Current Trends in Software Project Management

Pradip Peter Dey[1, a], Mohammad Amin[2, b], Bhaskar Raj Sinha[3, c], Shatha Jawad[4, d], Laith Al Any[5, e] and Hassan Badkoobehi[6, f]

[1,2,3,4,5,6]School of Engineering and Computing, National University, San Diego, California, USA

[a]pdey@nu.edu, [b]mamin@nu.edu, [c]bsinha@nu.edu, [d]sjawad@nu.edu, [e]Lalany@nu.edu, [f]hbadkoob@nu.edu

Keywords: Business, communication, risk analysis, software development.

Abstract. Complexity of software project management has been increasing rapidly in recent years. Project management strategies are evolving in order to address the complexity problems. A persistent struggle is to find an adequate balance between conflicting factors such as providing freedom and flexibility to software engineers in order to solve complex problems in a creative way while managing risks associated with freedom and flexibility. Risk based strategies for software project management appear to be promising. Potential risks are identified, analyzed, assessed and monitored. Appropriate steps are taken for managing imminent risks in a timely manner. This study suggests that communication risk plays a special role in software projects, because it interacts with other risks in startling ways. Interacting risks need to be studied at several levels with special attention to communication risk, because it could be considered as a super-risk. Current trends in software project management include analysis of all categories of risks such as technical risks, budget risks, scheduling risks, legal risks, ethical risks, operational risks, security risks, communication risks, project planning risks, and personnel risks that require timely intervention without restricting the creative power of executable software knowledge development.

1 Introduction

Practitioners in technology are eager to “change the world” by getting rid of the past and establishing something new that works for the future (See [1] for Steve Job’s expressions). The intended message behind “change the world” has been genuine, attractive, clear, and alluring. However, in the context of the emerging complexity of the world, it is not achievable in a straight forward way. Complexity of the technological project management has increased thousands of times in the last few years due to cybersecurity and related problems. From the publication of HRU Model, it has become clear to most practitioners that there is no algorithmic solution to the computer system security problems [2]. In addition to security risk, there are many other risk factors including communication risks, personnel risks, ethical risks, project planning risks, technical risks, budget risks, scheduling risks, legal risks and operational risks that require timely attention for managing software projects. All potential risks should be identified, analyzed, assessed and monitored for their imminent dangers. Appropriate strategies should be developed for managing impending risks in a timely manner. This paper reviews the new trends in software project management and assesses their potentials. In order to promote creative solutions to software problems, software engineers are usually given freedom and flexibility at the workplace. However, such flexibility comes with many risks which have been mentioned above. Risk based project management dominates current trends, because the intellectual nature of software is different from traditional products [3]. In most software products, hundreds of elements of executable knowledge interact in complex ways that are difficult to comprehend by outsiders [1][3]. It is believed by many that Steve Jobs hired an outsider in 1983 who then ousted Jobs from Apple, and then after a few years, Jobs had to come back to Apple in order to save it [1].

One size fits all approach does not work with software development projects. Multiple perspectives are often proposed for consideration for the study of the nature of a software problem.

Donald Knuth (1969) initially indicated that software writing is an art [4] while David Gries argued it to be a scientific endeavor [5]. Design and implementation of graphical user interfaces (GUIs) with amazing user experience usually requires an iterative approach whereas one pass approach was advocated in traditional structured programming and structured development [6-8]. Current trends in software project management have been characterized by several new factors such as globalization, geographically dispersed workforce, advances in technologies, security threats and risk based project management.

2 Literature Review

Henry Gantt was known for his pioneering work in planning and control techniques who proposed a chart, popularly known as the Gantt Chart, as a project management tool [9]. Gantt's insights and innovations helped project management in a significant way. Management, both as a field of study and as a practice has a long history, going back about 200 years [10-12]. However, modern approaches to management were developed in the U.S.A. after World War II and various versions of these are currently taught in educational institutions throughout the world. The four classic management functions one may learn in school are "planning, organizing, leading, and controlling" [13, p.5]. In addition to these four functions, experts may add several other functions and skills such as communicating, decision making, coaching, meeting ethical standards, analytical thinking, listening, negotiating, and visioning. Even after proper education, project managers face many challenges with processes, products, technologies etc. "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" [14, p.5]. Many software process concepts, life cycle models, and agile methods are now part of most software engineering textbooks [7-8].

