Student Name: Devansh Akruvala

Course: SOEN 6841 Software Project Management

Journal URL: https://github.com/devansh-akruvala/SOEN_6841_Software_Project_Management

Week 1: 18/01/2024 – 25/01/2024

Date: 22/01/2024

Key Concepts Learned:

Chapter 1

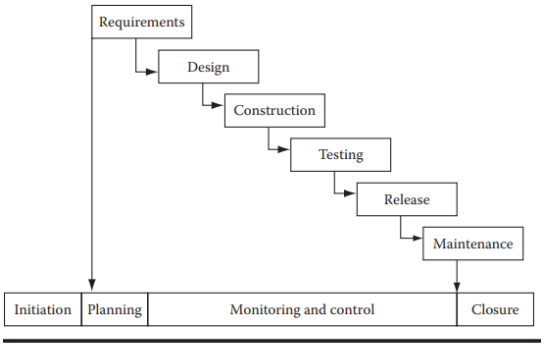
- **Project:** Project is a set of activities with definite start and end time aimed to achieve some predefined goals.
- Project consumes resources, budget and time.
- After the project is finished, the unconsumed resources and budget should be released.
- Project management processes may include project initiation, project planning, project monitoring and control, and finally project closure. ¡e software engineering processes may include requirement development, software design, software construction, software testing, and software maintenance. These software engineering processes have to be somehow accommodated in project management processes.
- A goal of any software project management is to develop/maintain a software product by applying good project management principles as well as software engineering principles so that the software project is delivered at minimum cost, within minimum time, and with good product quality.
- Organization level Process > Project Process > Lifecycle process
- When software is developed for use by the organization itself, it is known as a software application. When software is developed for the purpose of selling to customers and not for use by the organization itself then it is known as a software product.
- Requirements for a successful software project manager: Understand project management, understand software engineering, understand technology and tools, Manage team, customer and suppliers, Work under organization framework.
- Software Project task: Requirement management, Design Management, Source Code building, Software testing, software deployment and software maintenance.
- Project management processes form the basis on which a project can be initiated, planned, monitored, controlled, and closed. On the other hand, software engineering processes define structure, steps, and procedures to do various tasks in software development. But these processes lack the ability to schedule, plan, and control themselves. It is the project management processes that do the job of scheduling, planning, and controlling software engineering processes.

    1. Project Initialization:
        - First among project management processes is the project initiation process.
        - We can further divide the initiation process discussion into processes for application initiation, product initiation, and product implementation initiation.
        - **Software project initiation tasks:** Initial schedule estimates, Project charter, Project scope, Project objectives, Initial effort estimates, Initial effort estimates.

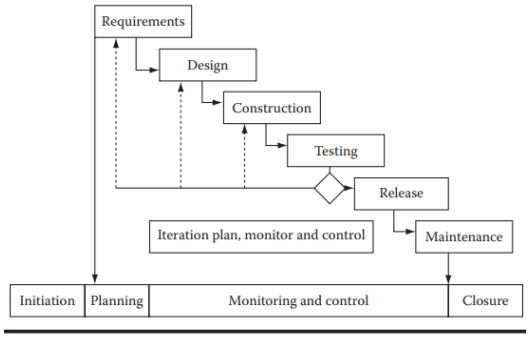
- Software applications are specially built based on a limited set of user requirements. So, they have limited features to fulfill the specific needs of end users. Software products on the other hand are built with a large number of features to take care of the needs of different kinds of users
- development of software products does not start with end-user requirements. ¡e software vendor sees a market opportunity of developing such a product. He develops the software product and sells it or provides services using this software product to customers.
- So, whereas a software application is created based on end-user requirements, a software product is made using market research data.
- **Software product initiation tasks:** Market analysis, Product development cost estimate, Product features, Marketing channels, Product delivery method, Product or service.
- **Software product implementation initiation tasks:** Customization effort, Initial schedule estimates, Project charter, Project scope, Project objectives, Initial effort estimates, Initial cost estimates, Migration from legacy system.

