AKETCH MARY FRANCES

S20B23/215

SUMMARY

Chapter 3:SOFTWARE STRUCTURE PROCESS

The software process is defined as a framework of activities, tasks, and tasks required to create quality software. Each activity in the framework is populated with a set of software development activities. The way software development works are defined by a set of activities that identify the activities to be performed, the work products to be produced, the quality assurance points required, and the milestones to indicate progress.

The generic process model for software engineering includes a set of activities, activities, and tasks of the framework and framework. Framework activities include communication, planning, modeling, construction, and implementation with general activities such as project monitoring and control, risk management, quality assurance, configuration management, engineering journals, etc.

Process Flow Describes how the framework and the activities and tasks within each framework are organized sequentially and over time

1. The linear process flow executes each of the framework's five activities in sequence, beginning with communication and ending with implementation.
2. An iterative process flow repeats one or more steps before moving on to the next.
3. The evolutionary flow of processes carries out activities in a "circular" manner. Each circuit through five stages results in a more complete version of the software
4. The parallel process flow performs one or more activities in parallel with other activities.

If the project is much more complex, the communication activity can include six different activities: initiation, procurement, development, negotiation, specification, and validation. The

task set defines the actual work that must be done to meet software development performance goals. Small, relatively simple projects do not require as many tasks

detailed as complex sets of design tasks faced by the design team. Task sets are tailored to the specific needs of the software project and the characteristics of the project team

A process model describes a process problem encountered in software development work, identifies the environment in which the problem occurred, and proposes one or more best practice solutions to the problem.

Among the various approaches that have been proposed for evaluating and improving the software development process are the CMMI standard evaluation method for process improvement (phases: initialize, diagnose, define, operate, and learn), the CMM-based evaluation for internal process improvement (provides a diagnostic technique for Evaluation of the relative maturity level of the development organization)

SPICE (ISO/IEC15504) (specifies a set of requirements for evaluating the software development process), ISO 9001:2000 for software (a generic standard applicable to any organization that seeks to improve the overall quality of the product, system, or service provided).

Chapter 4:PROCESS MODELS

The process model offers a concrete road-map for work in software engineering. It defines the flow of all activities, actions, and tasks, the degree of iteration, the work results, and the organization of the work to be carried out.

Prescriptive process models aim to organize and structure software development. Activities and tasks follow sequential guidelines with specific progression guidelines. Each of these models proposes a slightly different process flow, but they all follow the same general framework activities: communication, planning, modeling, construction, and implementation.

The Waterfall Model: Proposes a systematic, sequential approach to software development, beginning with the specification of customer requirements and moving through planning, modeling, building, and implementation, and ending with the continuous support of the finished software

V-pattern: A variant of the waterfall pattern representation is called a V-pattern.As the development team moves down the left side of the V, the basic requirements of the problem are refined into increasingly detailed and technical representations of the problem and its solution. Once the code is generated, the team goes up the right side of the V and essentially performs a series of tests (quality assurance tasks) that validate each of the models created as the team goes down the left side.

Incremental process models are iterative and produce working versions of the software fairly quickly. By distributing the software in small but useful chunks, each chunk builds on previously provided chunks. The plan includes modifying the base product to better meet customer needs and providing additional features and functionality. This process repeats after each delivered increment until a complete product is made.

Evolutionary Process Models recognize the iterative and incremental nature of most software development projects and are designed to adapt to change. For example prototyping. Quick Design focuses on presenting the aspects of the software that are visible to end users. A quick design leads to the construction of a prototype. The prototype is implemented and evaluated by stakeholders who provide feedback that is used to further refine the requirements. Iteration adapts a prototype to the needs of different stakeholders while providing a better understanding of what needs to be done. spiral model.: This approach combines the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model. Using a spiral model, the software is developed in a series of successive versions. During the first iteration, a release can be a mock-up or a prototype. In further iterations, more and more complete versions of the designed system are created. Concurrent Models: Allows the software team to represent the iterative and concurrent elements of each process model specialized process models include component-based development, which emphasizes component reuse and assembly; the formal methods model, which promotes a mathematical approach to software development and verification; and an aspect-oriented model that includes cross-sectional considerations of the overall system architecture

Unified Process is a "use case driven, architecture-eccentric, iterative and incremental" software process designed as a framework for UML methods and tools. Phases of the Unified Process include the Inception phase, Elaboration phase, Construction phase, Transition phase, Production phase,

Personal and team models for the software process have been proposed. Both emphasize measurement, planning, and self-direction as key ingredients for a successful software process.

Chapter 5: AGILE DEVELOPMENT

Agile methods were developed to overcome the supposed and actual weaknesses of conventional software engineering. One of the most compelling characteristics of Agile approach is the ability to reduce change costs throughout the software development process.

