Software Project Management

CONTENTS

I. Introduction to Software Project Management and its need.

(Question: explain the need of software project management- 4 Marks)

Software projects have several properties that make them very different to other kinds of engineering project.

1. **The product is intangible**: It's hard to claim a bridge is 90%complete if there is not 90% of the bridge there. It is easy to claim that a software project is 90% complete, even if there are no visible outcomes.
2. **We don't have much experience**. Software engineering is a new discipline, and so we simply don't have much understanding of how to engineer large scale software projects.
3. **Large software projects are often "bespoke"**. Most large software systems are one-off, with experience gained in one project being of little help in another.
4. **The technology changes very quickly**. Most large software projects employ new technology; for many projects.

II. The Management Spectrum – 4 Ps and their Significance

(Question: Explain the 4P's of management spectrum- 4 Marks)

Effective software project management focuses on these items (in this order)

1. **The people**

III. Deals with the cultivation of motivated, highly skilled people

   i. Consists of the stakeholders, the team leaders, and the software team

2. **The product**
   i. Product objectives and scope should be established before a project can be planned
3. **The process**
   i. The software process provides the framework from which a comprehensive plan for software development can be established
4. **The project**
   i. Planning and controlling a software project is done for one primary reason…it is the only known way to manage complexity
   ii. In a 1998 survey, 26% of software projects failed outright, 46% experienced cost and schedule overruns

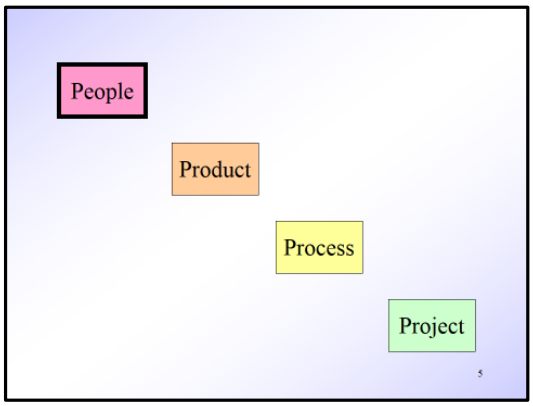

Figure 1: 4 P's of Project Management

IV. Project Scheduling

1. Concept of Project Scheduling

i. Changing customer requirements that are not reflected in schedule changes.
ii. An honest underestimate of the amount of effort and/or the number of resources that will be required to do the job.
iii. Predictable and/or unpredictable risks that were not considered when the project commenced.
iv. Technical difficulties that could not have been foreseen in advance.

2. Factors that delay Project Schedule

Although there are many reasons why software is delivered late, most can be traced to one or more of the following root causes:

i. An unrealistic deadline established by someone outside the software development group and forced on managers and practitioners within the group.
ii. Changing customer requirements that are not reflected in schedule changes.
iii. An honest underestimate of the amount of effort and/or the number of resources that will be required to do the job.
iv. Predictable and/or unpredictable risks that were not considered when the project commenced.
v. Technical difficulties that could not have been foreseen in advance.
vi. Human difficulties that could not have been foreseen in advance.
vii. Miscommunication among project staff that results in delays.
viii. A failure by project management to recognize that the project is falling behind schedule and a lack of action to correct the problem.

3. Principles of Project Scheduling

i. Compartmentalization: The project must be compartmentalized into a number of manageable activities and tasks.
ii. Interdependency: The interdependency of each compartmentalized activity or task must be determined.
iii. Time allocation: Each task to be scheduled must be allocated some number of work units (e.g., person-days of effort).
iv. Effort validation: the project manager must ensure that no more than the allocated number of people have been scheduled at any given time.
v. Defined responsibilities: Every task that is scheduled should be assigned to a specific team member
vi. Defined outcomes: Every task that is scheduled should have a defined outcome.
vii. Defined milestones: Every task or group of tasks should be associated with a project milestone.
viii. A milestone is accomplished when one or more work products has been reviewed for quality and has been approved.

4. Project Scheduling Techniques- Concept of Gantt Chart, PERT, CPM

1. Gantt Chart

i. A project control technique
ii. Defined by Henry L. Gantt
iii. Used for several purposes, including scheduling, budgeting, and resource planning.
iv. When creating a software project schedule, the planner begins with a set of tasks.
v. If automated tools are used, the work breakdown is input as a task network or task outline.
vi. Effort, duration, and start date are then input for each task. In addition, tasks may be assigned to specific individuals.
vii. As a consequence of this input, a timeline chart is generated also called Gantt chart.
viii. A timeline chart can be developed for the entire project.
ix. Alternatively, separate charts can be developed for each project function or for each individual working on the project.