2. Software Project Planning:

- Depending on the characteristics of a project, detailed project planning is done either after project initiation or after completion of project requirements. Generally, detailed project planning can be done only after the project team has complete requirements for the project since the requirements together with project scope determine effort, cost, and quality required. If complete details about these things are not available, a baseline for the project cannot be made. In project planning the main tasks that are to be planned are software life-cycle processes



Project management in waterfall model environment.



Project management in iterative model environment.

3. Software Project Monitoring and Control:
   - Due to the inherently risky nature of software projects, constant monitoring and control is required to rectify any event that may jeopardize the project.
4. Software Project Closure:
   - During project closure, all project artifacts are analyzed and completed. Data from these artifacts are transferred to central project repository so that these data can be used for future projects.

- Management Metrics Characteristics: Relevant, Meaningful Calibration ability, Activity level, Practical.
- Seven Tools of Quality: Check sheets, Histograms, Pareto charts, Cause and effect diagrams, Scatter diagrams, Control charts, Graphs.

Chapter 2

- **Project charter:** The project charter is the place where a big picture of the effort, even beyond the project, is captured. It should include things like project goals, project objectives, major responsibilities allocation, etc.

  Example: The project will provide a cutting-edge software solution to our sales team to provide excellent customer service for our customers so that all customer issues can be solved within 24hours of lodging of a complaint.

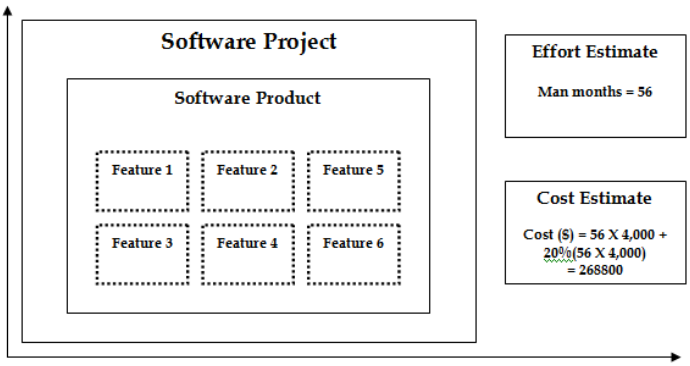
- **Project Scope:** The scope will include what functionalities are needed in the software product to be developed. It will also define level of quality needed in the software product.
- To deal with scope creep, it has to be ensured that the requirements are lucid and clear from the very start so that project effort estimation and project schedule are accurate.

  Example: The project will be delivered within 15 months from the date of start of the project. The software product that will be made through this project will have features for customer complaint logging, issue resolution, and issue closure. The software product should have the capability of supporting our customer base of 10,000, who will be using the service through an Internet connection by logging into our web portal.

- **Project Objectives:** The project should have a set of well-defined objectives that must be met. If any of these objectives are not met upon completion of the project, then the project will be considered to be a failure. The stakeholders state and set the project objectives.
- Objectives should be **SMART** (Specific, Measurable, Achievable, Relevant, Time Constrained).

  Example: The organization will be able to increase customer satisfaction to 99.5% from the existing level of 92%. This will help in reducing customer attrition, increasing repeat business from existing customers, and enhancing our brand value.

- **Estimate Initial Project Size:** At the project initiation stage, a rough project size should be estimated so that a sketch of the initial project plan can be realized. From the initial requirements a rough design estimate can be made which includes details about how the product can be broken down into parts. These parts can be sized from estimating, either the estimated number of lines of code required to build them or by using an estimated number of function points.
- **Estimate Initial Project Effort and Costs:** After preparing project charter and scope, an expert is hired who will make effort and cost estimate. Now bids are invited from software development companies for project planning & execution based on the effort & cost figured given by the expert.
- An initial budget is estimated for the project. As per the estimate a budget is sanctioned for the project. The budget includes costs to cover for salaries of people who will work on the software project, purchase of hardware, services, travel costs, management costs etc.
- The cost of the project is one of the most important considerations of stakeholders. If the project is going to cost more than they had anticipated and budgeted for, then most probably, the project will be called off.



Application in Real Projects:

How to initialize the project once we have Goals and objectives in mind and how to do Initial estimation of cost.

Peer Interactions:

- Discussed case studies given at end of chapter 1 and chapter 2.
- Discussed about the course project assigned with the group.