Agile development emphasizes continuous communication and collaboration between developers and customers. It encompasses a philosophy that promotes customer satisfaction, incremental software delivery, small project teams (consisting of software engineers and stakeholders), informal methodologies, and minimal software development work products.

Extreme Programming is the most commonly used approach to agile software development.S. Extreme Programming uses an object-oriented approach as its preferred programming paradigm and encompasses a set of principles and practices that occur in the context of the framework's four activities: planning, design, coding, and testing.XP has many innovative and powerful techniques that enable an agile team to create frequent software releases that deliver features and functionality that stakeholders have described and then prioritized Industrial XP includes six new practices designed to help ensure Project XP success for large-scale projects in a large organization, such as B. Readiness assessment, project community, project charter, test-driven project management, retrospectives, and continuous learning. Other agile process models also emphasize the cooperation of people and the self-organization of teams but define their framework activities and choose other focal points. Scrum: Within each framework activity, work activities occur in a process model called a sprint.

The work performed in a sprint (the number of sprints required for each framework activity depends on the complexity and size of the product) is tailored to the problem at hand and is defined in real-time by the Scrum Team and changed frequently. Scrum emphasizes using a proven set of software process models for projects with tight deadlines, changing requirements, and business-critical problems. Each of these process templates defines a set of development activities (backlog, sprint, scrum meeting, demo) and allows the scrum team to create a process customized to the needs of the project.

Dynamic Systems Development Method: This approach provides a framework for building and maintaining systems that meet tight time constraints by using incremental prototyping in a controlled development environment. Each increment provides only enough functionality to move to the next increment. DSDM then defines three different iteration cycles: functional model iteration, design and build iteration, and implementation Agile Modeling (AM) is a practice-oriented methodology for the effective modeling and documentation of software-based systems. Modeling principles include modeling for a purpose, using multiple models, transparent movement (keep only models with long-term value), content versus representation, knowledge of models, and the tools used to create and customize them. Agile modeling suggests that modeling is essential for all systems, but the complexity, type, and size of the model should be scaled with the software being built. Agile Unified Process: handles the classic UP activities: launch, development, build, and transition.

AUP provides a serial overlay (i.e., a linear sequence of software engineering activities) that enables a team to visualize the overall process flow for a software project. However, within each of the activities, the team iterates to achieve agility and deliver meaningful software increments to end users as rapidly as possible. Each AUP iteration addresses the following activities: Modeling, Implementation, Testing, Deployment, Configuration and project management, and Environment management.

Chapter 6: HUMAN ASPECTS OF SOFTWARE ENGINEERING

A software engineer must master technical problems, learn and apply the skills needed to understand the problem, design an effective solution, build the software, and test it to the highest quality possible.

The traits of a software engineer include: Individual responsibility for commitments Keen sense (ability to adapt behavior to both environment and people) Brutal honesty Resilience to pressure Strong sense of justice (willingly shares credit with others) Demonstrates attention to detail, is pragmatic (recognizes that software engineering is not a religion that follows dogmatic principles, but a discipline that can be adapted to circumstances in there)Software Engineering Psychology covers individual cognition and motivation, the group dynamics of a software development team, and the organizational behavior of a company.

To improve communication and collaboration, development team members can take on a cross-border role for a development team to be effective, it must have purpose, commitment, trust, and a drive to improve. In addition, the team must avoid "toxicity" characterized by a fast-paced and frustrating work atmosphere, an inappropriate software development process, high levels of frustration-induced friction among team members, unclear roles in the development team, and a constant risk of bugs.

Factors considered when planning the structure of software development teams include the difficulty of the problem to be solved, the "size" of the resulting program(s) in lines of code or function points, how long the team will be together (duration from of the team), the degree of modularity of the problem, the required quality and reliability of the system under construction, the rigidity of the delivery date and the degree of sociability (communication) required by the project.

Organizational paradigms for software development teams include closed paradigm structures (the team follows the traditional hierarchy of power), random paradigm structures (prefers a loose structure based on individual initiative), as well as an open paradigm and a synchronous paradigm Agile team follow the agile philosophy and generally have more autonomy than more conventional development teams.

Agile teams value communication, simplicity, feedback, courage, and respect. Social Media is now an integral part of many software projects. Blogs, micro blogs, forums, and social networking opportunities help create a software engineering community that communicates and coordinates more effectively.

Cloud computing can affect how software engineers organize their teams, how they do their work, how they communicate and connect, and how they manage software projects. Where the cloud can improve the social and collaborative aspects of software development, its benefits far outweigh the risks.

Collaborative Development Environments include several services that improve communication and collaboration within a development team. These environments are particularly useful for global software development, where geographical separation can accelerate barriers to software development success.