases all together make the issues more complicated where success depends heavily on a robust system of communication that should be fully developed in order to deal with the future software systems. Healthy communications may lead to innovative solutions by involvement of many participants with multiple perspectives; inadequate communication, on the other hand, often leads to project failure, deception, and long-term harm to individuals, groups or communities. What starts with inadequate and deceptive communication may go further beyond a project failure as in many cases mentioned

in the Forbes magazine: "Satyam Systems, a global IT company based in India, has just been added to a notorious list of companies involved in fraudulent financial activities, one that includes such names as Enron, WorldCom, Societe General, Parmalat, Ahold, Allied Irish, Bearings and Kidder Peabody" (Balachandran 2009).

4. Conclusions and Recommendations

It is evident from the published studies that software project management challenges are different from those of traditional project management because of the nature of software, which is very different from traditional products. Software development is characterized by highly creative activities marked by a wide range of complex communication patterns among software engineers in order to make joint decisions, often at an intellectual level that is not well understood. In this context, the logical importance of communication risk and its relationship with other risk factors as it permeates all facets of software development lead us to consider it as a super-risk, a special category for its volatile interactions with other risk factors and software elements in patterns that are often considered to be "beyond the reach" of ordinary humans. Well-coordinated team efforts by well-formed teams are needed in order to deal with interacting risks in complex software projects. That is, while individuals may find the challenges of managing complex software projects overwhelming, a strong team of professionals may overcome the challenges by integrating appropriate management strategies into an iterative software development process in a collaborative manner. Some iterative development models are flexible enough to accommodate the special role of communication risk within their processes allowing a broad range of communication forms including semi-structured, formal, informal, intuitive, structured and unstructured. Future studies may be suggested with the goals of developing a model of interactions appropriate for all types of risks for complex software projects and promoting a better-shared understanding of the range of complexities encountered in software systems.

Acknowledgement: Authors are grateful to Laith Al Any, Shatha Jawad, Debra Bowen, Khadija Amin, Arun Datta and many others for their comments and suggestions on earlier versions of this paper.

References

Armour, P. G., 2015, The business of software: Thinking thoughts, *Communications of the ACM*, Vol.58, No. 10, pages 32-34.

Balachandran, S. V., 2009, The Satyam scandal, *Forbes*, January 07, 2009. Retrieved December 14, 2015, from:
http://www.forbes.com/2009/01/07/satyam-raju-governance-oped-cx_sb_0107balachandran.html

Bannerman, P., 2008, Risk and risk management, in software projects: A reassessment, *Journal of Systems and Software*, Volume 81, Pages 2118-2133.

Barros, M., Werner, C. & Travassos, G., 2004, Supporting risks in software project management, *Journal of Systems and Software*, Volume 70, Issues 1–2, pages 21–35.

Boehm, B., 1991, Software risk management: principles and practices, IEEE Software 8,1, pages 32-41.

Braude, E. & Bernstein, M., 2011, Software engineering: Modern approaches, (2nd Edition), John Wiley & Sons.

Carmel, E. & Agarwal, R., 2001, Tactical approaches for alleviating distance in global software development, IEEE Software, Vol. 18, No. 2, pp. 22-29.

Charette, R.N., 1989, *Software engineering risk analysis and management*, McGraw-Hill, New York.

Charette, R.N., 2005, Why software fails? *IEEE Spectrum* 42 (9), pages 42–49.

Chemuturi, M. & Cagley Jr., T., 2010, Software *Project Management: Best Practices, Tools and Techniques*. J. Ross Publishing, Plantation, Florida.

Drucker, P., 2008, Management (Revised Edition), Harper Collins, New York.

Espinosa, J. A, Slaughter, S. A., Herbsleb, J. D., Kraut, R. E., 2007, Team Knowledge and Coordination in Geographically Distributed Software Development, *Journal of Management Information Systems*, Vol. 24, Issue 1, 2007.

Gantt, H., 1919, *organizing for work*, Harcourt, Brace, and Howe, New York.

Gries, D., 1981, *The Science of Programming*. Springer, 1981.

Gulla, J., 2012, Seven Reasons IT Projects Fail, *IBM Systems Magazine*, Retrieved August 8, 2015 from:
http://www.ibmsystemsmag.com/power/Systems-Management/Workload-anagement/project_pitfalls/

Harrison, M. A., Ruzzo, W. L. & Ullman, J. D., 1976, Protection in Operating Systems, *Communications of the ACM* Vol. 19, No.8, Pages 461–471.

Holmström, H., Ó Conchúir, E., Ågerfalk, P.J. & Fitzgerald, B., 2006, Global Software Development Challenges: A case Study on Temporal, Geographical and Socio-Cultural Distance, ICGSE2006, Costao do Santinho, Florianopolis, Brazil, Oct 16-19, 2006.

Humphrey, W., 1989, Managing the Software Process, Addison-Wesley, Reading, Mass.

Jones, C., 1994, *Assessment and control of software risks*, Prentice-Hall, Englewood Cliffs, N.J.

Jones, C., 1998, What it means to be 'Best in Class' for Software, Capers Jones, Software Productivity Research, Inc. www.spr.com.

Keil, M., Cule, P. E., Lyytinen, K., & Schmidt, R.C., 1998, A framework for identifying software project risks, Communications *of the ACM*, Vol.41, No. 11. (Pages 76-83).

Knuth, D.E., 1969, Seminumerical Algorithms: The Art of Computer Programming 2. Addison-Wesley, Reading, Mass.

Leffingwell, D. & Widrig, D., 2000, *Managing software requirements: a unified approach*, Addison-Wesley, New York.

Li, P. L., Ko, A. J. & Zhu, J., 2015, What Makes A Great Software Engineer?, The 37th International Conference on Software Engineering, May 20-22, 2015, Florence, Italy.

Los Angeles Times, 2015, VW bosses say they didn't know, Los Angeles Times, October 9, 2015, page C1.

McFarland, D., 1970, Management: Principles and Practices (3rd edition), The McMillan Company, London.

Nelson, B. & Economy, P., 2005, *The management bible*, John Wiley & Sons, Hoboken.

Pressman, R. S. & Maxim, B., 2014, Software Engineering: A Practitioner's Approach. (8th edition), McGraw-Hill.

Ropponen, J. & Lyytinen, K., 1997. Can software risk management improve system development: an exploratory study? European Journal of Information Systems 6, (1), pages 41–50.

Rumbaugh, R., Jacobson & Booch, G., 2005, I the Unified Modeling Language Reference Manual. (2nd Edition), Addison Wesley.

Sommerville,I., 2015, Software Engineering, (10th edition), Addison Wesley.

The Wall Street Journal, 2015, U.S. Begins Criminal Probe of VW, Reported by William Boston, in The Wall Street Journal, Sept. 22, 2015, page B1.

Taylor, H., 2006. Risk management and problem resolution strategies for IT projects: prescription and practice. *Project Management Journal*, 37 (5), pages 49–63.

Turner, R., 2014, A handbook for project management practitioners, in Turner, R. (ed.), 2014, *Gower Handbook of Project Management*, Gower Publishing, Burlington.

Xia, W., & Lee, G., 2004, Grasping the complexity of IS development projects, *Communications of the ACM*, Vol.47, No. 5 (pages 68-74).