Challenges Faced:

Identify any challenges encountered while studying this week.
Note specific areas that need further clarification or additional effort.

Personal development activities:

- Read about Software Management tools like Jira and revision on SDLC.

Goals for the Next Week:

- Start Working on the project
- Read upcoming chapters
- Read Chapter 3 and 4

Student Name: Devansh Akruvala

Course: SOEN 6841 Software Project Management

Journal URL: https://github.com/devansh-akruvala/SOEN_6841_Software_Project_Management

Week 2: 25/01/2024 – 01/02/2024

Date: 29/01/2024

Key Concepts Learned:

- Estimation techniques can be divided in to
    - Experience Based techniques
        - Estimation by Analogy
        - Estimation by judgement
            - FPA
            - Delphi
    - Algo Based techniques
        - COCOMO Model

- **Estimation By analogy:** Estimate new projects by comparing them to similar past projects, preferably decomposing the estimate.
- **FPA:** FPA = UFP * VAF (CAF)
  For Calculating UFP there is a table where there are 5 function types and for each type is ranked low, avg, high.
  For calculating VAF there are 14 general system characteristics is rated on a scale from 0 to 5.
  VAF = (TDI*0.01) + 0.65
- **DELPHI:** The process involves individual estimation by team members followed by a comparison meeting to discuss differences. The individual estimates are averaged to arrive at a agreement estimate accepted by the group. Finally, a range of effort estimation values is calculated based this agreement.
- **COCOMO:**
    - Basic COCOMO:
      $Effort = 2.94 * EAF * (KLOC)^{E}$
      $Duration = 3.67 * (Effort)^{SE}$
    - Intermediate COCOMO:
      $E = a(KLOC)^{(E)}$

Software Project Type	*a*	*E*
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

- When to use which technique:

	Project details	Estimation technique
1	Historical project data + current project data	FPA
2	Current project data	COCOMO, Wide Band Delphi
3	No data	No technique can be used

Application in Real Projects:

The Estimation Project Study helps in the initial phase by providing a rough estimate of the resources required for the project.

Peer Interactions:

- Discussed Case study with peers
- Started working on projects and discussed many points in problem Identification and Market Analysis.

Challenges Faced:

- COCOMO and FPA Calculations require some clarification.

Personal development activities:

- Read some articles on Project Management and continue learning the Project Management Tools

Goals for the Next Week:

- Finish Project
- Read Chapter 4 and 5

Student Name: Devansh Akruvala

Course: SOEN 6841 Software Project Management

Journal URL: https://github.com/devansh-akruvala/SOEN_6841_Software_Project_Management

Week 3: 01/02/2024 – 09/02/2024

Date: 06/02/2024

Key Concepts Learned:

- **Why Risk Management?** A project should be completed within the given time period, but there could be some kinds of risks we may encounter while the project is ongoing. These risks can hamper project progress. Identifying the likelihood of risk occurrences and finding ways to tackle them is important.

- A project has these components: budget, time, resources, quality, and technology. If any risk occurs that might affect any of these components, then the project may fail.

- **Internal and External Risk:** Risks can be categorized as external and internal. If a risk to the project arises due to an aspect being dealt with by the project team, then it is an internal risk. All other risks are external risks.

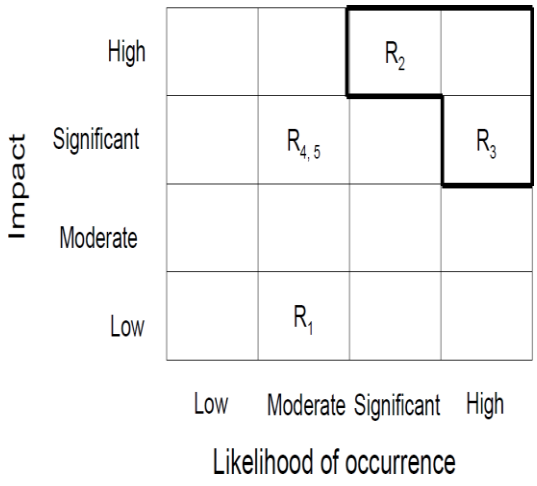
- **Types of Risk:**
    1. **Budget Risks:**
        - Risks impacting the project budget require continuous attention and control. If the budget exceeds the permissible limit, remedial action should be taken promptly.
        - Project steering committees may consider cutting product features, but this is not ideal.
        - Project expenses must be tracked and controlled at all times.
        - Market forces affecting salaries may be beyond the project manager's control, impacting project costs.
    2. **Time (Schedule) Risks:**
        - Project schedules should not be allowed to cross targeted release dates to avoid business opportunity loss.
        - Unforeseen circumstances, unexpected rework, and communication issues can lead to schedule slippage.
        - Meeting targeted deployment dates is crucial in today's fast-paced business environment.
    3. **Resource Risks:**
        - Project team members are costly resources, and creating reserved resources is challenging.
        - Software professionals' high demand poses a risk of team members leaving for better offers.
        - Keeping a pipeline open for potential replacements is suggested.
        - The risk of team members leaving mid-project can be mitigated by implementing a knowledge management system to store project knowledge and work.

