Software Project Management

MANAGING SOFTWARE PROJECTS

THE MANAGEMENT SPECTRUM:-

The People

- Effective software project management focuses on the four P's: people, product, process, and project.
- "people factor" is so important that the Software Engineering Institute has developed a *People Capability Maturity Model (People-CMM).*
- "every organization needs to continually improve its ability to attract, develop, motivate, organize, and retain the workforce needed to accomplish its strategic business objectives".

MANAGING SOFTWARE PROJECTS

- The people capability maturity model defines the following key practice areas for software people: staffing, communication and coordination, work environment, performance management, training, compensation, competency analysis and development, career development, workgroup development, team/culture development, and others.
- Organizations that achieve high levels of People-CMM maturity have a higher likelihood of implementing effective software project management practices.

MANAGING SOFTWARE PROJECTS

- The People-CMM is a companion to the *Software Capability Maturity Model–Integration that guides organizations in the creation of a mature* software process.

MANAGING SOFTWARE PROJECTS

The Product:-

- Before a project can be planned, product objectives and scope should be established, alternative solutions should be considered, and technical and management constraints should be identified
- Without this information, it is impossible to define reasonable (and accurate) estimates of the cost, an effective assessment of risk, a realistic breakdown of project tasks, or a manageable project schedule that provides a meaningful indication of progress.

MANAGING SOFTWARE PROJECTS

- As a software developer, you and other stakeholders must meet to define product objectives and scope.
- In many cases, this activity begins as part of the system engineering or business process engineering and continues as the first step in software requirements engineering
- Once the product objectives and scope are understood, alternative solutions are considered.
- It enable managers and practitioners to select a “best” approach, given the constraints imposed by delivery deadlines, budgetary restrictions, personnel availability, technical interfaces, and other factors.

MANAGING SOFTWARE PROJECTS

The Process :-

- software process provides the framework from which a comprehensive plan for software development can be established.
- A number of different task sets—tasks, milestones, work products, and quality assurance points—enable the framework activities to be adapted to the characteristics of the software project and the requirements of the project team.
- Finally, activities—such as software quality assurance, software configuration management, and measurement—overlay the process model and These activities are independent of any one framework activity and occur throughout the process.

MANAGING SOFTWARE PROJECTS

The Project:-

- To avoid project failure, a software project manager and the software engineers who build the product must avoid a set of common warning signs, understand the critical success factors that lead to good project management, and develop a commonsense approach for planning, monitoring, and controlling the project

MANAGING SOFTWARE PROJECTS

PEOPLE:-

- In a study published by the IEEE, the engineering vice presidents of three major technology companies were asked what was the most important contributor to a successful software project. They answered in the following way:
  - It’s not the tools that we use, it’s the people.
  - The most important ingredient that was successful on this project was having smart people
  - The only rule I have in management is to ensure I have good people—real good people

MANAGING SOFTWARE PROJECTS

The Stakeholders:-

- Senior managers who define the business issues that often have a significant influence on the project.
- Project (technical) managers who must plan, motivate, organize, and control the practitioners who do software work.
- Practitioners who deliver the technical skills that are necessary to engineer a product or application.
- Customers who specify the requirements for the software to be engineered and other stakeholders who have a peripheral interest in the outcome.
- End users who interact with the software once it is released for production use.

MANAGING SOFTWARE PROJECTS

Team Leaders

- In an excellent book of technical leadership, Jerry Weinberg suggests an MOI model of leadership:
- **Motivation :** The ability to encourage technical people to produce to their best ability.
- **Organization :** The ability to mold existing processes (or invent new ones) that will enable the initial concept to be translated into a final product.
- **Ideas or innovation :** The ability to encourage people to create and feel creative even when they must work within bounds established for a particular software product or application.

MANAGING SOFTWARE PROJECTS

- Another view of the characteristics that define an effective project manager emphasizes four key traits:
  - Problem solving: An effective software project manager can diagnose the technical and organizational issues that are most relevant, systematically structure a solution or properly motivate other practitioners to develop the solution, apply lessons learned from past projects to new situations, and remain flexible enough to change directions if initial attempts at problem solution are fruitless.
  - Managerial identity: A good project manager must take charge of the project. She must have the confidence to assume control when necessary and the assurance to allow good technical people to follow their instincts.

