Module 4

Software Project Management

Project

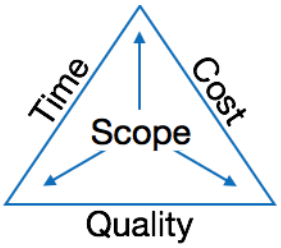
A project is a well defined task, which is a collection of several operations done in order to achieve a goal.

Software Project

A software project is the complete procedure of software development from requirement gathering to testing and maintenance, carries out according to the execution methodologies, in a specified period of time to achieve intended software product.

Software Project Management

Most software products are developed to satisfy the client's requirements. The underlying technology changes and advances so frequently and rapidly so that experience of one product may not be applied to the other one. These constraints bring risk in software development and hence it is essential to manage software projects efficiently.



The image above shows triple constraints for software projects. It is an essential part of software organization to deliver quality product, keeping the cost within client's budget and deliver the project as per scheduled. There are several factors, both internal and external, which may impact this triple constraint triangle. Any of three factor can severely impact the other two. Therefore, software project management is essential to incorporate user requirements along with budget and time constraints.

Software Project Management (SPM) is a critical aspect of overseeing the development of software applications. Here's an overview of its purpose, importance, complexities, the role of a project manager, their job responsibilities, necessary skills, and project planning activities:

Purpose:

The primary purpose of Software Project Management is to ensure the successful planning, execution, monitoring, and completion of software projects. It aims to deliver high-quality software on time and within budget while meeting stakeholder requirements.

Importance:

1. **Resource Optimization:** SPM helps in efficient allocation and utilization of resources, including time, budget, and personnel.
2. **Risk Management:** It identifies and mitigates risks to minimize the chances of project failure.
3. **Quality Assurance:** Ensures the development of a high-quality product that meets user expectations.
4. **Cost Control:** Manages the budget effectively to prevent cost overruns.
5. **Stakeholder Communication:** Facilitates communication among team members and stakeholders.
6. **Timely Delivery:** Ensures that the project is completed within the specified timeline.

Complexities:

SPM can be complex due to:

1. **Changing Requirements:** Requirements often evolve during the project, requiring adaptation.
2. **Technical Challenges:** Dealing with the intricacies of software development technologies.
3. **Team Dynamics:** Managing diverse teams with varying skills and personalities.
4. **Scope Creep:** Controlling changes to the project scope.
5. **External Factors:** External factors like market shifts can impact project goals.

Project Manager:

A Project Manager is responsible for overseeing the software project from initiation to closure. Their role includes planning, organizing, leading, and controlling the project.

Job Responsibilities:

1. **Scope Definition:** Clearly define project objectives, deliverables, and scope.
2. **Planning:** Create a project plan, schedule, and budget.
3. **Resource Allocation:** Assign tasks to team members and allocate resources.
4. **Risk Management:** Identify and mitigate risks that may impact the project.

5. **Quality Assurance:** Ensure that the software meets quality standards.
6. **Communication:** Foster effective communication among team members and stakeholders.
7. **Monitoring:** Track project progress and make adjustments as necessary.
8. **Issue Resolution:** Address and resolve issues that arise during the project.
9. **Documentation:** Maintain project documentation and reports.
10. **Closure:** Ensure a smooth project closure, including final deliverables and lessons learned.

Necessary Skills for Managing Projects:
1. **Leadership:** To guide and motivate the project team.
2. **Communication:** Strong communication skills to interact with stakeholders.
3. **Problem Solving:** The ability to address challenges and find solutions.
4. **Risk Management:** Identifying and mitigating risks effectively.
5. **Technical Knowledge:** Understanding software development processes.
6. **Time Management:** Efficiently managing project timelines.
7. **Budgeting:** Keeping the project within budget.
8. **Adaptability:** Adapting to changing project requirements.
9. **Negotiation:** Resolving conflicts and reaching agreements.
10. **Quality Management:** Ensuring the software meets quality standards.

Project Planning Activities:
1. **Requirements Analysis:** Understanding and documenting project requirements.
2. **Work Breakdown Structure (WBS):** Breaking the project into manageable tasks.
3. **Scheduling:** Creating a project timeline with task dependencies.
4. **Resource Allocation:** Assigning team members and resources to tasks.
5. **Risk Assessment:** Identifying potential risks and their impacts.
6. **Budgeting:** Estimating and allocating project costs.
7. **Communication Plan:** Defining how information will be shared among stakeholders.
8. **Quality Plan:** Outlining quality assurance and control processes.
9. **Change Management:** Establishing a process for handling scope changes.
10. **Monitoring and Control:** Continuously tracking progress and making adjustments.