4. **Quality Risks:**
    - High-quality software is essential for minimizing support costs during operations.
    - Poor software quality is a significant risk and can result from bad design or construction.
    - Defects may inadvertently occur due to complexity, large integration interfaces, or changes in requirements.
    - To address quality risks, it's recommended to integrate quality checks into the project schedule (quality planning).
    - Peer reviews, code reviews, and formal quality review processes should be consistently applied to all work products.
5. **Technology Risks:**
    - Technology obsolescence is a reality, with older products quickly becoming obsolete due to rapid market advancements.
    - Projects may face the challenge of using outdated technology, rendering the software product unusable before implementation.
    - The appropriate selection of programming language, hardware platform, and user access methods is crucial to prevent obsolescence.
    - When choosing technology tools and techniques, it's essential to contact vendors to ensure ongoing support for the tools.

- For **Risk Assessment** we can perform the following in beginning of project development as well as beginning of the iterations.
    1. **Risk Identification:** In this step we Identify risk related to overall project, product and business. The output of this step is called risk Items.
    2. **Risk Analysis:** In this step we assess the identified risk items on the basis of the likelihood of occurrence and the impact on project, product and business of each risk item.
        - **Qualitative Scale**: Low, Moderate, Significant, High.
        - **Quantitative:** probability of occurrence.

    And then

    Qualitative Scale: we plot it to filter out risk of high concerns as well as to take decisions.



Quantitative Scale: we calculate risk exposure i.e product of impact and probability, which can be used to rank risk items.

Risk exposure = risk probability $\times$ impact

3. **Risk prioritization:** Once risk items have been identified and analyzed we need to set priorities in order to determine where to focus risk mitigation efforts.

- **Risk Control:**
  - **Risk planning:** Can be performed at the beginning of the project development and reassessed at the beginning of the iterations.
  - **Resolution:** Can be performed throughout the project development.
  - **Risk monitoring:** Can be performed throughout the project development.

- **Risk response strategies**:
  - **Acceptance:** Acceptance means that the project has decided not to change the project plan to deal with a risk.
    1. Develop contingency plans
    2. Identify risk-trigger points
    3. Periodic review of risks and trigger points
    4. Use contingency allowance (e.g., time, budget, staff)

  - **Avoidance:** Avoidance is defined as changing the project plan to eliminate the risk of the condition to protect the project goals
    1. Do not use unfamiliar subcontractors
    2. Reduce scope to eliminate high-risk activities
    3. Add resources or time to critical tasks during planning
    4. Use familiar approaches rather than innovative ones

  - **Risk transfer:** Transference involves shifting the consequence of a risk to a third party, together with ownership of the risk response. Transferring a risk gives someone else the responsibility for its management. It does not eliminate the risk.
    1. Use of insurance and performance bonds
    2. Use of warranties and guarantees
    3. Use of contracts to transfer liability
    4. Use of a fixed-price contract with subcontractors

  - **Mitigation:** Mitigation reduces the probability and/or consequences of an adverse risk to an acceptable level. Taking early action to reduce a risk's probability and/or impact is more effective than reacting later.
    1. Adopt fewer complex processes
    2. Plan for additional testing of complex elements
    3. Use a more reliable or more stable vendor
    4. Use a prototype in the development process