- Achievement: A competent manager must reward initiative and accomplishment to optimize the productivity of a project team. She must demonstrate through her own actions that controlled risk taking will not be punished.
- Influence and team building: An effective project manager must be able to “read” people; she must be able to understand verbal and nonverbal signals and react to the needs of the needs of the people sending these signals. The manager must remain under control in high-stress situations.

MANAGING SOFTWARE PROJECTS

The Software Team:

- The “best” team structure depends on the management style of your organization, the number of people who will populate the team and their skill levels, and the overall problem difficulty.

MANAGING SOFTWARE PROJECTS

- Seven project factors that should be considered when planning the structure of software engineering teams:
  - Difficulty of the problem to be solved
  - “Size” of the resultant program(s) or lines of code
  - Time that the team will stay together (team lifetime)
  - Degree to which the problem can be modularized
  - Required quality and reliability of the system to be built
  - Rigidity of the delivery date
  - Degree of sociability (communication) required for the project

MANAGING SOFTWARE PROJECTS

Constantine suggests four “organizational paradigms” for software engineering teams:

1. **A closed paradigm** structures a team along a traditional hierarchy of authority. Such teams can work well when producing software that is quite similar to past efforts, but they will be less likely to be innovative when working within the closed paradigm.
2. **A random paradigm** structures a team loosely and depends on individual initiative of the team members. When innovation or technological breakthrough is required, teams following the random paradigm will excel. But such teams may struggle when “orderly performance” is required.

MANAGING SOFTWARE PROJECTS

3. An open paradigm attempts to structure a team in a manner that achieves some of the controls associated with the closed paradigm but also much of the innovation that occurs when using the random paradigm.

- Work is performed collaboratively, with heavy communication and consensus-based decision making the trademarks of open paradigm teams. Open paradigm team structures are well suited to the solution of complex problems but may not perform as efficiently as other teams.

MANAGING SOFTWARE PROJECTS

4. **A synchronous paradigm** relies on the natural compartmentalization of a problem and organizes team members to work on pieces of the problem with little active communication among themselves.

MANAGING SOFTWARE PROJECTS

- One of the earliest software team organizations was a paradigm structure originally called the chief programmer team. This structure was first proposed by Harlan Mills.
- The nucleus of the team was composed of a senior engineer (the chief programmer), who plans, coordinates, and reviews all technical activities of the team.
- Technical staff (normally two to five people), who conduct analysis and development activities; and a backup engineer, who supports the senior engineer in his or her activities and can replace the senior engineer with minimum loss in project

MANAGING SOFTWARE PROJECTS

- The chief programmer may be served by one or more specialists (e.g., telecommunications expert, database designer), support staff (e.g., technical writers, clerical personnel), and a software librarian.

MANAGING SOFTWARE PROJECTS

To achieve a high-performance team:

- Team members must have trust in one another.
- The distribution of skills must be appropriate to the problem.
- Mavericks may have to be excluded from the team, if team cohesiveness is to be maintained.

Regardless of team organization, the objective for every project manager is to help create a team that exhibits cohesiveness.

MANAGING SOFTWARE PROJECTS

- DeMarco and Lister contend that members of jelled teams are significantly more productive and more motivated than average.
- But not all teams jell. In fact, many teams suffer from what Jackman calls “team toxicity.” She defines five factors that “foster a potentially toxic team environment”:

(1) A frenzied work atmosphere

(2) High frustration that causes friction among team members

(3) “Fragmented or poorly coordinated” software process

(4) An unclear definition of roles on the software team

(5) “Continuous and repeated exposure to failure.”

MANAGING SOFTWARE PROJECTS

- In addition to the five toxins a software team often struggles with the differing human traits of its members. Some team members are extroverts, others are introverts.
- Some people gather information intuitively. Others process information linearly, collecting and organizing minute details from the data provided.
- Some team members are comfortable making decisions only when a logical, orderly argument is presented. Others are intuitive, willing to make a decision based on “feel.”
- Some practitioners want a detailed schedule populated by organized tasks that enable them to

MANAGING SOFTWARE PROJECTS

- Some work hard to get things done long before a milestone date, thereby avoiding stress as the date approaches, while others are energized by the rush to make a last-minute deadline.
- It is important to note that recognition of human differences is the first step toward creating teams that jell.

MANAGING SOFTWARE PROJECTS

Agile Teams:

- Over the past decade, agile software development has been suggested as an antidote to many of the problems that have plagued software project work.
- The agile philosophy encourages customer satisfaction and early incremental delivery of software, small highly motivated project teams, informal methods, minimal software engineering work products, and overall development simplicity.

