ISSN No. 0976-5697

Volume 5, No. 7, September-October 2014

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

Steps for Software Project Management and Its Applications

Shikha Vashist, Ayush Gupta
Student, Dronacharya College of Engineering,
Gurgaon, Hr., India-123506
shikha.vashist21@gmail.com

Abstract: In this paper we empirically explore knowledge sharing in a group of project managers. We apply a framework of sensemaking that focuses on how people participate in creating shared meanings. The framework is applied in the analysis of the case where project managers create shared knowledge in a handbook for software project management. The framework provides a rich apprehension of how meaning is created. [3]It explains the importance of impacts and negotiation of the project managers' expectations and experience. The findings add to existing thesis of knowledge sharing in software development. We give in particular with an in [1]depth explanation of the complex process where personal knowledge gradually turns into shared knowledge and some of it in codified form becomes part of the software project management handbook.

Keywords: Knowledge sharing, software development, project management, sense making.

I. INTRODUCTION

Software project management encompasses the knowledge, techniques, and tools necessary to man-age the development of software products. This educational module discusses material that managers need to create a plan for software development, using effective evaluation of size and effort, and to do that plan with attention to productivity and quality. Within this context, some topics such as [7]risk management, alternative life-cycle models, development team organization, and management of technical people are also discussed. Software project management remains different from project management in other, more established fields for a number of reasons: Software is a "brain product" only, unconstrained by the laws of physics or by the limits of manufacturing processes. It is difficult to detect and prevent defects in software. Software can be highly complex. Finally, as a discipline, software development is so young that measurable, effective techniques are not yet available, and those that are available are not well-calibrated. Despite these difficulties, there is an increasing body of knowledge about software project management. This module presents that knowledge and also points to promising new conceptual material.

II. SYSTEM AND METHOD FOR SOFTWARE PROJECT MANAGEMENT

The invention provides a method for computer-implemented management of a project using project management software. The project is defined by a series of development phases wherein each development phase of the project must be evaluated by one or more predetermined standards. Each of the standards defines a set of quality assurance steps to be followed in order to achieve compliance with the standard for each phase. Such a method includes various steps of selecting one of the development phases, displaying a reporting screen containing reporting instructions for the selected development phase, which instructions relate to conformity with the quality assurance steps for that phase according to at least one of the standards, inputting reporting information concerning the selected development phase, and saving the reported information concerning the selected development phase. The instructions on the reporting screen will usually ask if one or more documents relating to compliance with one or more of the quality assurance steps for that phase were completed. The step of saving the reporting information may comprise printing it out, transmitting it to a recipient such as a quality assurance entity, or saving it to a data storage medium for later access.

III. APPARATUSES AND METHODS

A method, computer readable medium and apparatus that manages a software project includes assigning one of one or more virtual hosts in one of one or more workspaces in a development computing device to a remote computing device. The development computing device generates at least one link in the one of the one or more workspaces to at least one of one or more working copies of projects in one of one or more work benches in the one of the one or more virtual hosts. The development computing device generates at least one other link in the one of the one or more workspaces to the linked one of the one or more working copies of projects activated in a running area of the development computing device. The development computing device provides access to the activated one of the one or more working copies of projects to the remote computing device to execute one or more tasks.

IV. AUTOMATIC PROJECT MANAGEMENT APPLICATION

In computer-implemented project management, defined tasks are associated with data elements. Elapsed time intervals are computed, during which data elements are checked-out by project members. A project management software application automatically determines an indication of progress of respective associated tasks, based on the elapsed time intervals. The technique supports concurrent

management of multiple projects, in which data elements are shared. The need to manage large and complex projects has led to the development of many software applications that [2]partially automate project management functions. This software typically has both data storage and management capabilities. It purports to assist in such functions as project planning, budgeting, time and resource allocation and scheduling, cost tracking, and report generation. Some applications are comprehensive, while others focus on particular aspects of project management. For example, in the domains of scheduling and workflow, critical path software has been used for many years. Typically, project management software is electronically linked to its users. Indeed, some applications are groupware-oriented, designed to facilitate collaboration through such modalities as email, video, chat, writing and drawing systems. Many project management applications rely heavily on electronic messaging to record and maintain project status.