- **Risk reduction leverage (RRL):** It is ratio of reduction in risk exposure over the cost of risk reduction.

$$Risk\ reduction\ leverage\ (RRL) = \frac{RE_{before} - RE_{after}}{Cost\ of\ risk\ reduction}$$

- **Risk in waterfall vs Iterative development model:** Using a waterfall model for project execution poses a significant risk because the outcome, the software product, is only ready after the entire project is completed, which often takes a prolonged period, say 6 months. This means that the result, whether positive or negative, is only known after investing time and resources for this duration. Waiting for such a long time to assess the outcome is indeed a major risk. To mitigate this risk, iterative approaches in software development have been adopted. Instead of tackling all requirements at once, they are broken down into manageable sets, with each set used to develop a small product. These small products, or software features, are developed within shorter time frames, typically 4-6 weeks or even less. After each iteration, there is a demonstrable product that can be tested against requirements, significantly reducing the overall risk by breaking it down into smaller, manageable parts.

Application in Real Projects:

- Real projects utilize risk management to identify critical components such as budget, time, resources, quality, and technology. By understanding potential risks to these components, project managers can proactively mitigate them to prevent project failure.
- Real projects conduct comprehensive risk assessments at the beginning of project development and iterations. This involves identifying and analyzing risks based on their likelihood of occurrence and impact on project, product, and business objectives.
- Project teams employ both qualitative and quantitative analysis techniques to assess risks. Qualitative scales help filter high-concern risks, while quantitative scales calculate risk exposure to rank risk items based on their probability and impact.
- After identifying and analyzing risks, real projects prioritize them to determine where to focus risk mitigation efforts. This ensures that resources are allocated efficiently to address the most critical risks first.
- Real projects implement various risk response strategies, including acceptance, avoidance, transfer, and mitigation. These strategies allow project teams to manage risks effectively and minimize their impact on project outcomes.

Thus, in real projects, risk management involves identifying critical components such as budget, time, resources, quality, and technology to proactively mitigate potential failures. This includes distinguishing between internal and external risks, categorizing them into types, conducting comprehensive assessments, employing qualitative and quantitative analysis techniques, implementing various response strategies, continuous monitoring, adapting approaches based on development models, and assessing risk reduction leverage to optimize mitigation strategies.

Peer Interactions:

- Started working on final draft of problem identification document of the project
- Discussed case study 3 and 4 with peers
- Reviewed examples of COCOMO model with peers.
- Worked on drafting market analysis

- Brainstorming ideas for features to include in project.

Challenges Faced:
- Risk Identification and Risk Analysis Needs Additional Effort
- Need to see some solved examples of RRL and need to review it because it is not provided in the textbook.

Personal development activities:
- Learning Jira
- Learning Trello
- Learning Asana
- Reading Blogs and case studies other than provided by the professor.

Goals for the Next Week:
- Submit first derivable of the project
- Read chapter 5 and 6
- Revise chapter 1-4
- Read Book and highlight important text.

Student Name: Devansh Akruvala

Course: SOEN 6841 Software Project Management

Journal URL: https://github.com/devansh-akruvala/SOEN_6841_Software_Project_Management

Week 4: 09/02/2024 – 17/02/2024

Date: 14/02/2024

Key Concepts Learned:

Chapter 5

- When users request changes, it's standard practice to adjust software features in software projects. Throughout the project, there are often numerous change requests, leading to multiple versions of the software product. These changes and versions are managed through configuration management processes.

- **What are the changes?**
  - Requirements changes
  - Changes in funding.
  - Technology advancements. Scaling system etc.
  - Solutions to problems.
  - Scheduling constraints.
  - Customer expectations.
  - Unexpected opportunities for an improved system

- **Good Configuration Management includes:**
  - **Version Control:** Tracking different versions of work is vital for project management, ensuring teams use the correct and latest documents and code.
  - **Centralized Configuration System:** A centralized setup fosters smooth collaboration, maintaining consistency across distributed teams.
  - **Continuous Integration:** Integrating new code frequently ensures software stability, aided by automated tools like smoke tests.
  - **Audit Facility:** A robust audit system tracks changes to documents and code, ensuring transparency and accessibility of previous versions.
  - **Secure Access:** Role-based control safeguards project data, allowing only authorized individuals to access documents and code.
  - **Easy Branching:** Streamlined branching mechanisms expedite the creation of new workspaces for different software versions.
  - **Verification and Accessibility:** Easy access to archived documents promotes transparency and accountability in project management.