MANAGING SOFTWARE PROJECTS

- The small, highly motivated project team, also called an *agile team, adopts many* of the characteristics of successful software project teams.
- However, the agile philosophy stresses individual (team member) competency coupled with group collaboration as critical success factors for the team.
- If the people on the project are good enough, they can use almost any process and accomplish their assignment.
- To make effective use of the competencies of each team member and to foster effective

MANAGING SOFTWARE PROJECTS

- A self-organizing team does not necessarily maintain a single team structure but instead uses elements of Constantine's random, open, and synchronous paradigms
- Many agile process models (e.g., Scrum) give the agile team significant autonomy to make the project management and technical decisions required to get the job done.
- Planning is kept to a minimum, and the team is allowed to select its own approach (e.g., process, methods, tools), constrained only by business requirements and organizational standards. As the project proceeds, the team self-organizes to focus individual competency in a way that is most

MANAGING SOFTWARE PROJECTS

- To accomplish this, an agile team might conduct daily team meetings to coordinate and synchronize the work that must be accomplished for that day.
- Based on information obtained during these meetings, the team adapts its approach in a way that accomplishes an increment of work.

MANAGING SOFTWARE PROJECTS

Coordination and Communication Issues

- There are many reasons that software projects get into trouble. The scale of many development efforts is large, leading to complexity, confusion, and significant difficulties in coordinating team members.
- New software must communicate with existing software and conform to predefined constraints imposed by the system or product.

MANAGING SOFTWARE PROJECTS

- To deal with them effectively, you must establish effective methods for coordinating the people who do the work.
- To accomplish this, mechanisms for formal and informal communication among team members and between multiple teams must be established.
- Formal communication is accomplished through “writing, structured meetings, and other relatively non-interactive and impersonal communication channels”.
- Informal communication is more personal. Members of a software team share ideas on basis, ask for help as problems arise, and interact with one another on a daily basis.

MANAGING SOFTWARE PROJECTS

THE PRODUCT

- A detailed analysis of software requirements would provide necessary information for estimates, but analysis often takes weeks or even months to complete
- One must examine the product and the problem it is intended to solve at the very beginning of the project.

MANAGING SOFTWARE PROJECTS

Software Scope:-

- The first software project management activity is the determination of software scope. Scope is defined by
- **Context :-** How does the software to be built fit into a larger system, product, or business context, and what constraints are imposed?
- **Information objectives :-** What customer-visible data objects are produced as output from the software? What data objects are required for input?
- **Function and performance :-** What function does the software perform to transform input data into output? Are any special performance characteristics to be addressed?

MANAGING SOFTWARE PROJECTS

- Software project scope must be unambiguous and understandable at the management and technical levels.
- The statement of the software scope must be bounded, That is, quantitative data are stated explicitly, constraints and/or limitations are noted, and mitigating factors are described.

MANAGING SOFTWARE PROJECTS

Problem Decomposition:-

- Problem decomposition, sometimes called *partitioning or problem elaboration, is an* activity that sits at the core of software requirements.
- During the scoping activity no attempt is made to fully decompose the problem. Rather, decomposition is applied in two major areas:

(1) The functionality and content (information) that must be delivered

(2) The process that will be used to deliver it.

MANAGING SOFTWARE PROJECTS

- Human beings tend to apply a divide-and-conquer strategy when they are confronted with a complex problem. A complex problem is partitioned into smaller problems that are more manageable.
- This is the strategy that applies as project planning begins.
- Major content or data objects are decomposed into their constituent parts, providing a reasonable understanding of the information to be produced by the software.

MANAGING SOFTWARE PROJECTS

THE PROCESS

- The framework activities that characterize the software process are applicable to all software projects. The problem is to select the process model that is appropriate for the software to be engineered by your project team.
- Team must decide which process model is most appropriate for

  (1) the customers who have requested the product and the people who will do the work

  (2) The characteristics of the product itself

  (3) The project environment in which the software team works.