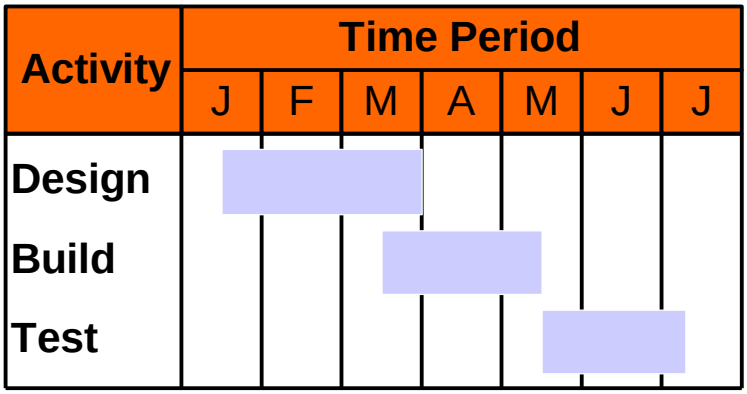

Figure 2: Gantt chart

2. Differentiate between Pert and CPM

i. Network techniques
ii. Developed in 1950's
iii. CPM by DuPont for chemical plants
iv. PERT by U.S. Navy for Polaris missile
v. Consider precedence relationships & interdependencies
vi. Each uses a different estimate of activity times

Benefits of PERT/CPM

i. Useful at many stages of project management
ii. Mathematically simple
iii. Use graphical displays
iv. Give critical path & slack time
v. Provide project documentation
vi. Useful in monitoring costs

Disadvantage of PERT and CPM

i. Clearly defined, independent, & stable activities
ii. Specified precedence relationships
iii. Activity times (PERT) follow beta distribution
iv. Subjective time estimates
v. Over emphasis on critical path

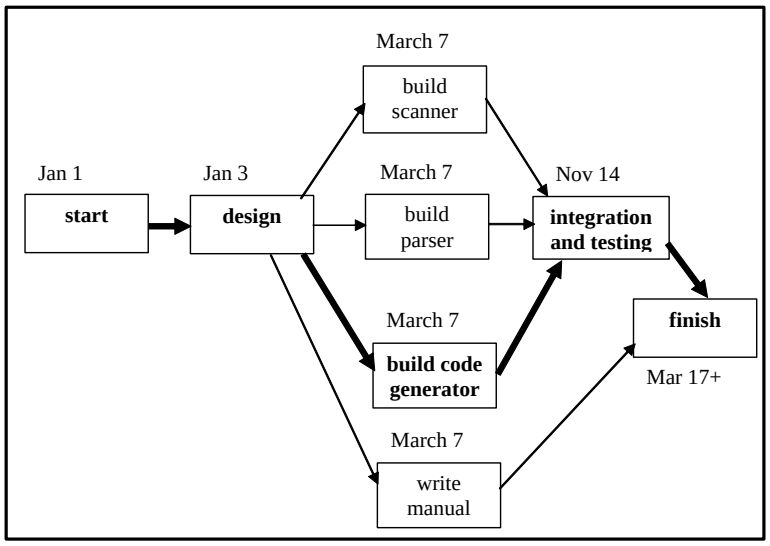

Figure 3: PERT

V. Concept of Task Network

1. Individual tasks and subtasks have interdependencies based on their sequence.
2. A task network is a graphic representation of the task flow for a project.
3. A schematic network for a concept development project.
4. Critical path: The tasks on a critical path must be completed on schedule to make the whole project on schedule.

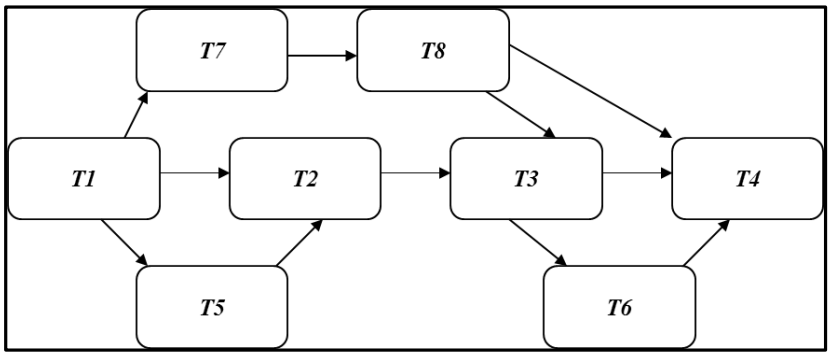

Figure 4: Task Network

VI. Ways of Project Tracking

1. Scheduling of a software project does not differ greatly from scheduling of any multitask engineering effort.
2. Two project scheduling methods:
   i. Program Evaluation and Review Technique (PERT)
   ii. Critical Path Method (CPM)
3. Both methods are driven by information developed in earlier project planning

