4th IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2010)
© 2010 IEEE.

A state of the art review on software project performance management

Surasak Komchaliaw
DEBI Institute, Curtin University
Perth, Australia
Maxtor_p@hotmail.com

Pornpit Wongthongtham
DEBI Institute, Curtin University
Perth, Australia
P.Wongthongtham@cbs.curtin.edu.au

Abstract— **Several domain experts in the field of software development and project management have commented on the high failure rate of software engineering and project management. A lot of money has been wasted on failed software projects. Additionally, software quality is not improving. Thus the successful management of software projects is critical. It is vital to understand what is important to complete software project on time within budget, and meet user requirements. Many literatures present project failure causes. However, project failure still persists. In this paper we outline software development failure. Then we present two key variables in software project performance management i.e. trust and knowledge sharing.**

Keywords: software development failure; software project performance management; trust; knowledge sharing

I. Introduction

Several domain experts in the field of software development and project management have commented on the failure rate of software engineering and project management e.g.

- Various failure rates for software development projects are up to 85% [1].
- 50% of all software projects are total failures and another 40% are partial failures according to widely quoted surveys in UK, USA, and Norway [2].
- Approximately 31% of corporate software development projects are cancelled before completion and 53% are challenged and cost 180% above their original estimate according to the Standish Group in 1994 [3].
- 46% of software projects are having cost or time overruns or not fully meeting user's requirements and 19% are outright failures according to the Standish Group in 2007 [4].

This has shown that project failure rate is high. A lot of money has been wasted on failed software projects. According to the Standish Group International, roughly 15% never deliver a final product costing $67 billion per year [4]. Stories of software failure attract public attention. Additionally Cerpa and Verner [5] believe that software quality is not improving neither but getting worse. Thus the successful management of software projects is critical.

It is vital to understand what is important to complete software project on time within budget, and meet user requirements. Many literatures [5-11] present project failure causes. However, project failure still persists. In this paper we give overview of software development failure in section 2. Then we present the two key variables in software project performance management in section 3. We discuss and conclude the paper in section 4.

II. Overview of Software Development Failure

A. Teamwork

Teamwork issues refer to issues related to team member development, communication between members, and team management. Team members also include customers, users, and stakeholders. Reason that is most cited for project failure is ineffective communication and coordination among project teams. Other factors include inexperienced project manager, lack of specialized skills, low confidence in team members, insufficient support from manager, inadequately train of team members. DeMarco and Lister argued that aspect of the skills and interactions of software team is most critical and hard to overcome [11].

B. Project Management

Project management issues refer to issues related to project plan and schedule, budget, assessment, control, quality assurance. This includes uncertainty of project milestones, change management, progress report, and project management methodology.

C. Technical Aspects

Technical aspects refer to issues related to software process activities including requirement engineering, design, implementation, testing, validation and verification. It could cause by ambiguous system requirement, incorrect system requirement, wrong development strategies, inappropriate software design, inadequate testing, lack of reusable support of data, code, component, document, etc. However, McCreery and Moranta believe that project challenges were more behavioral and interpersonal than technical [9]. Issues related to communication, collaboration, and team connectedness are more critical.

D. *Project Complexity*

Project complexity issues refer to issues related to the complexity of project requirements. This includes the projects utilizing cutting edge technology and that require high level of technical complexity.

III. Key Variables in Performance Management

From the above overview of software development failure, we identifies two key variables i.e. trust and knowledge sharing as critical influence factors in software development.

A. *Trust*

The concept of trust is related to different and various fields including philosophy, sociology, business, computing. There are number of trust definitions. Mayer et al. define trust as "the willingness of a party to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to the trust or, irrespective of the ability to monitor or control that other party" [12]. Moe and Smite define trust as "the shared perception by the majority of team members that individuals in the team will perform particular actions important to its members and that the individuals will recognize and protect the rights and interests of all the team members engaged in their joint endeavour" [10]. Jarvenpaa et al believe that trust has direct positive effect on cooperation and performance and an increase in trust is likely to have a direct, positive impact on team members' attitudes and perceived outcomes [13]. Giddens [14] sees trust in different view and says that there would be no need of trust if the activities were clearly visible and easy to understand. Hence from his view the prime condition for lack of trust is lack of full information or ambiguous information. As a result trust requires a good knowledge sharing.

Trust can be founded in different ways. The most common way is a direct relationship. In vertical view trust is important to leadership while in horizontal view trust is important for knowledge sharing and team working. In relation to teamwork the two most important dimensions of trust that should be focused are benevolence and competency. Benevolence is related to willingness within teamwork based on the idea that members will not intentionally harm another when given the opportunity to do so. This kind of trust can be positive or negative which members in the team may believe on others willingness to share knowledge and trust level can be in highest level. On the other hand, they may refuse to others willingness and trust can be negative. The second dimension of trust is competency. This kind of trust refers to trusting agent's believe on trusted agent's competency. It describes a relationship in which a member believes that another member is knowledgeable about a given subject area. Competence-based trust also can be negative or positive and members can believe on others ability or they completely refuse others ability in a given subject area.

B. *Knowledge Sharing*

Wang and Yang define knowledge sharing as the action in which people dispread relevant information to others across the organization [11]. Melnik and Maurer divide knowledge sharing into two perspectives i.e. codification approach and personalization approach [15]. Codification approach is based on the notion of knowledge as object [16-19] which can be created, collected, stored, and reused [15]. Personalization approach is based on the notion of knowledge as relationship [20-23] which is uncertain, soft, and embedded in work practices and social relationships [15].