MANAGING SOFTWARE PROJECTS

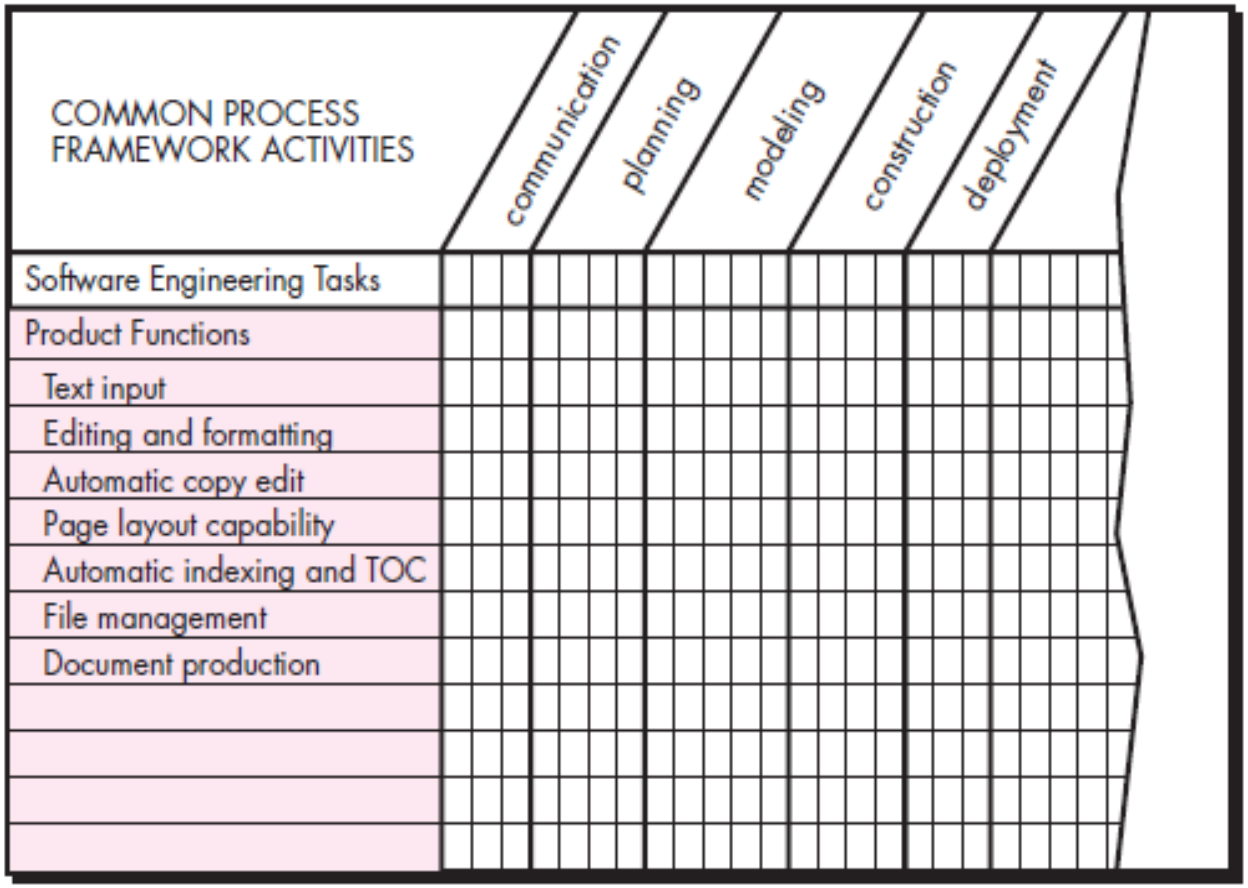
- Once the preliminary plan is established, process decomposition begins. That is, a complete plan, reflecting the work tasks required to populate the framework activities must be created.

MANAGING SOFTWARE PROJECTS

Melding the Product and the Process:-

- Project planning begins with the melding of the product and the process. Each function to be engineered by your team must pass through the set of framework activities that have been defined for your software organization.
- Assume that the organization has adopted the generic framework activities— **communication, planning, modeling, construction, and deployment**
- The team members who work on a product function will apply each of the framework activities to it.

MANAGING SOFTWARE PROJECTS



MANAGING SOFTWARE PROJECTS

- Each major product function is listed in the left-hand column.
- Framework activities are listed in the top row. Software engineering work tasks (for each framework activity) would be entered in the following row.
- The job of the project manager (and other team members) is to estimate resource requirements for each matrix cell, start and end dates for the tasks associated with each cell, and work products to be produced as a consequence of each task.