V. PROJECT MANAGEMENT AND ASSESSMENT METHOD

The present invention relates to a project assessment method and software featuring processes and tools for the implementation and assessment of large, business critical, and [4]high risk business programs and projects. This method provides a work breakdown structure life cycle; a set of work breakdown structure milestones, selected specifically from health check, monthly status, and certification milestones; assessing each milestone based upon process, delivery, and certification process assessment areas; determining best practices; optional feedback mechanisms; and reports at each step to evaluate and provide structured recommendations regarding the status or execution of a project.

VI. WORK PACKET ENABLED ACTIVE PROJECT MANAGEMENT SCHEDULE

A method for managing projects in a software factory is presented. A project management tool includes an end-to-end project plan for a project to create a software product by using a software factory in a global delivery network. A status block is appended to a work packet that is utilized when executing the project. After initiating the project, an alert is automatically triggered whenever the execution status of the work packet changes. The alert is transmitted to the project management tool to update a project schedule for the project, such that a completion status of the end-to-end project plan reflects a status of a project schedule for the project described by the end-to-end project plan.

VII. SYSTEM AND METHOD FOR ASSESSMENT

A system for monitoring and assessing the performance of a project includes a computer and a software program associated with the computer, with the software program and computer operable in combination to receive project task data from a project management software file, determine current earned value (EV) information from the project task data, and graphically displaying the earned value information. A method for monitoring and assessing the performance of a project may be accomplished by entering task data for each task of the project in a project management software file; obtaining the task data from the project management software file; calculating a current earned value position; obtaining historical earned value positions if any exists; and displaying the current earned value position and any historical earned value positions.

VIII. QUALITY MANAGEMENT FRAMEWORK FOR A SOFTWARE LIFECYCLE

A quality management framework (hereinafter referred as "framework") for a software project lifecycle is provided. The framework enables generation of at least one software program while ensuring compliances with at least one standard. The plurality of items of the library includes tool an organization chart depository, a policy depository, a procedure depository, a life cycle model depository, a guideline depository, a template depository, a process asset depository, a workflow diagram depository and a checklist depository. Each of the items of the library is configured to be updated and accessed. The life cycle model depository of the library is configured to include a modular interface that enables handling of multiple lifecycle [6](V-model, waterfall model, iterative model etc.) models. The framework is configured to allow access the library.

IX. CONCLUSION

In this article we addressed the research questions of how software project managers draw on past experience when they collectively contribute to a process description, in this case a software project management handbook, and how they collectively make sense of its contents and use. The presented exploratory research, which we analysed with a [5]framework for organizational sensemaking, contributes to a better understanding of the phenomenon. This understanding can be summarised as follows:

a. The project managers' construed realities are essential for the creation of a process description.
b. Collisions of their expectations and experiences are important and should be accepted.
c. Negotiation of the process description is necessary.
d. As a result the process description become a shared product, which the project managers can adapt to.

X. REFERENCES

[1]. Basili, V.; Selby, R.; Hutchens, D. (1986): Experimentation in Software Engineering. IEEE Transactions on Software Engineering, 12 (1986), No. 7, pp. 733-743.

[2]. Arent, J., and Nørbjerg, J., (2000). Software Process Improvement as Organizational Knowledge Creation—A multiple case analysis. In: Proceedings of the 33rd Hawaii International Conference on System Sciences, Wailea, Hawaii, p. 11.

[3]. Desouza, K. C., (2003). Barriers to Effective Use of Knowledge Management Systems in Software Engineering. Communication of the ACM, (46:1): 99-101.

[4]. Elzer, P.F. (1989): Management von Softwareprojekten. Informatik-Spektrum, 12 (1989), pp. 181-196.

[5]. Alavi, M., and Leidner, D. E., (2001). Review: Knowledge Management and Knowledge Management Systems:

Conceptual Foundation and Research Issues. MIS Quarterly, (25:1): 107-136.

[6]. Althoff, K.-D., Bomarius, F., and Tautz, C., (2000a). Knowledge Management for Building Learning Software Organizations. Information Systems Frontiers, (2:3-4): 349-367.

[7]. Althoff, K. D., Müller, W., Nick, M., and Snoek, B., (2000b). KM-PEB: An Online Experience Base on Knowledge Management Technology. In: Advances in Case-Based Reasoning. Proc. of the 5th European Workshop on Case-Based Reasoning (EWCBR'00), E. Blanzieri and L. Portinale, editors, Springer, Berlin.