FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
University of Lodz

Master's in computer science 2024/25 Winter Semester
Lecture 01
Date: 04/10/2024

Introduction to Software Project Management
Lecturer: Himanshi

1. Software Project

A **Software Project** refers to the entire process involved in creating software, starting from gathering requirements, through development, testing, and ongoing maintenance. It follows specific methodologies and is completed within a defined time frame to deliver a desired software product.

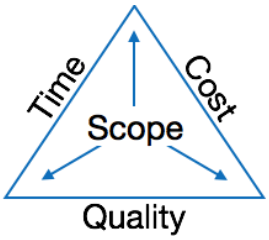
1.1. Why is Software Project Management Important?

Software is considered an **intangible product**, meaning it doesn't have a physical form like traditional products. Developing software is relatively new in the global business world, so there isn't much long-term experience in building software products compared to other industries.

Here are the key reasons why **software project management** is essential:

1. **Customization**: Most software products are custom-built to meet the specific needs of clients. This means each project can be unique, requiring careful management to meet the client's expectations.
2. **Rapid Technology Changes**: The technology used in software development evolves very quickly. A technique or tool used successfully in one project may become outdated or irrelevant for the next. This constant change increases the complexity and risks associated with software development.
3. **Risk Management**: Because of these factors—customization, rapid changes in technology, and the unique nature of software products—there are inherent risks in every software project. Efficient project management is crucial to identifying, controlling, and minimizing these risks to ensure successful project completion.

1.2. Understanding the Triple Constraints in Software Projects



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 constrain and deliver the project as per scheduled.

The **Triple Constraints** in software projects refer to three critical factors: **quality**, **cost**, and **time**. These constraints form a triangle that software organizations must carefully balance to deliver a high-quality product.

- **Quality**: Delivering a product that meets the client's needs and maintains high standards.
- **Cost**: Ensuring the project stays within the client's budget.
- **Time**: Completing the project on schedule.

1.2.1. The Impact of the Triple Constraints

Various internal and external factors can affect this triangle. For example, if one aspect is changed, it can have significant consequences on the other two:

- **Increasing the quality** may require more time and raise costs.
- **Reducing the budget** might affect the quality or extend the timeline.
- **Tightening the schedule** could lead to increased costs or reduced quality.

1.2.2. The Importance of Software Project Management

Managing these **triple constraints** effectively is essential in ensuring a successful software project. Software project management helps incorporate **user requirements** while balancing **budget** and **time constraints**. This balance ensures that the project remains on track and delivers the desired results to the client.

2. Software Project Manager

A **Software Project Manager** is the person responsible for overseeing and guiding a software project from start to finish. They don't necessarily write code or directly create the software themselves, but they ensure that all the processes and people involved are working smoothly. The project manager understands the entire **Software Development Life Cycle (SDLC)** and manages all the activities that take place during software production.

Their role involves **monitoring progress**, creating and executing various plans, arranging resources, and ensuring smooth communication among team members. The goal of a software project manager is to manage **costs, budgets, resources, time**, and ensure **quality** while making sure the **client is satisfied** with the end product.

2.1. Responsibilities of a Software Project Manager

The project manager has many responsibilities, which can be grouped into three main areas:

2.1.1. Managing People

- **Project Leader**: The manager acts as the leader of the project, guiding the team.
- **Communication with Stakeholders**: They maintain contact with clients and other key people involved in the project.

- Managing Human Resources: The manager ensures that the right people with the necessary skills are part of the team.
- Reporting Structure: They set up clear reporting channels so that everyone knows who to report to and how information flows.

2.1.2. Managing the Project

- Defining the Project Scope: They set clear boundaries on what the project will and won't include.
- Managing Project Activities: The manager oversees all project management tasks, such as planning and execution.
- Monitoring Progress: They track the development process and ensure the project stays on track.
- Risk Analysis: At every stage, they identify potential risks and take steps to avoid or resolve them.
- Project Spokesperson: The manager represents the project, communicating its status to stakeholders.

3. Software Management Activities

Software project management involves a wide range of activities, from planning and resource allocation to cost estimation and scheduling. Key activities include:

- Project Planning: Developing a plan before the project begins, outlining how the project will proceed.
- Scope Management: Defining exactly what the project will include and ensuring that it stays within those boundaries.
- Project Estimation: Estimating the size, effort, time, and cost required to complete the project.

3.1. Key Components of Software Project Management

3.1.1. Project Planning

Project planning happens before the actual work on the software starts. Though it doesn't directly produce the software, it is crucial for making sure everything runs smoothly. Planning involves creating a series of steps and processes that guide the project to completion.