- **Benefits of Change Management:**
  - Reduces confusion and establishes order.
  - Organizes the activities necessary to maintain product integrity.

- Ensures correct product configurations.
- Limits legal liability by providing a record of actions.
- Reduces life-cycle costs.
- Enables consistent conformance with requirements.
- Provides a stable working environment.
- Enhances compliance with standards.
- Enhances status accounting

- Purpose of Configuration Management is to establish and maintain the integrity of work products.
- To achieve the purpose we use
  - **configuration identification:** Define baseline components and establish identification schema**.**
  - **configuration control:** To provide a mechanism for preparing, evaluating, approving or disapproving all changes throughout the lifecycle.
  - **configuration status accounting:** To provide a mechanism for maintaining a record of the evolution of a system (e.g. history) at any time and report on the traceability of all changes to the baseline throughout the software lifecycle
  - **configuration audits:** Provide a mechanism for determining the degree to which the current state of the system mirrors the system pictured in baseline and requirements documentation and to provide the mechanism for establishing a baseline as well as ensure SCM process and procedures are performed combine this and write it as one small para.

- **Change Control Policy:**
  - All requirement changes must follow the process. If a change request is not documented according to this procedure, it will not be acted upon.
  - The change control board (CCB) designated for each project will decide which changes to implement.
  - The contents of the change database shall be visible to all project stakeholders.
  - The original text of a change request shall not be modified or deleted from the database.
  - Every incorporated requirement change shall be traceable back to an approved change request.
  - No design or implementation work will be performed on unapproved requirement changes other than feasibility exploration to assist with making the approval decision.

CHAPTER 6

- Project planning is a systematic process that defines project objectives, scope, resources, schedule, and deliverables, providing a roadmap for project execution and management.

- It involves setting goals, allocating resources, identifying risks, and establishing communication channels to ensure stakeholders are informed and engaged, facilitating efficient project management and goal achievement.

- Project scheduling can be done in 2 ways.

    - **Top-Down:** In top-down planning you first assign time duration for the entire project. Later you assign time duration for smaller tasks within the time periods of their respective container bigger tasks.
    - In a top-down approach to software project planning, product release dates are predetermined by software vendors or companies needing software systems to meet market demands. This method is essential for businesses operating under market pressures, ensuring timely delivery of software systems to prevent potential losses.

    - **Bottom-Up:** In bottom-up planning, time duration is assigned to small tasks first. Later, the time duration of all smaller tasks is added up to come up with time duration of their container larger task.

    - Bottom-up project planning is common for large software projects lacking clarity at the start. At the project's outset, teams gather information on tasks, project scope, requirements, and SLAs to formulate a development strategy.

- **Work Breakdown Structure:** It is a methodical approach to dissecting a project into smaller, manageable tasks. It establishes connections between these tasks, indicating their sequence and dependencies. This hierarchical breakdown enables clarity on task precedence, ensuring that certain tasks can't commence until others are completed.

- **Resource Allocation:** Software projects exhibit varying resource needs across phases; construction and testing demand extensive resources, while requirement and design phases require comparatively fewer. Additionally, in software projects, skills tend to be non-transferable; for instance, a software architect involved in design is typically not engaged in construction, allowing for efficient allocation to different projects upon completing their design role.

- **Supplier Management Plan:** Ensures quality and integration of components for outsourced projects, defining SLAs and compliance.

- **Configuration Management Plan:** With many scattered teams working on the same project in many cases, it is most important that configuration management is done carefully. It should be ensured that all teams have the same version of source code and document otherwise chances of rework will increase.

- **Communication Management:** Communication management depends solely on project organization structure, customer management strategy, and supplier management needs. For effective communication among all of these parties, it is essential that a proper communication management strategy is in place.

- **Defect Prevention Strategy (Quality Assurance):** Integral to project success, involves validation and verification of work products at each phase.

- **Project Duration:** Determined by the critical path, representing the longest sequence of tasks in the project.

