Journal Of Harmonized Research (JOHR)

Journal Of Harmonized Research in Engineering
1(2), 2013, 87-90

ISSN 2347 – 7393

Review Article

SOFTWARE PROJECT MANAGEMENT

Shweta Garg, Shrishti Vashist, Shruti Aggarwal

CSE department, Dronacharya College of Engineering,
Gurgaon, India

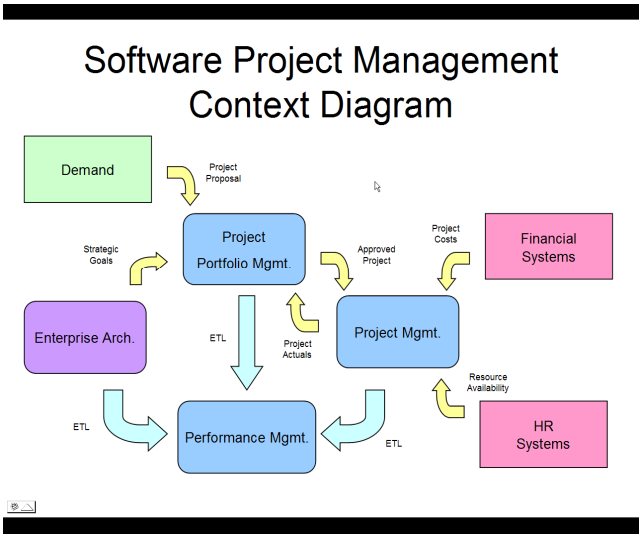
Abstract:
This article is about software engineering project management, the universality of a five-element management model, and the application of the five-element model to software engineering project management.Management involves the activities undertaken by one or more persons for the purpose of planning and controlling the activities of others in order to achieve objectives that could not be achieved by the others acting alone. The five-element management model categorizes management functions into planning, organizing, staffing, leading, and controlling.Project management is a system of management procedures, practices, and know-how needed to manage a project successfully. Know-how, in this case, means the skill, background, and experience required to apply knowledge to practical situations.

Keywords: Sculpture-the art of craving, Contigency-dependence on chance, Eliciting-to draw, Severity- harshness, Typos-typographical error
Abbreviation: SPM-software project management, UAT-user acceptance testing

1. Introduction

Software project management(SPM) is the sculpture and science of planning and chief software projects. It is a sub-discipline of project management in which software projects are planned, implemented, monitored and controlled.

For Correspondence:
shivigarg101ATgmail.com
Received on: October 2013.
Accepted after revision: November 2013.
Downloaded from: www.johronline.com



2. History

In the 1970s and 1980s, the software industry grew very swiftly, as computer companies quickly recognized the relatively low cost of software production compared to hardware fabrication and circuitry. To manage new development efforts, companies applied the established project management methods, but project schedules slipped during test runs, especially when confusion occurred in the gray zone between the user specifications and the delivered software. To be able to avoid these problems, *software* project management methods focused on matching user requirements to delivered products, in a method known now as the waterfall model.

As the industry has matured, analysis of software project management failures has shown that the following are the most common causes.

1. Unrealistic or unarticulated project goals
2. Inaccurate estimates of needed resources
3. Badly defined system requirements
4. Poor reporting of the project's status
5. Unmanaged risks
6. Poor communication among customers, developers, and users
7. Use of immature technology
8. Inability to handle the project's complexity
9. Sloppy development practices
10. Poor project management
11. Stakeholder politics
12. Commercial pressures

3. Software development process

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.

- 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 Requirements management, which includes Requirements analysis, is an important part of the software engineering process;

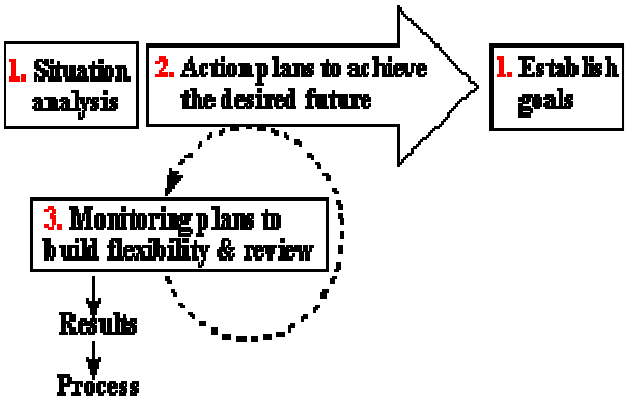
whereby business analysts or software developers identify the needs or requirements of a client; having identified these requirements they are then in a position to design a solution.

- 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 Requirement 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).

3.1 Project planning, monitoring and control

The purpose of project planning is to identify the scope of the project, estimate the work involved, and create a project schedule. Project planning begins with requirements that define the software to be developed. The project plan is then developed to describe the tasks that will lead to completion.

The purpose of project monitoring and control is to keep the team and management up to date on the project's progress. If the project deviates from the plan, then the project manager can take action to correct the problem. Project monitoring and control involves status meetings to gather status from the team. When changes need to be made, change control is used to keep the products up to date.



3.2 Severity levels

Issues are often categorized in terms of severity levels. Different companies have different definitions of severities, but some of the most common ones are:

High

The bug or issue affects a crucial part of a system, and must be fixed in order for it to resume normal operation.

Medium

The bug or issue affects a minor part of a system, but has some impact on its operation. This severity level is assigned when a non-central requirement of a system is affected.

Low

The bug or issue affects a minor part of a system, and has very little impact on its operation. This severity level is assigned when a non-central requirement of a system (and with lower importance) is affected.

Cosmetic
The system works correctly, but the appearance does not match the expected one. For example: wrong colors, too much or too little spacing between contents, incorrect font sizes, typos, etc. This is the lowest severity issue.

3.3 Philosophy

As a subdiscipline of project management, some regard the management of software development akin to the management of manufacturing, which can be performed by someone with management skills, but no programming skills. John C. Reynolds rebuts this view, and argues that software development is entirely design work, and compares a manager who cannot program to the managing editor of a newspaper who cannot write.

References:-

1. Stellman, Andrew; Greene, Jennifer (2005). *Applied Software Project Management*. O'Reilly Media. ISBN 978-0-596-00948-9.
2. IEEE magazine article "Why Software Fails"
3. John C. Reynolds, *Some thoughts on teaching programming and programming languages.*