3.1.2. Scope Management

Scope management defines the boundaries of the project—what will be done and what will not. This is important to avoid confusion, delays, and cost overruns. Key tasks include:

- Defining the Scope: Clearly outlining what the project will achieve.
- Verification and Control: Ensuring the scope is followed and making adjustments if necessary.
- Dividing the Project: Breaking the project into smaller, manageable parts.
- Scope Control: Managing any changes to the scope as the project progresses.

4. Project Estimation

Accurate estimation is critical for managing a project effectively. It helps in planning and controlling the project by estimating the size of the software, the effort required, the time it will take, and the cost involved.

a. Software Size Estimation

The size of the software is often measured by:

- **KLOC (Kilo Line of Code)**: The number of lines of code required.
- **Function Points**: Units that measure the functionality provided to the user.

b. Effort Estimation

Effort estimation calculates the number of people and hours needed to complete the project. This can be based on:

- The **size of the software**.
- The manager's **experience**.
- The organization's **past data**.

c. Time Estimation

Once the size and effort are estimated, the time required can be calculated. This involves breaking the project down into smaller tasks using a **Work Breakdown Structure (WBS)**. Each task is then scheduled, and the total time required is the sum of all tasks.

d. Cost Estimation

Estimating the cost of a project is one of the most challenging tasks, as it depends on various factors such as:

- The size of the software.
- The quality required.
- The hardware and additional tools needed.
- Personnel costs.
- Travel, communication, and training expenses.

5. Project Estimation Techniques

In software project management, **project estimation** involves calculating several key factors, such as the **size**, **effort**, **time**, and **cost** required to complete a project. Project managers use two widely recognized techniques to estimate these factors:

5.1. Decomposition Technique

This method views the software as a product made up of various components or compositions. The two main models under this technique are:

- **Line of Code (LOC) Estimation**: This model estimates the size of the software based on the total number of lines of code (LOC) it will contain. The larger the codebase, the more complex and time-consuming the project is likely to be.
- **Function Points (FP) Estimation**: This method estimates the size of the software by counting the number of **function points**, which represent the software's functional features from the user's perspective. This approach focuses on functionality rather than just code.

5.2. Empirical Estimation Technique

This technique uses formulae derived from previous project experiences to estimate key parameters like effort, cost, and time. The two common models under this technique are:

- **Putnam Model**: Developed by Lawrence H. Putnam, this model is based on a Rayleigh curve (Norden's frequency distribution). It estimates the relationship between the size of the software, time, and the effort required for its development.
- **COCOMO Model (COnstructive COst MOdel)**: Created by Barry W. Boehm, COCOMO divides software projects into three categories:
    - **Organic**: Projects that are relatively small, simple, and developed by small teams.
    - **Semi-detached**: Projects that are moderately complex, requiring medium-sized teams with mixed levels of experience.
    - **Embedded**: Projects that are highly complex, often involving hardware and software integration, and developed by large teams.

5.3. Project Scheduling

Project Scheduling refers to creating a detailed roadmap that outlines all the activities required to complete the project. Each activity is scheduled in a specific order, with a defined time slot for its completion.

Key steps in project scheduling include:

1. **Breaking Down Tasks**: The project is divided into smaller, manageable tasks or activities.
2. **Identifying Dependencies**: Each task is correlated with others to identify dependencies, meaning which tasks need to be completed before others can start.
3. **Estimating Time Frames**: The project manager estimates how long each task will take to complete.
4. **Dividing Time into Work Units**: Time is allocated to tasks in **work-units** (e.g., hours, days).
5. **Assigning Resources**: Adequate work-units (time and personnel) are assigned to each task to ensure it can be completed as planned.
6. **Critical Path Identification**: The manager identifies tasks that lie on the **critical path**, which are tasks that must be completed on time because of their dependencies. Any delay in critical path tasks can delay the entire project.
7. **Managing Non-Critical Tasks**: Tasks that are not on the critical path have more flexibility and are less likely to impact the overall schedule.

6. Resource Management in Software Projects

In software development, **resources** are everything used to create the final product. This includes **human resources**, such as developers and project managers, as well as **tools**, **software libraries**, and other assets necessary for development.

Resources are often limited and are managed as a pool within the organization. If resources are not available when needed, the project can be delayed, causing it to fall behind schedule. On the other hand, adding extra resources can increase the project's overall cost. Therefore, it's important to properly estimate and allocate the right amount of resources to complete the project efficiently.

Key activities involved in **resource management** include:

- **Organizing the Project Team**: Create a project team and assign specific responsibilities to each team member.
- **Determining Resource Requirements**: Identify the resources needed at different stages of the project and check their availability.
- **Managing Resources**: Request resources when needed and deallocate them once they are no longer required to avoid wasting them.

