Name: Anei Agany Them

Reg No:IS19B00/023

Course unit: software construction (BSCS3)

Summary:

Chapter 3: Software Processes Key Points

Software processes are the activities involved in producing a software system. Software process models are abstract representations of these processes.

General process models describe the organization of software processes. Examples of these general models include the waterfall model, incremental development, and reusable component configuration and integration. -Requirements engineering is the process of developing a software specification. Specifications are intended to communicate the system needs of the customer to the system developers.

Design and implementation processes are concerned with transforming a requirements specification into an executable software system. -Software validation is the process of checking that the system conforms to its specification and that it meets the real needs of the users of the system. -Software evolution takes place when you change existing software systems to meet new requirements. Changes are continuous, and the software must evolve to remain useful. Processes should include activities to cope with change. This may involve a prototyping phase that helps avoid poor decisions on requirements and design. Processes may be structured for iterative development and delivery so that changes may be made without disrupting the system as a whole. -Process improvement is the process of improving existing software processes to improving existing software processes to improve software quality, lower development costs, or reduce development time.

The process framework is required for representing common process activities. Five framework activities are described in a process framework for software engineering. Communication, planning, modeling, construction, and deployment are all examples of framework activities. Each engineering action defined by a framework activity

comprises a list of needed work outputs, project milestones, and software quality assurance (SQA) points.

Communication: By communication, customer requirement gathering is done. Communication with consumers and stakeholders to determine the system's objectives and the software's requirements.

Planning: Establish engineering work plan, describes technical risk, lists resources requirements, work produced and defines work schedule.

Modeling: Architectural models and design to better understand the problem and for work towards the best solution. The software model is prepared by:

- o Analysis of requirements
- o Design

Construction: Creating code, testing the system, fixing bugs, and confirming that all criteria are met. The software design is mapped into a code by:

- Code generation
- Testing

Deployment: In this activity, a complete or non-complete product or software is represented to the customers to evaluate and give feedback. On the basis of their feedback, we modify the product for the supply of better products.

Chapter 4: prescriptive process models Key Points

Prescriptive process models advocate an orderly approach to software engineering prescriptive" because they prescribe a set of process elements—framework activities, software engineering actions, tasks, work products, quality assurance, and change control mechanisms for each project.

Each process model also prescribes a process flow (also called a work flow)—that is, the manner in which the process elements are interrelated to one another.

All software process models can accommodate the generic framework activities but each applies a different emphasis to these activities with different process flow.

The waterfall model, sometimes called the classic life cycle, suggests a systematic, sequential approach to software development. Real projects rarely follow the sequential flow that the model proposes. The V-model provides a way of visualizing how verification and validation actions are applied to earlier engineering work.

Incremental development is useful when staffing is unavailable for a complete implementation. Rapid Application Development (RAD) is a "high-speed" adaptation of the linear sequential model. Evolutionary models are characterized in a manner that enables you to develop increasingly more complete versions of the software.

The spiral model is an evolutionary software process that couples the iterative nature of prototyping with the controlled and systematic aspects of waterfall development.

The spiral model is a realistic approach to the development of large-scale systems and software. In each step of the development process, the developer and customer better understand and react to risks at each evolutionary level. Each activity represents the state of software engineering activity; similarly, other activities, actions, or tasks can be represented in analogous manner.

chapter 5: Agile Software Development Key Points

Agile development is the ability to develop software quickly, in the face of rapidly changing requirements. In order to achieve this agility, we need to employ practices that provide the necessary discipline and feedback. We need to employ design principles that keep our software flexible and maintain ably, and we need to know the design patterns that have shown to balance those principles for specific problem.

Agile methods are iterative development methods that focus on reducing process overheads and documentation and on incremental software delivery. They involve customer representatives directly in the development process.
The decision on whether to us an agile or plan-driven approach to development should depend on the type of software being developed, the capabilities of the development

team, and the culture of the company developing the system. In practice, a mix of agile
and plan-based techniques may be used. Agile development practices include requirements expressed as user stories, pair
programming, refactoring, continuous integration, and test-first development.
-Scrum is an agile method that provides a framework for organizing agile projects. It is
centered around a set of sprints, which are fixed time periods when a system increment
is developed. Planning it based on prioritizing a backlog of work and selecting the highest
priority tasks for a sprint.
To scale agile methods, some plan-based practices have to be integrated with agile practice. These include up-front requirements, multiple customer representatives, more
documentation, common tooling across project teams, and alignment of releases across
teams.

Agile software development methods support a broad range of software development life cycle, with advantages such as frequent delivery, face-to-face communication with clients, efficient design and fulfils the business requirement, anytime changes are acceptable, and reduced total development time. However, there are disadvantages such as lack of formal documents, confusion, and maintenance of the finished project. Traditional models are not flexible for modification in project development if the requirement changes, and success is measured by confirmation to the plan.

chapter 6: Human aspects of software engineering Key Points

The Psychology of software engineering suggests a set of "boundaries spanning roles" that allow members of a software team to effectively move across team boundaries.

Traits of successful software engineers include a sense of individual responsibility, acutely aware of the needs of team members and stakeholder, Brutally honest about design flaws and offers constructive criticism, Resilient under pressure, Heightened sense of fairness, Attention to detail, Pragmatic, and a number of project factors such as difficulty of the problem, size of the program, modularization, quality and reliability of the system, rigidity of the delivery date, and degree of sociability. XP Team Values include communication, Simplicity, Feedback, Courage, Respect, and Impact of social media: blogs, microblogs, targeted on-line forums, social networking sites, and social book marking. Engineering using the Cloud provides access to all software engineering work, removes device dependencies, and provides avenues for distributing and testing software. Concerns include potential for interoperability problems and security risks. Collaboration tools include namespace, calendars, templates, metrics, communication analysis, and Artifact clustering. Global software teams face additional challenges associated with collaboration, coordination, and coordination difficulties. Agile teams are self-organizing and have many structures. Planning is limited by business requirements and organizational standards.