MANAGING SOFTWARE PROJECTS

Process Decomposition:

- A software team should have a significant degree of flexibility in choosing the software process model that is best for the project.
- relatively small project that is similar to past efforts might be best accomplished using the linear sequential approach.
- If the deadline is so tight that full functionality cannot reasonably be delivered, an incremental strategy might be best.
- Similarly, projects with other characteristics (e.g., uncertain requirements, difficult customers) will lead to the selection of other process models.

MANAGING SOFTWARE PROJECTS

- The process framework is invariant and serves as the basis for all work performed by a software organization.
- Process decomposition commences when the project manager asks, “How do we accomplish this framework activity?”

1. Develop list of clarification issues.
2. Meet with stakeholders to address clarification issues.
3. Jointly develop a statement of scope.
4. Review the statement of scope with all concerned.
5. Modify the statement of scope as required.

MANAGING SOFTWARE PROJECTS

- Now, consider a more complex project, which has a broader scope. Such a project might require the following work tasks for the **communication:**

MANAGING SOFTWARE PROJECTS

1. Review the customer request.
2. Plan and schedule a formal, facilitated meeting with all stakeholders.
3. Conduct research to specify the proposed solution and existing approaches.
4. Prepare a “working document” and an agenda for the formal meeting.
5. Conduct the meeting.
6. Jointly develop mini-specs that reflect data, functional, and behavioral features of the software. Alternatively, develop use cases that describe the software from the user’s point of view.
7. Review each mini-spec or use case for correctness, consistency, and lack of ambiguity.
8. Assemble the mini-specs into a scoping document.
9. Review the scoping document or collection of use cases with all concerned.

MANAGING SOFTWARE PROJECTS

THE PROJECT:-

In order to manage a successful software project, you have to understand what can go wrong so that problems can be avoided.

MANAGING SOFTWARE PROJECTS

1. Software people don’t understand their customer’s needs.
2. The product scope is poorly defined.
3. Changes are managed poorly.
4. The chosen technology changes.
5. Business needs change .
6. Deadlines are unrealistic.
7. Users are resistant.
8. Sponsorship is lost [or was never properly obtained].
9. The project team lacks people with appropriate skills.

MANAGING SOFTWARE PROJECTS

How does a manager act to avoid the problems just noted? five-part commonsense approach to software projects:

1. Start on the right foot. This is accomplished by working hard to understand the problem that is to be solved and then setting realistic objectives and expectations for everyone who will be involved in the project.
   - It is reinforced by building the right team and giving the team the authority, and technology needed to do the job.

MANAGING SOFTWARE PROJECTS

2. Maintain momentum. Many projects get off to a good start and then slowly disintegrate. To maintain momentum, the project manager must provide incentives , the team should emphasize quality in every task it performs.
3. Track progress. For a software project, progress is tracked as work products (e.g., models, source code, sets of test cases) are produced and approved (using technical reviews) as part of a quality assurance activity.

MANAGING SOFTWARE PROJECTS

4. Make smart decisions. In essence, the decisions of the project manager and the software team should be to “keep it simple.” Whenever possible, decide to use existing software components or patterns, decide to identify and then avoid obvious risks, and decide to allocate more time than you think is needed to complex or risky.
5. Conduct a postmortem analysis. Establish a consistent mechanism for extracting lessons learned for each project. Evaluate the planned and actual schedules, get feedback from team members and customers, and record findings .

MANAGING SOFTWARE PROJECTS

THE W5HH PRINCIPLE

- Barry suggests an approach that addresses project objectives, milestones and schedules, responsibilities, management and technical approaches, and required resources.
- It the W5HH Principle, after a series of questions that lead to a definition of key project characteristics and the resultant project plan:

MANAGING SOFTWARE PROJECTS

- Why is the system being developed?

  All stakeholders should assess the validity of business reasons for the software work. Does the business purpose justify the expenditure of people, time, and money?

- What will be done?

  The task set required for the project is defined.

- Who is responsible for a function?

  The role and responsibility of each member of the software team is defined.

- Where are they located organizationally?

  Not all roles and responsibilities reside within software practitioners. The customer, users, and other stakeholders also have responsibilities.

- How will the job be done technically and managerially?

  Once product scope is established, a management and technical strategy for the project must be defined.

- How much of each resource is needed?

  The answer to this question is derived by developing estimates based on answers to earlier questions.

MANAGING SOFTWARE PROJECTS

CRITICAL PRACTICES

- There is a list of “critical software practices for performance-based management.”
- Metric-based project management
- Empirical cost and schedule estimation
- Earned value tracking
- Defect tracking against quality targets
- And people aware management