6.1. Project Risk Management

Risk management is the process of identifying, analyzing, and preparing for both expected and unexpected risks that may arise during a project. Risks can come from various sources, and if not managed properly, they can derail the project.
Common risks include:

- **Staff Turnover**: Key team members leaving and new members joining the project.
- **Organizational Changes**: Shifts in management or leadership.
- **Requirement Issues**: Misinterpreting or changing project requirements.
- **Underestimation**: Failing to accurately estimate the time and resources needed.
- **Technological or Environmental Changes**: New technologies emerging or changes in the business environment, including competition.

6.2. Risk Management Process

The process of managing risk involves several steps:

1. **Identification**: List all possible risks that could impact the project, both predictable and unpredictable.
2. **Categorization**: Group the identified risks based on their potential impact into categories such as high, medium, and low intensity.
3. **Management**: Analyze the likelihood of each risk occurring at different stages of the project. Develop a plan to either avoid or deal with these risks, aiming to minimize their negative effects.
4. **Monitoring**: Continuously monitor risks and their early warning signs. Track how effective your risk mitigation strategies are and make adjustments as needed.

By effectively managing resources and risks, a software project can stay on track and be completed within time and budget constraints, even in the face of challenges.

7. Project Execution and Monitoring

The **Execution and Monitoring** phase is where the tasks outlined in the project plan are carried out according to their scheduled timelines. This phase requires continuous monitoring to ensure that everything is progressing as expected and to identify any potential risks.
Key activities involved in **execution and monitoring** include:

- **Activity Monitoring**: All scheduled activities are observed on a day-to-day basis. Each task is considered complete only when all associated activities are finished.
- **Status Reports**: These reports provide an update on the progress of activities and tasks, typically on a weekly basis. Tasks are marked as finished, pending, or work-in-progress based on their status.
- **Milestones Checklist**: The project is broken down into phases, with key tasks or milestones to be achieved at each stage. This checklist is reviewed periodically (e.g., every few weeks) to report the completion status of these milestones.

7.1. Project Communication Management

Effective communication is critical to the success of any project. It ensures that there is clear understanding between the client and the organization, among team members, and with other stakeholders like hardware suppliers.

The **communication management** process includes the following steps:

1. **Planning**: This step involves identifying all the project stakeholders and determining the best modes of communication for each group. It may also include arranging for additional communication tools if required.
2. **Sharing**: Once the communication plan is in place, the project manager ensures that the right information is shared with the right person at the right time. This keeps everyone up to date on the project's progress and current status.
3. **Feedback**: The project manager uses feedback mechanisms to gather input from stakeholders. Status reports and performance evaluations are shared, and stakeholders provide their feedback, which helps in making necessary adjustments.
4. **Closure**: At the end of a major event, phase, or the entire project, a formal closure is announced. This is communicated to all stakeholders through emails, hard copy documents, or other effective methods of communication to ensure everyone is updated.

Effective communication and monitoring during the project execution phase help identify and resolve issues early, keeping the project on track and stakeholders informed.

8. Configuration Management in Software Projects

Configuration Management (CM) is the process of tracking and controlling changes in software during its lifecycle. These changes may include adjustments to requirements, design, functionality, or

development processes. The goal of CM is to ensure that all modifications are properly managed, documented, and verified to avoid confusion or unintended consequences.
According to IEEE, CM is defined as **"the process of identifying and defining the items in the system, controlling the change of these items throughout their life cycle, recording and reporting the status of items and change requests, and verifying the completeness and correctness of items."**
Key Elements of Configuration Management

- **Baseline**: In the software development life cycle (SDLC), a phase is considered "baselined" when all activities related to that phase are completed and documented. If the phase is not the final one, its output will be used as input for the next phase. Configuration management ensures that any changes made after the baseline are handled carefully to avoid disruption.
- **Change Control**: A critical function of configuration management, change control ensures that all modifications to the software system are made according to the organization's policies. This prevents unapproved changes from causing issues in the software.