Effective Software Project Management is essential for delivering successful software products that meet user needs and business goals while managing constraints and challenges effectively.

Project Estimation

For an effective management, accurate estimation of various measures is a must. With correct estimation managers can manage and control the project more efficiently and effectively. Project estimation may involve the following:

Software size estimation

Software size may be estimated either in terms of KLOC (Kilos Lines of Code) or by calculating number of function points in the software. Lines of code depend upon coding practices and Function points vary according to the user or software requirement. The main metrics used in software size estimation are

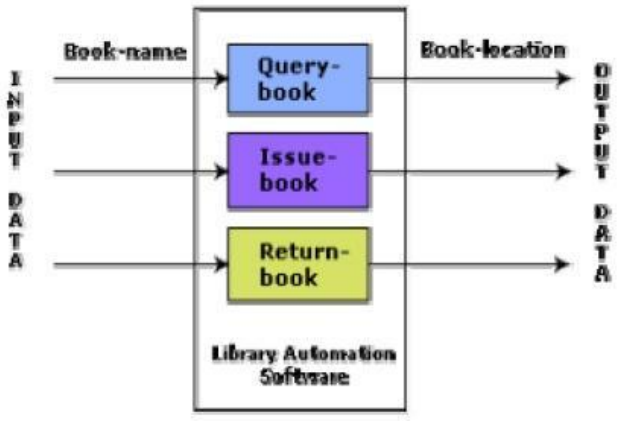
1. **Lines of Code (LOC)**

This metric measures the size of a project by counting the number of source instructions in the developed program. However, accurate estimation of the LOC count at the beginning of a project is very difficult. Project managers usually divide the problem into modules and sub modules and lines of code are estimated using a systematic guess from past experience. There are a lot of shortcomings for this method.

1. LOC gives a numerical value of problem size that can vary widely with individual coding style.
2. LOC is a measure of the coding activity alone even though coding is only a small part of the overall software development activities.
3. LOC measure correlates poorly with the quality and efficiency of the code. Larger code size does not imply better quality or higher efficiency.
4. LOC metric penalizes use of higher-level programming languages, code reuse etc.
5. LOC metric measures the lexical complexity of a program and does not address the more important issues of logical or structural complexities.
6. It is very difficult to accurately estimate LOC in the final product from the problem specification.

2. **Function Point Metric**

This method can be used to easily estimate the size of a software product directly from the problem specification. The conceptual idea behind the function point metric is that the size of a software product is directly dependent on the number of different functions or features it supports. For example, the different functions in a Library Automation Software is as shown



According to function point metric, size of a project depends on the following

1. Number of inputs
2. Number of outputs
3. Number of inquiries – number of distinct interactive queries
4. Number of files
5. Number of interfaces – interfaces used to exchange information with other systems

Function point is computed in three steps.

Step 1:

The first step is to compute the Unadjusted Function Point (UFP).

Step 2:

The computed UFP is refined in the next step. The complexity level of each of the parameters are graded as simple, average, or complex. The weights for the different parameters can be computed based on the table.

Parameter	Simple	Average	Complex
Inputs	3	4	6
Outputs	4	5	7
Inquiries	3	4	6
Files	7	10	15
Interfaces	5	7	10

Example:

UFP = (Number of inputs) * 4 + (Number of outputs) * 5 + (Number of inquiries) * 4 + (Number of Files) * 10 + (Number of interfaces) * 10

Step 3:

A technical complexity factor (TCF) for the project is computed and the TCF is multiplied with UFP to yield FP. The method identified 14 parameters that can influence the development effort. Each of these 14 factors is assigned a value from 0 (no influence) to 6 (strong influence). The resulting numbers are summed, yielding the total degree of influence (DI). Then TCF is computed as (0.65+0.01*DI). Finally FP is computed as

FP = UFP * TCF

Feature Point Metric

Function point metric only takes the number of functions that the system supports without considering the difficulty level of developing the various functionalities. To overcome this problem, an extension of the function point metric called **feature point metric** has been proposed. This metric incorporates **algorithm complexity** as an extra parameter. This metric reflects the fact that the more is the complexity of a function, the greater is the effort required to develop it.

Effort estimation

The managers estimate efforts in terms of personnel requirement and man-hour required to produce the software. For effort estimation software size should be known. This can either be derived by managers' experience, organization's historical data or software size can be converted into efforts by using some standard formulae.

