Learning Journal

Student Name: Manpreet Singh Rana (40227463)

Course: Software Project Management (SOEN 6841)

Journal URL: https://github.com/msrana25/SOEN-6841-Journals

Week 3: Feb 11- Feb 17

Date: Feb 16, 2024

Key Concepts Learned:

New Terminologies

Configuration Management: - It is a set of processes and tools that help to manage change requests and different versions of software product.

Smoke test: - Whenever a new build occurs, a series of test cases, known as smoke tests, are executed to verify the proper functioning of critical system components. These tests aim to ensure that the system remains operational and that any new changes introduced reflect the latest functionality without breaking the existing system.

Project Planning: It encompasses continuous activity from initial concept through to system delivery, which is the most time-consuming project management activity.

Top-down Planning: - As the name suggests, top-down planning starts with allocation of time duration to the entire project and then later time duration us assigned to smaller tasks within time period of their respective bigger container classes.

Bottom-up Planning: - It is the reverse of top-down planning. Time duration is assigned first to smaller tasks which are sub part of bigger tasks.

Work Breakdown structure – It is systematic way of breaking down complete project work into smaller tasks. It also maintains relationships among tasks so that a clear task precedence is established to ensure that certain tasks do not start before the completion of other tasks.

Main concepts

Sources of change requests include change in requirements, changes in funding, technology advancements, unexpected opportunities for an improved system etc.

Four key functions of CM are **identification, change control, auditing,** and **status accounting.**

Configuration Identification deals with defining baseline components. The major activities in this phase include data identification requirements, identification of configuration items, documentation of configuration items, requirements, identification scheme and baselines, defining baselines and establishing identification schemas.

Configuration Control provides a mechanism for preparing, evaluating, approving or disapproving all changes throughout the life cycle.

The elements of a change control process include specifying who can initiate change requests and defining criteria for placing software components under formal change control. It involves conducting change impact analyses for each requested change, maintaining revision history, and implementing check-in/check-out procedures to manage versioning. The process typically involves the approval of changes by a Software Configuration Control Board (SCCB) and linking change requests to the trouble-reporting system for tracking and resolution. Reviews and regression tests are conducted to ensure changes do not cause unintended effects on the baseline system, and procedures are established for updating all affected software components in accordance with approved changes.

Configuration Status Accounting provides a mechanism for maintaining a record of evolution of a system at any time. It also keeps a record of traceability of all changes to the baseline throughout the software lifecycle.

Configuration Auditing offers a means to assess the extent to which the current state of the system aligns with the system depicted in the baseline and requirement documentation. It also facilitates the establishment of a baseline and ensure the execution of SCM processes and procedures.

Project Planning– Involves elaborate planning for all project components. The baseline structures are created in this phase, which help to execute, monitor and control the project. Project planning consists of various subcomponents like project scheduling, project budgeting, communication planning, quality planning etc.

Project scheduling can be done in 2 ways either as **top-down planning** or **bottom-up planning.**

Work Breakdown Structure – When it is ready, resources are allocated for each task. The allocation should be such that it matches closely matching required skills with those available. The number of resources required for a task depends on amount of effort required and how long it will be done.

The remaining portion of this chapter will be covered in next week's journal. This week I worked with my project team to prepare the content for the pitch of the project in line with the guideline rubrics mentioned shared by the professor.

Application in Real Projects:

Need of Configuration Management in Software Project Management

Managing software effectively is crucial because it's highly changeable and often invisible. Uncontrolled changes can lead to project chaos, delays, and quality issues. Different types of components, both executable (like code) and non-executable (such as documentation), need to be tracked throughout the software's lifecycle.

Customers might not have a clear idea of what they want, leading to scope creep and rework. This ambiguity can result in the "I'll Know It When I See It" syndrome, where requirements are uncertain until demonstrated. Without proper management, changes can be haphazard, leading to errors, missing features, or even the inability to locate the latest version of code.
Examples of poor practices include losing track of source code versions, fixing defects only to have them reappear, and features disappearing mysteriously. Without traceability between requirements, documentation, and code, confusion reigns, and mistakes are common.

Configuration management helps mitigate these issues by providing a structured approach to tracking changes, ensuring the right components are used, and maintaining consistency throughout the development process. It enables better communication, reduces errors, and ensures that the software delivered meets the customer's expectations.
Furthermore, risk management helps project stakeholders make informed decisions by providing insights into the potential impact and likelihood of risks occurring. It allows them to prioritize risks based on their severity and develop strategies to minimize their impact on project objectives.

Peer Interactions:

This week, our discussions revolved around Configuration Management (CM) in software project management. We dived into how CM helps manage change requests and software versions systematically. We explored concepts like smoke testing, project planning (both top-down and bottom-up), and work breakdown structure. Understanding the four key functions of CM – identification, change control, auditing, and status accounting – was pivotal. We delved into how these functions play a crucial role in maintaining project coherence and managing software effectively. Also discussed the real-world implications of CM, particularly in mitigating risks such as scope creep, rework, and unclear requirements.

Challenges Faced:

In depth understanding of four key functions of CM was bit challenging because there is a lot of data available to read and understand about each item which was little overwhelming. In preparation for the content of project pitches, getting the team on the same page regarding the content of pitch was challenging as well, as everyone wanted their ideas to be included for the presentation, but it was not feasible because the presentation time limit was only 4 minutes.

Personal development activities:

I am familiar with GitHub, so I explored Jira, since it also has limited Configuration Management capabilities including tracking changes to software components, manage versions and document configuration.

Goals for the Next Week:

Complete study of Project Planning phase of software project management. Preparation of midterm, which includes covering all chapters that are journalled till this week.