- **Project Cost:** Estimated based on effort, productivity, resource costs, and overhead expenses.

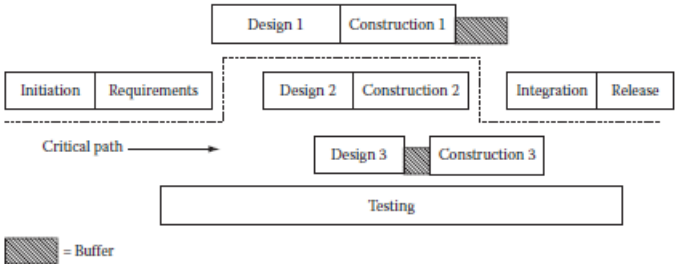
- **Tool Management:** Involves selecting programming languages, platforms, productivity tools, and project tracking systems. Jira.

- **Scope Management:** Defines project requirements and volume of work, ensuring alignment with project goals.

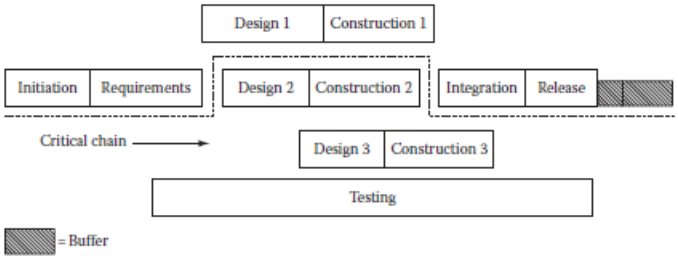
- **Effort Estimate:** Critical for planning resource allocation and project timelines, detailed in Chap 3.

- **Risk Management:** Crucial for identifying and mitigating potential risks throughout the project lifecycle.

- **Project Planning Techniques:**
    - **Critical Path Method (CPM):** All the tasks are first laid out on a sheet in an order based on their start dates. then the order in which tasks must be carried out is identified. Similarly tasks dependent on other tasks are identified and a relation is made between the tasks. Tasks with no relation among them are put in parallel. When all the tasks are thus laid out, a path is made, which runs along the longest path of execution.



    - **Goldratt's Critical Chain Method:** Introduced as an enhancement to CPM, this method addresses uncertainties and risks in project planning by focusing on protecting projects from failure. It involves padding uncertain tasks with buffers and closely monitoring buffer consumption to ensure project success.



- **Project Planning Artifacts:** In project planning, several essential artifacts are crafted, including project plans, risk management plans, effort and cost estimates, resource allocations, communication plans, and configuration management plans. These documents serve as comprehensive roadmaps, offering detailed instructions and structures for managing and overseeing project activities effectively.

Application in Real Projects:
Consider a project where a team is tasked with developing a fitness tracking app. When users request a new feature like meal tracking, change management becomes crucial. The process kicks off with evaluating the feasibility and impact of the addition, followed by stakeholder buy-in. Developers then integrate the feature, subjecting it to thorough testing for functionality and user experience. After successful testing, the feature is rolled out to users. Continuous monitoring and user feedback play a pivotal role in identifying and addressing any post-deployment issues, ensuring a seamless integration and effective utilization of the new feature.

In project planning, ensuring the app's timely release and alignment with user needs are top priorities. Techniques like Agile methodologies allow for iterative development, enabling quick adaptation to user preferences and market trends. Risk management strategies are essential for mitigating potential threats such as data security breaches or compatibility issues across different devices. Through meticulous planning and effective execution, the team can deliver a high-quality fitness tracking app that meets user expectations and enhances overall fitness journeys.

Peer Interactions:
- Completed working on final draft of problem identification document of the project
- Created Diagrams for project
- Discussed case study 5 and 6 with peers
- Worked on what is to be done in project pitch
- Created PPT for pitch.

Challenges Faced:
- Preparing for pitch according to rubrics took much effort
- Chapter 6 needs to be reviewed again

Personal development activities:
- Learning Jira
- Learning Trello
- Learning Asana
- Reading Blogs and case studies other than provided by the professor.

Goals for the Next Week:
- Read Chapter 5 and 6
- Prepare for mid term exam