© 2014 IJIRT | Volume 1 Issue 6 | ISSN : 2349-6002

SOFTWARE PROJECT MANAGEMENT

Surbhi Thakur, Srijan S Rawat
Information Technology
Dronacharya College of Engineering, Gurgaon,India

Abstract- **In this paper we discuss that Software projects have several properties that make them very different to other kinds of engineering project and discuss a platform where people share their skills and work on the project. In this platform the project managers make out a handbook in which all the basics and essentials about the project are laid out. It is a rich knowledge about managers experience and view about the work atmosphere to be created and maintained. The research in the handbook enhances the previous knowledge about a software's project management. This handbook becomes the shared knowledge from personal knowledge and aids in the development of the software's project. The basics and the use of expert knowledge in managing a software project are the key factors in a proper development of the project.**

I. INTRODUCTION

Software project management is the art and science of planning and leading of software projects.

The challenges in software development are vast, and considerable research has addressed how these challenges can be met by passing on experience, hard earned knowledge, and well-proven practices to other software developers and managers . This has, in particular, been studied from a knowledge sharing perspective. This,however cannot be done useless there is accurate information and how this is provided will be explored.

II. MAKING SENSE OF SOFTWARE PROJECT MANAGEMENT

A software project has two main activity dimensions:engineering and project management.A software development process is concerned primarily with the production aspect of software development, as opposed to the technical aspect, such as software tools. These processes exist primarily for supporting the management of software development, and are generally skewed toward addressing business concerns. Many software development processes can be run in a similar way to general project management processes. Examples are:

Risk management is the process of measuring or assessing risk and then developing strategies to manage the risk. In general, the strategies employed include transferring the risk to another party, avoiding the risk, reducing the negative effect of the risk, and accepting some or all of the consequences of a particular risk. Risk management in software project management begins with the business case for starting the project, which includes a cost-benefit analysis as well as a list of fallback options for project failure, called a contingency plan.

A subset of risk management that is gaining more and more attention is Opportunity Management, which means the same thing, except that the potential risk outcome will have a positive, rather than a negative impact. Though theoretically handled in the same way, using the term "opportunity" rather than the somewhat negative term "risk" helps to keep a team focused on possible positive outcomes of any given risk register in their projects, such as spin-off projects, windfalls, and free extra resources.

Requirements management is the process of identifying, eliciting, documenting, analyzing, tracing, prioritizing and agreeing on requirements and then controlling change and communicating to relevant stakeholders. New or altered computer system[1] Requirements management, which includes Requirements analysis, is an important part

Project management is the application of knowledge ,skills,tools andtechniques to project activities to meet project requirenments of the software engineering process; whereby business analysts or software developers identify the needs or requirements of a client; having identified these requirements they are then in a position to design a solution.

Change management is the process of identifying, documenting, analyzing, prioritizing and agreeing on changes to scope (project management) and then controlling changes and communicating to relevant stakeholders. Change impact analysis of new or altered scope, which includes Requirements analysis at the change level, is an important part of the software engineering process; whereby business analysts or software developers identify the altered needs or requirements of a client; having identified these requirements they are then in a position to re-design or modify a solution. Theoretically, each change can impact the timeline and budget of a software project, and therefore by definition must include risk-benefit analysis before approval.

Software configuration management is the process of identifying, and documenting the scope itself, which is the software product underway, including all sub-products and changes and enabling communication of these to relevant stakeholders. In general, the processes employed include version control, naming convention (programming), and software archival agreements.

Release management is the process of identifying, documenting, prioritizing and agreeing on releases of software and then controlling the release schedule and communicating to relevant stakeholders. Most software projects have access to three software environments to which software can be released; Development, Test, and Production. In very large projects, where distributed teams need to integrate their work before releasing to users, there will often be more environments for testing, called unit testing, system testing, or integration testing, before release to User acceptance testing (UAT).

A subset of release management that is gaining more and more attention is Data Management, as obviously the users can only test based on data that they know, and "real" data is only in the software environment called "production". In order to test their work, programmers must therefore also often create "dummy data" or "data stubs". Traditionally, older versions of a production system were once used for this purpose, but as companies rely more and more on outside contributors for software development, company data may not be released to development teams. In complex environments, datasets may be created that are then migrated across test environments according to a test release schedule, much like the overall software release schedule.

III. THEORETICAL FRAMEWORK

Structure of Development Plan

1. Introduction brief intro to project —references to requirements spec

2. Project organisation intro to organisations, people, and their roles

3. Risk Analysis what are the key risks to the project?

4. Hardware and software resources what h/ware and s/ware resources will be required for the project and when?

5. Work breakdown the project divided into activities, milestones, deliverables; dependencies between tasks etc
6. Project schedule actual time required —allocation of dates

7. Reporting and progress measurement mechanisms to monitor progress.

Work Breakdown

_ There are many ways of breaking down the activities in a project, but the most usual is into:

– work packages;

– tasks;

– deliverables;

– milestones.
- A workpackage is a large, logically distinct section of work:

– typically at least 12 months duration;

– may include multiple concurrent activities;

– independent of other activities;

– but may depend on, or feed into other activities;

– typically allocated to a single team.
_ A task is typically a much smaller piece of work:

A part of a workpackage.

– typically 3–6 person months effort;

– may be dependent on other concurrent activities;
– typically allocated to a single person.

_ A deliverable is an output of the project that can meaningfully be assessed.

Examples:

– a report (e.g., requirements spec);
– code (e.g., alpha tested product).

Deliverables are indicators (but only indicators) of progress.

_ A milestone is a point at which progress on the project may be assessed.

Typically a major turning point in the project.

EXAMPLES:

– delivery of requirements spe. . .
– work packages are numbered WP1, WP2, . . . ;
– tasks are numbered T1.1, T1.2, etc, the first number is the number of the workpackage; the second is a sequence number.
– deliverables are numbered D1.1, D1.2, etc
– milestones are numbered M1, M2 etc.

_ For each workpackage & task, it is usual to document:

– brief description;

– earliest start date;
– earliest end date;

– total person months effort;

– pre-requisite WPs or tasks;

– dependent WPs or tasks;

– who is responsible.

3.1 Creating

The cognitive processes in the first period of creating are based on the understanding that or-ganizational members notice in particular the parts of the ongoing flow of information that they are exposed to.

IV. CONCLUSION

In this article we addressed the research questions of how software project managers draw on software project issues and how to work on them collectively.

The framework is usually important to work in an organized way.

REFERENCES

[1] 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.

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

[3] 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.

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