SOFTWARE PROJECT MANAGEMENT

RONIT YADAV

SACHIN SOLANKI

SAHIB ARORA

➢ ABSTRACT

Software project management is the art and science of planning and leading software projects. It is a sub-discipline of project management in which software projects are planned, implemented, monitored and controlled.

➢ History

In the 1970s and 1980s, the software industry grew very quickly, as computer companies quickly recognized the relatively low cost of software production compared to hardware production 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. Insufficient end-user involvement
2. Poor communication among customers, developers, users and project managers
3. Unrealistic or unarticulated project goals
4. Inaccurate estimates of needed resources
5. Badly defined or incomplete system requirements and specifications
6. Poor reporting of the project's status
7. Poorly managed risks
8. Use of immature technology
9. Inability to handle the project's complexity
10. Sloppy development practices

11. Stakeholder politics (e.g. absence of executive support, or politics between the customer and end-users)
12. Commercial pressures

The first five items in the list above show the difficulties articulating the needs of the client in such a way that proper resources can deliver the proper project goals. Specificsoftware project management tools are useful and often necessary, but the true art in software project management is applying the correct method and then using tools to support the method. Without a method, tools are worthless. Since the 1960s, several proprietary software project management methods have been developed by software manufacturers for their own use, while computer consulting firms have also developed similar methods for their clients. Today software project management methods are still evolving, but the current trend leads away from the waterfall model to a more cyclic project delivery model that imitates a software development process.

➢ 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:

- ✓ Interpersonal communication and conflict management and resolution. Active, frequent and honest communication is the most important factor in increasing the likelihood of project success and mitigating problematic projects. The development team should seek end-user involvement and encourage user input in the development process. Not having users involved can lead to misinterpretation of requirements, insensitivity to changing customer needs, and unrealistic expectations on the part of the client. Software developers, users, project managers, customers and project sponsors need to communicate regularly and frequently. The information gained from these discussions allows the project team to analyze the strengths, weaknesses, opportunities and threats (SWOT) and to act on that information to benefit from opportunities and to minimize threats. Even bad news may be good *if* it is communicated relatively early, because problems can be mitigated if they are not discovered too late. For example, casual conversation with users, team members, and other stakeholders may often surface potential problems sooner than formal meetings. All communications need to be intellectually honestand authentic, and regular, frequent, high quality criticism of development work is necessary, as long as it is provided in a calm, respectful, constructive, non-accusatory, non-angry fashion. Frequent casual communications between developers and end-users, and between project managers and

clients, are necessary to keep the project relevant, useful and effective for the end-users, and within the bounds of what can be completed. Effective interpersonal communication and conflict management and resolution are the key to software project management. *No methodology or process improvement strategy can overcome serious problems in communication or mismanagement of interpersonal conflict.* Moreover, outcomes associated with such methodologies and process improvement strategies are enhanced with with better communication. The communication must focus on whether the team understands the project charter and whether the team is making progress towards that goal. End-users, software developers and project managers must frequently ask the elementary, simple questions that help identify problems before they fester into near-disasters. *While end-user participation, effective communication and teamwork are not sufficient, they are necessary to ensure a good outcome, and their absence will almost surely lead to a bad outcome.*

- ✓ 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 Requirements management, which includes Requirements analysis, is an important part of thesoftware 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.

➢ 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.

▪ Issue

In computing, the term **issue** is a unit of work to accomplish an improvement in a system. An issue could be a bug, a requested feature, task, missing documentation, and so forth.

For example, OpenOffice.org used to call their modified version of BugZilla IssueZilla. As of September 2010, they call their system Issue Tracker.

-

The word "issue" is also used as synonym for "problem," as in other English usage. Problems occur from time to time and fixing them in a timely fashion is essential to achieve correctness of a system and avoid delayed deliveries of products.

➢ 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.

In many software companies, issues are often investigated by Quality Assurance Analysts when they verify a system for correctness, and then assigned to the developer(s) that are responsible for resolving them. They can also be assigned by system users during the User Acceptance Testing (UAT) phase.

Issues are commonly communicated using Issue or Defect Tracking Systems. In some other cases, emails or instant messengers are used.

➢ 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 amanager who cannot program to the managing editor of a newspaper who cannot write.[6]

➢ See also

- Estimation

- Estimation in software engineering
- Incremental funding methodology
- Issue management
- Project management
- Risk management
- Software development process
- Software engineering
- Anti-pattern — A number of anti-patterns (ineffective and/or counter-productive design patterns) directly relate to software project management and the software development process in general.
- NNPP Net-Negative-Producing-Programmer; jargon related to concepts from Software Project Management