Time estimation

Once size and efforts are estimated, the time required to produce the software can be estimated. Efforts required is segregated into sub categories as per the requirement specifications and interdependency of various components of software. Software tasks are divided into smaller tasks, activities or events by Work Breakthrough Structure WBS. The tasks are scheduled on day-to-day basis or in calendar months. The sum of time required to complete all tasks in hours or days is the total time invested to complete the project.

Cost estimation

This might be considered as the most difficult of all because it depends on more elements than any of the previous ones. For estimating project cost, it is required to consider -

- Size of software
- Software quality
- Hardware
- Additional software or tools, licenses etc.

- Skilled personnel with task-specific skills
- Travel involved
- Communication
- Training and support

Project Estimation Techniques

The important project parameters such as size, effort, schedule and cost can be estimated using different techniques. These techniques are broadly classified into three categories:

1. Empirical Estimation Techniques – Based on making an educated guess of the project parameters using prior experience. Main techniques are
   - Expert judgment technique
   - Delphi cost estimation
2. Heuristic Estimation Techniques – Relationships among different project parameters are modeled using mathematical expressions. It includes
   - Single variable – COCOMO model
   - Multi-variable – intermediate COCOMO
3. Analytical Estimation Techniques – Derive the required results starting with certain basic assumptions. The main method in this category is
   - Halstead's software science

Empirical Estimation Techniques

1. Expert Judgement Technique

In this technique, an expert makes an educated guess of the problem size after analysing the problem thoroughly. The expert estimates the cost of the different components and combines them to arrive at the overall estimate. This technique is subject to human errors and individual bias which can be eliminated by using a group of experts rather than a single one.

2. Delphi Cost Estimation

This technique is carried out by a team comprising of a group of experts and a coordinator. In this approach, the coordinator provides each estimator with a copy of SRS and a form for recording his cost estimate. The coordinator prepares a summary of the responses of all the estimators. The prepared summary is distributed to the estimators. Based on this summary, the estimators re-estimate. This process is iterated several rounds and finally the coordinator prepares the final estimate.

Heuristic Estimation Techniques - COnstructive COst estimation Model (COCOMO)

COCOMO was proposed by Boehm. According to Boehm any software development project can be classified into any of the three categories based on the development complexity.

1. Organic – The project deals with developing a well-understood application program, the size of the development team is reasonably small, and the team members are experienced in developing similar types of projects.
2. Semidetached – The development team consists of a mixture of experienced and inexperienced staff. Team members may have limited experience on related systems but may be unfamiliar with some aspects of the system being developed.
3. Embedded – Software being developed is strongly coupled to complex hardware.

   Boehm provides different sets of expressions to predict the effort and development time

   Organic

   Effort = 2.4 (KLOC) $^{1.05}$ PM

   Tdev = 2.5 (Effort) $^{0.38}$ Months

   Semidetached

   Effort = 3.0 (KLOC) $^{1.12}$ PM

   Tdev = 2.5 (Effort) $^{0.35}$ Months

   Embedded

   Effort = 3.6 (KLOC) $^{1.20}$ PM

   Tdev = 2.5 (Effort) $^{0.32}$ Months

Analytic estimation techniques involve a more systematic and data-driven approach to estimating various parameters, such as time, cost, or resources, by breaking down a complex problem into smaller, more manageable components. These techniques rely on mathematical or analytical methods to arrive at estimates. Some common analytic estimation techniques include:

-
- 1. **PERT (Program Evaluation and Review Technique):** PERT is commonly used in project management to estimate the duration of tasks in a project. It incorporates three time estimates for each task: optimistic, pessimistic, and most likely. These estimates are then used to calculate an expected duration and standard deviation.
-
- 2. **Monte Carlo Simulation:** This technique involves running a large number of simulations to model the possible outcomes of a complex process. It's often used for risk analysis and can provide a distribution of possible values rather than a single point estimate.
-