activities:

i. Estimates of effort
ii. A decomposition of product function
iii. The selection of the appropriate process model
iv. The selection of project type and task set

4. Both methods allow a planer to do:
   i. Determine the critical path
   ii. Time estimation
   iii. Calculate boundary times for each task
5. Boundary times:
   i. The earliest time and latest time to begin a task
   ii. The earliest time and latest time to complete a task
   iii. The total float.

VII. Risk Management

1. What is Software Risk?

(Question: Explain the concept of risks in software- 4 marks)

i. Process of restating the risks as a set of more detailed risks that will be easier to mitigate, monitor, and manage.
ii. CTC (condition-transition-consequence) format may be a good representation for the detailed risks (e.g. given that <condition> then there is a concern that (possibly) <consequence>).
iii. This general condition can be refined in the following manner:
   a. Sub condition 1. Certain reusable components were developed by a third party with no knowledge of internal design standards.
   b. Sub condition 2. The design standard for component interfaces has not been solidified and may not conform to certain existing reusable components.
   c. Sub condition 3. Certain reusable components have been implemented in a language that is not supported on the target environment.
iv. The consequences associated with these refined sub conditions remains the same (i.e., 30 percent of software components must be customer engineered), but the refinement helps to isolate the underlying risks and might lead to easier analysis and response.

2. Concept of Proactive and Reactive risk strategies

(Question: What are reactive and proactive risk strategies? 4 Marks, 2 Marks each)

1. Reactive risk strategies

i. Reactive risk strategies follows that the risks have to be tackled at the time of their occurrence.
ii. No precautions are to be taken as per this strategy.
iii. They are meant for risks with relatively smaller impact.

2. Proactive risk strategies

i. Proactive risk strategies follows that the risks have to be identified before start of the project.
ii. They have to be analyzed by assessing their probability of occurrence, their impact after occurrence, and steps to be followed for its precaution.
iii. They are meant for risks with relatively higher impact.

3. Types of Software Risks

1. Software Requirement Risk

i. Lack of analysis for change of requirements.
ii. Change extension of requirements.
iii. Lack of report for requirements.
iv. Poor definition of requirements.
v. Ambiguity of requirements.
vi. Change of requirements.
vii. Inadequate of requirements.
viii. Impossible requirements.
ix. Invalid requirements.

2. Software Cost Risks

i. Lack of good estimation in projects
ii. Unrealistic schedule
iii. The hardware does not work well
iv. Human errors
v. Lack of testing
vi. Lack of monitoring
vii. Complexity of architecture
viii. Large size of architecture
ix. Extension of requirements change
x. The tools does not work well
xi. Personnel change, Management change, technology change, and environment change
xii. Lack of reassessment of management cycle

3. **Software Scheduling Risks**
   i. Inadequate budget
   ii. Change of requirements and extension of requirements
   iii. Human errors
   iv. Inadequate knowledge about tools and techniques
   v. Long-term training for personnel
   vi. Lack of employment of manager experience
   vii. Lack of enough skill
   viii. Lack of good estimation in projects

4. **Software Quality Risks**
   i. Inadequate documentation
   ii. Lack of project standard
   iii. Lack of design documentation
   iv. Inadequate budget
   v. Human errors
   vi. Unrealistic schedule
   vii. Extension of requirements change
   viii. Poor definition of requirements
   ix. Lack of enough skill
   x. Lack of testing and good estimation in projects

VIII. Risk Assessment

(Question: What is the concept of risk Assessment? - 4 Marks)

1. Risk assessment is another important case that integrates risk management and risk analysis.
2. There are many risk assessment methodologies that focus on different types of risks. Risk assessment requires correct explanations of the target system and all security features.
3. It is important that a risk referent levels like performance, cost, support and schedule must be defined properly for risk assessment to be useful.