Knowledge sharing in software development can be defined as activities between team members in spreading project data/information/agreement. As seen in Figure 1 knowledge sharing includes communication, updates, advice, problem solving, decision making, issue raising, discussion, etc. over project data/information/agreement.

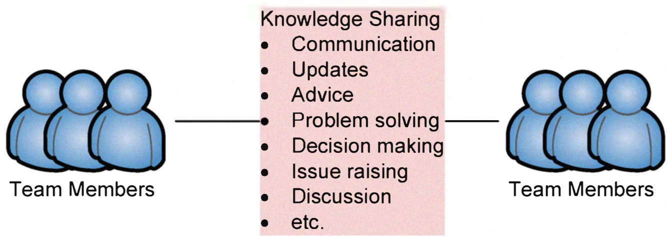

Figure 1. Knowledge sharing definition

Knowledge sharing in software development situation enables team members to enhance their competency and mutually generate new knowledge [15]. Knowledge sharing can be considered by knowledge complexity and knowledge transferability. The complex knowledge and/or long knowledge transfer chain suffer from information distortion and loss which could lead to inefficient knowledge sharing.

IV. Conclusion

Trust is the most important issue to create relationship making value in knowledge sharing. Knowledge itself cannot lead to a success, as knowledge sharing or flow of knowledge is of prime importance in an organization. Knowledge sharing depends on trust between trusted and trustee members in specific knowledge context and specific time slot. Trust level between members has a high impact on software project performance. The future work could include defining a role and measurement of trust and knowledge sharing in the software project performance. A developed framework is required to measure embedded trust in teamwork. Additionally, the framework should be developed and measured the software project performance in a dynamic environment as knowledge and trust are dynamic entities.

References

[1] Jorgensen, M. and K. Molokken-Ostvold, How large are software cost overruns? A review of the 1994 CHAOS report, in Information and Software Technology 48. 2006. p. 297-301.

[2] Gilb, T. No Cure No Pay: How to Contract for Software Services. 2006 [cited accessed 24 May 2006]; Available from: http://roots.dnd.no/repository/05_Gilb_Tom_No_Cure_No_Pay.pdf.

[3] International, S.G. CHAOS: Project failure and success report. 1994 [cited; Available from: http://www.pm2go.com/sample_research/chaos_1994_2.asp.

[4] Rubenstein, D. Standish group report: There's less development chaos today. SD Times [cited Mar. 1, 2007]; Available from: http://www.sdtimes.com/article/story-20070301-01.

[5] Cerpa, N. and J.M. Verner, Why Did Your Project Fail? Communications of the ACM, 2009. 52(12): p. 130-134.

[6] Linberg, K.R., Software developer perceptions about software project failure: a case study. Journal of Systems and Software, 1999. 49(2-3): p. 177-192.

[7] Chen, J.-C. and S.-J. Huang, An empirical analysis of the impact of software development problem factors on software maintainability. Journal of Systems and Software, 2009. 82(6): p. 981-992.

[8] Yong, H., et al. A Neural Network Approach for Software Risk Analysis. in Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06). 2006. Hong Kong, China.: IEEE.

[9] McCreery, J. and V. Moranta. Mapping Team Collaboration in Software Development Projects. in Portland International Conference on Management of Engineering and Technology (PICMET'09). 2009. Portland, Oregon USA.

[10] Moe, N.B. and D. Šmite, Understanding a lack of trust in Global Software Teams: a multiple-case study. 2007, Springer-Verlag Berlin Heidelberg.

[11] Juan-Ru, W. and Y. Jin. Study on Knowledge Sharing Behavior in Software Development Team. in Fourth International Conference on Wireless Communications, Networking and Mobile Computing (WiCom'08). 2008. Dalian, China.

[12] Mayer, R.C., D.J. Davis, and F.D. Shoorman, An Integrative Model of Organizational Trust. Academy of Management Review, 1995. 20(3): p. 709-734.

[13] Jarvenpaa, S.L., T.R. Shaw, and D.S. Staples, Toward contextualized theories of trust: The role of trust in global virtual teams. Information Systems Research 2004. 15(3): p. 250-267.

[14] Giddens, A., The Consequences of Modernity. 1990: Stanford University Press.

[15] Melnik, G. and F. Maurer. Direct Verbal Communication as a Catalyst of Agile Knowledge Sharing. in Proceedings of the Agile Development Conference(ADC'04). 2004. Salt Lake City, UT, USA.

[16] Alavi, M. and D. Leidner, Knowledge Management Systems: Issues, Challenges, and Benefits. Communications of the AIS, 1999. 1(7): p. 2-36.

[17] Hansen, M. and M. Haas, Competing for Attention in Knowledge Markets: Electronic Document Dissemination in a Management Consulting Company. Administrative Science Quarterly, 2001. 46(1): p. 1-28.

[18] Szulanski, G., The Process of Knowledge Transfer: A Diachronic Analysis of Stickiness. Organizational Behavior and Human Decision Processes, 2000, 82(1): p. 9-27.

[19] Zack, M., Managing Codified knowledge. Sloan Management Review, 1999. 40(4): p. 45-58.

[20] Boland, R. and R. Tenkasi, Perspective Making and Perspective Taking in Communities of Knowing. Organization Science, 1995. 6(4): p. 350-372.

[21] Brown, J. and P. Duguid, The Social Life of Information. 2000, Boston, MA: Harvard Business School Press.

[22] Nidumolu, S., M. Subramani, and A. Aldrich, Situated Learning and the Situated Knowledge Web. Journal of Management Information Systems, 2001. 18(1): p. 115-150.

[23] Nonaka, I. and N. Konno, The Concept of "Ba": Building a foundation for Knowledge Creation. California Management Review, 1998. 40(3): p. 40-55.