- 3. **Regression Analysis:** Regression analysis is used when there is a relationship between a dependent variable and one or more independent variables. It can be used to estimate future values based on historical data and trends.
-
- 4. **Cost-Benefit Analysis:** This technique is used to estimate the financial implications of a decision or project by comparing the costs and benefits associated with it. It helps in determining whether the benefits outweigh the costs.
-
- 5. **Earned Value Management (EVM):** EVM is a project management technique that involves tracking and analyzing the progress of a project in terms of its planned cost, schedule, and scope. It provides estimates of cost and schedule performance based on actual progress.
-
- 6. **Queuing Theory:** Queuing theory is used to estimate waiting times and queue lengths in systems where entities wait in line for service. It's applied in fields like operations research and service optimization.
-
- 7. **Bayesian Estimation:** Bayesian methods use probability distributions to update estimates as new data becomes available. It's particularly useful in situations where prior knowledge or beliefs can inform the estimation process.
-
- 8. **Simulation Modeling:** Simulation models use computer programs to replicate real-world processes, allowing for the estimation of various outcomes under different scenarios. It's commonly used in fields such as operations research and engineering.
-
- 9. **Decision Trees:** Decision trees are used to model decisions and their potential outcomes in a structured way. They can help estimate the expected value of different courses of action.
-
- 10. **Queuing Models:** These models estimate wait times and service times in systems with queues, such as customer service centers or manufacturing lines.

Project scheduling

Project scheduling is a critical aspect of project management. Here are the steps involved in creating a project schedule and introducing the tool of Gantt Charts:

1. Define Project Scope: Clearly outline the project's objectives, deliverables, and constraints.

2. Identify Tasks: Break down the project into smaller, manageable tasks or activities.

3. Sequence Tasks: Determine the order in which tasks need to be executed. Identify dependencies between tasks.

4. Estimate Durations: Estimate the time required for each task based on historical data or expert judgment.

5. Assign Resources: Allocate the necessary resources (people, equipment, materials) to each task.

6. Create a Gantt Chart: Introduce the tool of Gantt Charts. A Gantt Chart is a visual representation of the project schedule, showing tasks as bars on a timeline.

7. Input Data: Enter task names, start and end dates, and dependencies into the Gantt Chart software or tool.

8. Customize: Tailor the Gantt Chart to your project's needs by adding milestones, critical paths, and resource assignments.

9. Review and Adjust: Continuously monitor the project's progress on the Gantt Chart and make adjustments as needed.

10. Communicate: Share the Gantt Chart with the project team and stakeholders to keep everyone informed about the project's timeline.

Gantt Charts are valuable tools for project scheduling as they provide a clear visual representation of the project plan, making it easier to track progress and manage resources effectively.

A Gantt chart is a popular project management tool that provides a visual representation of a project schedule. It was developed by Henry L. Gantt in the 1910s and is widely used in various industries for planning and tracking projects. Here are some key features of a Gantt chart:

1. **Timeline Visualization:** A Gantt chart displays a timeline horizontally, with tasks or activities represented as bars or blocks along the timeline.

2. **Task Duration:** Each task is represented by a bar, and the length of the bar indicates the duration of the task.

3. **Task Dependencies:** Gantt charts can show dependencies between tasks, indicating which tasks must be completed before others can begin.

4. **Milestones:** Significant project milestones are often marked on the Gantt chart to highlight key achievements.

5. **Resource Allocation:** You can assign resources (people, equipment, etc.) to specific tasks on the chart.

6. **Critical Path:** Gantt charts can highlight the critical path, which is the sequence of tasks that, if delayed, will delay the entire project.

7. **Progress Tracking:** As the project progresses, you can update the chart to reflect actual start and end dates, allowing for real-time tracking.

8. **Communication:** Gantt charts are useful for communicating the project schedule to team members, stakeholders, and clients.

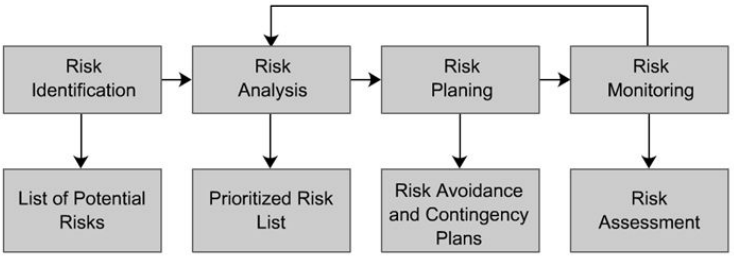
Gantt charts are commonly created using project management software or spreadsheet tools. They help project managers and teams plan, execute, and monitor projects more effectively by providing a visual roadmap of the project's timeline and tasks.

Risk Management

Risk is an expectation of loss, a potential problem that may or may not occur in the future. It is generally caused due to lack of information, control or time. A possibility of suffering from loss in software development process is called a software risk. Loss can be anything - increase in production cost, development of poor quality software, not being able to complete the project on time.