IX. Risk Analysis

(Question: What is the concept of risk analysis? - 4 Marks)

1. There are quite different types of risk analysis that can be used.

2. Risk analysis is used to identify the high risk elements of a project in software engineering.
3. It provides ways of detailing the impact of risk mitigation strategies.
4. Risk analysis has also been found to be most important in the software design phase to evaluate criticality of the system, where risks are analyzed and necessary counter measures are introduced.
5. The main purpose of risk analysis is to understand risks in better ways and to verify and correct attributes.
6. A successful risk analysis includes important elements like problem definition, problem formulation, data collection.

X. Risk control- Need, RMMM strategy

(Question: Explain RMMM and its need in software project management- 8 Marks)

An effective strategy for dealing with risk must consider three issues

1. Risk mitigation (i.e., avoidance)
2. Risk monitoring
3. Risk management and contingency planning

Need for RMMM

i. Meet with current staff to determine causes for turnover (e.g., poor working conditions, low pay, competitive job market)
ii. Mitigate those causes that are under our control before the project starts
iii. Once the project commences, assume turnover will occur and develop techniques to ensure continuity when people leave
iv. Organize project teams so that information about each development activity is widely dispersed
v. Define documentation standards and establish mechanisms to ensure that documents are developed in a timely manner
vi. Conduct peer reviews of all work (so that more than one person is "up to speed")
vii. Assign a backup staff member for every critical technologist.
viii. During risk monitoring, the project manager monitors factors that may provide an indication of whether a risk is becoming more or less likely
ix. Risk management and contingency planning assume that mitigation efforts have failed and that the risk has become a reality
x. RMMM steps incur additional project cost

xi. Large projects may have identified 30 – 40 risks
xii. Risk is not limited to the software project itself
xiii. Risks can occur after the software has been delivered to the user

XI. Software Configuration Management (SCM)

1. Need of SCM

(Question: Give the need of SCM in software engineering- 4 Marks)

i. Also called software configuration management (SCM)
ii. It is an umbrella activity that is applied throughout the software process
iii. It's goal is to maximize productivity by minimizing mistakes caused by confusion when coordinating software development
iv. SCM identifies, organizes, and controls modifications to the software being built by a software development team
v. SCM activities are formulated to identify change, control change, ensure that change is being properly implemented, and report changes to others who may have an interest

2. Benefits of SCM

(Question: Explain the benefits of SCM- 4 marks)

i. Identify all items that collectively define the software configuration
ii. Manage changes to one or more of these items
iii. Facilitate construction of different versions of an application
iv. Ensure the software quality is maintained as the configuration evolves over time
v. Provide information on changes that have occurred

3. SCM Repository-Functions and Features supported

(Question: Explain SCM Repository with Diagram- 4 Marks, 2 marks each)

Automated SCM Repository

i. It is a set of mechanisms and data structures that allow a software team to manage change in an effective manner
ii. It acts as the center for both accumulation and storage of software engineering information
iii. Software engineers use tools integrated with the repository to interact with it