In the past, many software projects have failed due to challenging management issues. Several researchers have studied these challenges and made progress towards a better understanding of them [15-28]. One of the research investigations has 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 an accurate way" [18]. In September 2015, the U.S. Environmental Protection Agency (EPA) disclosed that Volkswagen could face billions of dollars of fines 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.", according to the Wall Street Journal, Sept. 22, 2015, page B1 [29]. It was evident that Volkswagen's deceptive software was designed to sense when the car was being tested and then activated equipment that reduced emissions in order to pass the test. It is clear to practitioners that some conditional logic statements such as "if test-mode is on, then do { }" must have been used by the members of the Volkswagen software team. By implications, software engineers must have violated ethical and professional standards. The management of ethical risk factors failed in this case and unethical conducts did not surface in time due to lack of proper communication. Structured and unstructured communication among software developers, managers, analysts, and other stakeholders usually plays an important role in identifying risk factors in a timely manner. Risk assessment and mitigation should be part of every software project planning [30], because the creative development of software as executable knowledge involves risks [3]. The creative aspects of software development cannot be compromised for any reason because the expectation is that the best user experience must be delivered with most software applications in order to achieve competitive edge.

3 Trends and Challenges

Software systems are among the most complex artifacts humans ever built. The current trend is to get a deeper understanding of software as a special kind of knowledge product as explained by Armour [3]. The project management strategies should be based on conditions for creating executable knowledge in a cost effective manner. Armour's view of software [3] reveals some important aspects of software development methods that pose serious challenges to the traditional concept used by many project managers [1][3][8]. Project managers should be receptive to collaborative knowledge development in an innovative way where structured and unstructured communication among the developers plays one of the most important roles. A professional team of software engineers should be motivated to develop a shared understanding of the problem to be solved [31] and, therefore, communication by technical and non-technical means become important. In order to achieve the shared understanding, practitioners often depend on intuitive extensions of meanings of words, phrases, terms, gestures, diagrams etc. in semi-structured or unstructured communication environments, because formal languages (such as UML [32]) proposed so far are inadequate for the purpose [33-34]. Limited terminologies, diagrams, languages and artifacts that are available today can be found in popular software engineering textbooks [7][8] and additional tools can be found in IEEE and ISO standards, journals, monographs and reference manuals [32]; however, these are inadequate for designing complex systems. Great design is challenging [34]. A recent study by Gulla has identified inadequate or insufficient communication as one of the most important factors for project failures or setbacks [19]. Poor project planning and direction is at the top of Gulla's list of factors and an important part of planning is to assign the right people to the right task and make clear assignments to each team member, with defined goals and responsibilities [19] where communication plays an important role. Suppose the project under consideration has considerable security risk and three team members need to be assigned security related responsibilities with resulting interaction dynamics. It would be extremely difficult to communicate many aspects of security due to inadequate representation of security in design models while the general problem of operating system security is proven to be un-decidable [2]. The lack of adequate terms, diagrams, and artifacts makes it difficult to communicate many crucial aspects of software. A substantive argument is that the communication problem in software projects interacts with other problems in such a way that many software projects suffer extensive loss resulting from the interacting risk factors. Risks should be analyzed at least at two levels: in the first level, risk factors should be analyzed individually emphasizing their unique features; in the second level, interactions among risk factors should be studied for their connections and ramifications. In addition, communication risk should be treated as a special super-risk that may negatively affect other risks for a considerable time period because of the challenges involved in the process of communication as software development involves knowledge creation by collaborative means [3].

Risk based project management with adequate attention to communication is a part of modern trends. Redesigning and reworking parts of a project leading to missed schedules or budget overrun is considered unacceptable by stake-holders. This drives the re-evaluation of project management techniques in order to find new approaches to minimize risks and avoid schedule over-runs. The current considerations include involving the customers early in the development phase, prototyping at an early stage to determine requirements, staged delivery to meet the market 'window of opportunity', frequent planned upgrades to meet customer expectations and structured and un-structured communication among stakeholders.