A software risk can be of two types (a) internal risks that are within the control of the project manager and (2) external risks that are beyond the control of project manager. Risk Management comprises of following processes:

1. Risk Identification
2. Risk Analysis
3. Risk Planning
4. Risk Monitoring



Risk Management Process

Risk Identification

Risk identification is a systematic attempt to specify threats to the project plan (estimates, schedule, resource loading, etc.). By identifying known and predictable risks, the project manager takes a first step toward avoiding them when possible and controlling them when necessary.

There are two distinct types of risks: generic risks and product-specific risks. *Generic risks* are a potential threat to every software project. *Product-specific risks* can be identified only by those with a clear understanding of the technology, the people, and the environment that is specific to the project at hand.

One method for identifying risks is to create a *risk item checklist.* The checklist can be used for risk identification and focuses on some subset of known and predictable risks in the following generic subcategories:

• *Product size*—risks associated with the overall size of the software to be built or modified.

• *Customer characteristics*—risks associated with the sophistication of the customer and the developer's ability to communicate with the customer in a timely manner.

• *Process definition*—risks associated with the degree to which the software process has been defined and is followed by the development organization.

• *Development environment*—risks associated with the availability and quality of the tools to be used to build the product.

• *Technology to be built*—risks associated with the complexity of the system to be built and the "newness" of the technology that is packaged by the system.

• *Staff size and experience*—risks associated with the overall technical and project experience of the software engineers who will do the work.

Risk Analysis

In this phase the risk is identified and then categorized. After the categorization of risk, the level, likelihood (percentage) and impact of the risk is analyzed. **Likelihood** is defined in percentage after examining what are the chances of risk to occur due to various technical conditions. These technical conditions can be:

1. Complexity of the technology
2. Technical knowledge possessed by the testing team
3. Conflicts within the team
4. Teams being distributed over a large geographical area
5. Usage of poor quality testing tools

With **impact** we mean the consequence of a risk in case it happens. It is important to know about the impact because it is necessary to know how a business can get affected:

1. What will be the loss to the customer?
2. How would the business suffer?
3. Loss of reputation or harm to society
4. Monetary losses
5. Legal actions against the company
6. Cancellation of business license

Level of risk is identified with the help of:

Qualitative Risk Analysis: Here you define risk as:

- High
- Low
- Medium

Quantitative Risk Analysis: It can be used for software risk analysis but is considered inappropriate because risk level is defined in % which does not give a very clear picture.

RISK MITIGATION, MONITORING, AND MANAGEMENT

All of the risk analysis activities have a single goal—to assist the project team in developing a strategy for dealing with risk. An effective strategy must consider three issues:

- Risk Avoidance
- Risk Monitoring
- Risk Management and Contingency Planning

If a software team adopts a proactive approach to risk, avoidance is always the best strategy. This is achieved by developing a plan for *risk mitigation.* Once the causes for risk are identified, it is possible to mitigate those causes that are under our control before the project starts. As the project proceeds, risk monitoring activities commence. The project manager monitors factors that may provide an indication of whether the risk is becoming more or less likely. In addition to monitoring these factors, the project manager should monitor the effectiveness of risk mitigation steps.

Risk management and contingency planning assumes that mitigation efforts have failed and that the risk has become a reality. Suppose a project is well underway and a number of people announce that they will be leaving. If the mitigation strategy has been followed such that staff backup is available, the project manager may temporarily refocus resources (and readjust the project schedule) to those functions that are fully staffed, enabling newcomers who must be added to the team to "get up to speed". Those individuals who are leaving are asked to stop all work and spend their last weeks in "knowledge transfer mode". This might include video-based knowledge capture, the development of "commentary documents", and/or meeting with other team members who will remain on the project.

The RMMM Plan

A risk management strategy can be included in the software project plan or the risk management steps can be organized into a separate *Risk Mitigation, Monitoring and Management Plan.* The RMMM plan documents all work performed as part of risk analysis and is used by the project manager as part of the overall project plan. Some software teams do not develop a formal RMMM document. Rather, each risk is documented individually using a *risk information sheet* (RIS). In most cases, the RIS is maintained using a database system, so that creation and information entry, priority ordering, searches, and other analysis may be accomplished easily.

Once RMMM has been documented and the project has begun, risk mitigation and monitoring steps commence. Risk mitigation is a problem avoidance activity. Risk monitoring is a project tracking activity with three primary objectives:

(1) To assess whether predicted risks do, in fact, occur;

(2) To ensure that risk aversion steps defined for the risk are being properly applied; and

(3) To collect information that can be used for future risk analysis.