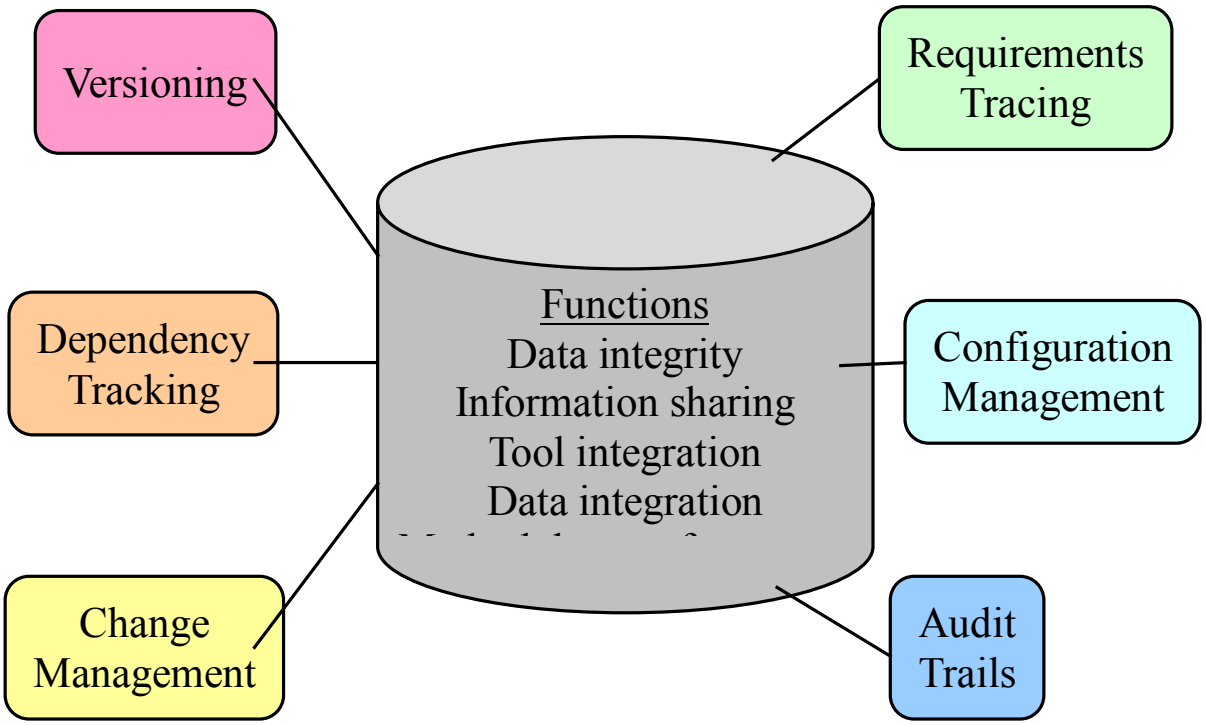

Figure 5: SCM Repository

Functions of SCM repository

(Question: List and explain the functions of SCM Repository- 6 Marks)

1. **Data integrity** :Validates entries, ensures consistency, cascades modifications
2. **Information sharing** :Shares information among developers and tools, manages and controls multi-user access
3. **Tool integration** : Establishes a data model that can be accessed by many software engineering tools, controls access to the data
4. **Data integration** :Allows various SCM tasks to be performed on one or more CSCIs
5. **Methodology enforcement** :Defines an entity-relationship model for the repository that implies a specific process model for software engineering
6. **Document standardization** :Defines objects in the repository to guarantee a standard approach for creation of software engineering documents

XII. SCM Process- Change control and version Control

(Question: What is version control and change control in SCM Process.- 8 Marks)

1. **Requirements Management**. Every requirement for the system should be tracked from end to end. This prevents development time being spent on non-essential features; it also allows QA to do a better job *of* testing. Should changes be introduced mid-development, their impact can be measured. Additionally, capturing requirements facilitates communicating them in a coordinated manner.
2. **Design.** The architecture and implementation blueprints of the system can be captured, clearly communicated, and easily distributed among developers, managers, and other decision-makers.
3. **Version Control**. As the software is developed, changes in the source are tracked. This is useful to developers, especially for development and bug fixing; it is useful for build automation, QA reporting, providing information to help desks, obtaining past releases, and managing patches.
4. **Build Tools**. The build process can be made fairly automated, easily constructing different platforms and releases as desired. Some tools provide the ability to track the interface versions that make up a release.
5. **Defect Tracking**. Allows problem reports to be directed to the development team, with reporting to management of problem areas and identifying trends. Developers need to be able to communicate problems among themselves.
6. **Automated Testing**. Provides assurance for regression testing and identifies problems during stress testing. Automation can rigorously beat on software and run through permutations faster than a human.
7. **Release Management**. Captures the contents of a particular release and information to construct a consistent build environment. This includes software patches for post-releases. Certain releases may have their own histories or pre-installation requirements.

8. **Distribution Management**. Captures how a release is distributed, whether by floppies, CD-ROM, WWW,FTP, etc. Certain distributions and platforms need their own install instructions.
9. **Installation Tools**. Some packages assist or construct the install process, or provide options for what gets installed, including licensing.
10. **Configuration Management**. Captures the hardware and software configuration at the installation site. Some customers may have different platforms, licenses, or patch releases applied.
11. **Help Desk**. Customers need a point of contact where they can go to report a problem, check the status of a fix, ask for a new feature, or to learn about the product.
12. **Impact Analysis**. When a new requirement is levied on the system, the impact from development, to QA, to documentation, to distribution, and so on, should be measured. Sometimes it isn't cost effective to do something just because you can.
13. **Process Management**. The product's lifecycle can be defined, communicated, sometimes enforced,[1] tracked, measured, and improved.
14. **Document Tracking**. Documentation consistency, changes, distribution, and store-housing are all managed under this branch.