8.1. Change Control Process

The change control process involves the following steps:

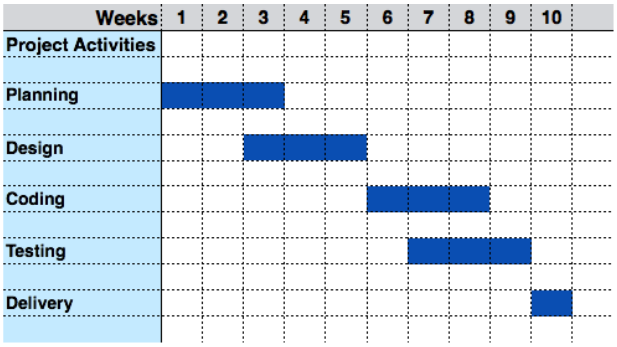
1. **Identification**: A change request is submitted, either from an internal or external source. Once identified, the request is formally documented.
2. **Validation**: The validity of the change request is checked to ensure it is legitimate and necessary.
3. **Analysis**: The potential impact of the change is analyzed in terms of time, cost, and effort. The overall effect on the software system is assessed to decide if the change is worth pursuing.
4. **Control**: If the change significantly impacts the system or multiple components, approval from higher management is required. This ensures that only essential changes are made.
5. **Execution**: Once approved, the change is implemented and carefully monitored.
6. **Closure**: After the change has been successfully incorporated, it is verified, documented, and the change request is formally closed.

9. Project Management Tools

Project management involves using tools to handle risk, uncertainty, and the overall complexity of software projects. Here are some key tools that assist in managing large-scale projects:

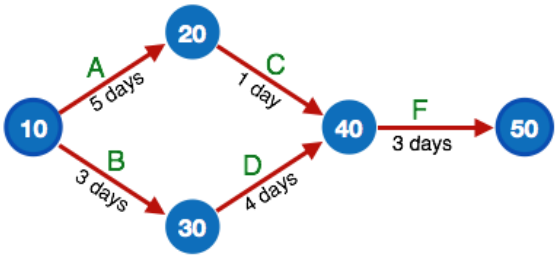
9.1. Gantt Chart

A **Gantt chart** is a visual representation of a project's schedule. Developed by Henry Gantt in 1917, it uses horizontal bars to show the duration of each task within the project timeline. Each bar represents an activity and its start and end dates, helping project managers monitor progress and ensure tasks are completed on time.



9.2. PERT Chart (Program Evaluation & Review Technique)

A **PERT chart** is a graphical tool that represents project tasks and their dependencies. It shows the sequence of tasks and identifies which tasks can be done in parallel and which must follow a specific order. PERT charts use nodes (events) and arrows (tasks) to depict how project tasks are interrelated.

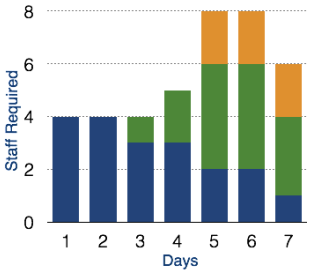


Events are shown as numbered nodes. They are connected by labeled arrows depicting sequence of tasks in the project.

9.3. Resource Histogram

A **Resource Histogram** is a bar chart that shows the number of resources (e.g., skilled staff) required over time for different phases of the project. It is useful for staff planning, helping to ensure that the right number of people are available when needed without over- or under-staffing.

Staff	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
Designer	4	4	3	3	2	2	1
Developer	0	0	1	2	4	4	3
Tester	0	0	0	0	2	2	2
Total	4	4	4	5	8	8	6



9.4. Critical Path Analysis

Critical Path Analysis is used to identify the sequence of tasks that must be completed on time for the project to stay on schedule. Like the PERT chart, it shows task dependencies, but it also highlights the **critical path**—the longest sequence of dependent tasks that determines the shortest time in which a project can be completed. Tasks on the critical path must be finished in the specified order, and any delay in these tasks will delay the entire project.

Class Activity: Understanding Triple Constraints in Software Projects

Objective:
Students will work in groups to discuss and analyze the **triple constraints** (quality, cost, and time) in software projects and visually represent the relationships between these constraints.

Instructions for the Groups:

Task:
As a group, you will explore the **triple constraints** involved in software projects: **quality**, **cost**, and **time**. Discuss how these constraints are interconnected and how changes to one can impact the others.

Questions to Discuss:

1. **How do changes in one constraint (quality, cost, or time) affect the other two?**
    - Consider different scenarios, such as extending the project timeline, increasing the budget, or improving the quality.
2. **Provide an example where changes in one of the triple constraints caused challenges in a project.**
    - Think of a real-life example or a hypothetical project where a shift in one constraint affected the project outcome.

Activity:

Create a **diagram** that visually represents the relationship between **quality, cost, and time**. Use arrows, symbols, or colors to show how changing one constraint impacts the other two. Your diagram should clearly explain the cause-effect relationships between these three factors.

Presentation:

After completing the task, each group will:

1. Present their **diagram** on the board.
2. Explain how changes in one constraint (quality, cost, or time) impact the other constraints.
3. Discuss their example and how it demonstrates the effect of the triple constraints on a project.