4 Conclusion

It is evident from the practices and published studies that software project management principles are different from those of traditional project management, because of the complex nature of software, which is very different from traditional other products. The current trends in software project management take into account the intellectual nature of software creation as distinct from

traditional product development. The creative process of executable knowledge development is relatively riskier than various other product development. However, it is clear that under proper risk management, innovative software can be developed in a cost effective way. One of the most important aspects of risk based software project management is to understand the importance of communication risk as a super-risk and its relationship with other risk factors as it permeates all facets of software development. Agile team efforts by well-motivated teams are needed for dealing with interacting risks in complex software projects. Team efforts for software development needs to be properly integrated with team efforts for managing complex software projects. Future studies may be suggested with the goals of quantitative analyses of emerging trends by collecting experimental data for measuring various aspects of emerging trends in software project management. The trends need to be continually watched and studied because they are likely to change during the emerging process.

Acknowledgement

Authors are grateful to John Cicero, Abdul R. Amin and many others for their comments and suggestions on earlier versions of this paper.

References

[1] Walter Isaacson, Steve Jobs, Simon & Schuster, 2011.

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

[3] P. G. Armour, The Business of Software: Thinking Thoughts, Communications of the ACM [M], Vol.58, No.10, 2015, Pages 32-34.

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

[5] D. Gries, The Science of Programming. Springer, 1981.

[6] Edsger W. Dijkstra, "On the role of scientific thought". Selected writings on Computing: A Personal Perspective. New York, NY, USA: Springer-Verlag, 1982, Pages 60–66, ISBN 0-387-90652-5.

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

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

[9] H. Gantt, Organizing for Work, Harcourt, Brace, and Howe, New York, 1919.

[10] Peter F. Drucker, Management (Revised Edition), Harper Collins, New York, 2008.

[11] Morgan Witzel, A History of Management Thought, 5th Edition, Wiley, 2004.

[12] Bernard W. Taylor III, Introduction to Management Science, (12th Edition), 2015.

[13] B. Nelson & P. Economy, The Management Bible, John Wiley & Sons, Hoboken, 2005.

[14] R. Turner, A Handbook for Project Management Practitioners, in Turner, R. (ed.), 2014, Gower Handbook of Project Management, Gower Publishing, Burlington, 2014.

[15] B. Boehm, Software Risk Management: Principles and Practices, IEEE Software 8 [J], 1, 1991, Pages 32-41.

[16] M. Barros, C. Werner, & G. Travassos, Supporting Risks in Software Project Management, Journal of Systems and Software [J], Volume 70, Issues 1-2, 2014, Pages 21-35.

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

[18] C. Jones, What it means to be 'Best in Class' for Software, Capers Jones, Software Productivity Research, Inc. 1998.

[19] J. Gulla, Seven Reasons IT Projects Fail, IBM Systems Magazine [M], Retrieved August 8, 2015 from: http://ibmsystemsmag.com/power/systems-management/workload-management/project_pitfalls/

[20] M. Keil, P. E. Cule, K. Lyytinen, & R. C. Schmidt, A Framework for Identifying Software Project Risks, Communications of the ACM [M], Vol.41, No. 11. 1998, Pages 76-83.

[21] W. Xia, W. & G. Lee, Grasping the complexity of IS development projects, Communications of the ACM [M], Vol.47, No. 5, 2004, Pages 68-74.

[22] E. Carmel & R. Agarwal, Tactical Approaches For Alleviating Distance In Global Software Development, IEEE Software [J], Vol. 18, No. 2, 2001, Pages 22-29.

[23] P. P. Dey, M. Khan, M. Amin, B. Sinha, H. Badkoobehi, S. Jawad, L. Al Any, Managing interacting software project risks [C], Proceedings of the Global Business Research Congress (GBRC), 2016.

[24] R. N. Charette, Software Engineering Risk Analysis and Management, Intertext, New York, 1989.

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

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

[27] C. Jones, Assessment and Control of Software Risks, Prentice-Hall, Englewood Cliffs, N.J., 1994.

[28] U.S. Air Force Systems Command, "Software Risk Abatement," AFSCIAFLC Pamphle 800-45, 1988.

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

[30] P. Jalote, Software Project Management in Practice. Boston: Addison Wesley, 2002.

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

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

[33] P. P. Dey, B. R. Sinha, G. Romney, M. Amin, H. Badkoobehi, Innovative User Interface Engineering. Proceedings of the International Conference on Innovative Engineering Technologies (ICIET 2014) Bangkok [C], December 28-29, 2014. Pages 10-15.

[34] J. Hong, "Why is Great Design so Hard?", Part 1-2, Communications of the ACM [M], July-August, 2010.