****

NAME : NANDAWULA MARIA

REG NO : S20B23/207

ACCESS NO : A90651

COURSE : BSCS

COURSE UNIT : SOFTWARE CONSTRUCTION

LECTURER : MR. SIMON FRED LUBAMBO

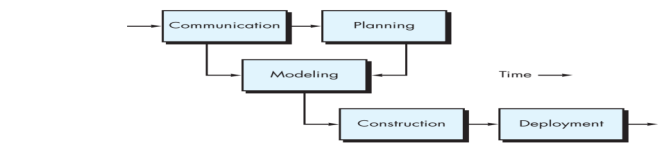
**Chapter 3 :SOFTWARE STRUCTURE PROCESS**

**Software process** is defined as a framework for the activities, actions, and tasks that are required to build high-quality software. Each framework activity is populated by a set of software engineering actions. Each software engineering action is defined by a task set that identifies the work tasks that are to be completed, the work products that will be produced, the quality assurance points that will be required, and the milestones that will be used to indicate progress.

**A generic process model** for software engineering encompasses a set of framework and umbrella activities, actions, and work tasks .The **framework activities** include communication, Planning, Modeling, construction, deployment. The **umbrella activities** include project tracking and control, risk management, quality assurance, configuration management, technical reviews and others

**Process flow:** Describes how the framework activities and the actions and tasks that occur within each framework activity are organized with respect to sequence and time

1. A linear process flow executes each of the five framework activities in sequence, beginning with communication and culminating with deployment.
2. An iterative process flow repeats one or more of the activities before proceeding to the next.
3. An evolutionary process flow executes the activities in a “circular” manner. Each circuit through the five activities leads to a more complete version of the software
4. A parallel process flow executes one or more activities in parallel with other activities.



If the project is considerably more complex with the communication activity might have six distinct actions inception, elicitation, elaboration, negotiation, specification, and validation.

**Task Set** defines the actual work to be done to accomplish the objectives of a software engineering action. Small ,relatively simple projects do not require task sets that are as large and  
detailed as complex projects team oriented project task sets .Task sets are adapted to meet the specific needs of a software project and the project team characteristics

**A process pattern** describes a process-related problem that is encountered during software engineering work, identifies the environment in which the problem has been encountered, and suggests one or more proven solutions to the problem.

Different approaches to **software process assessment and improvement** proposed include Standard CMMI Assessment Method for Process Improvement (phases: initiating, diagnosing, establishing, acting, and learning),CMM-Based Appraisal for Internal Process Improvement ( provides a diagnostic technique for assessing the relative maturity of a software organization),  
SPICE (ISO/IEC15504)( defines a set of requirements for software process assessment.), ISO 9001:2000 for Software(a generic standard that applies to any organization that wants to improve the overall quality of the products, systems, or services that it provides.)

**Chapter 4:PROCESS MODELS**

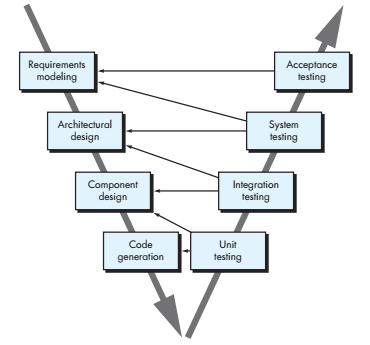
**A process model** provides a specific roadmap for software engineering work. It defines the flow of all activities, actions and tasks, the degree of iteration, the work products, and the organization of the work that must be done.

**Prescriptive process** models strive for bring order and structure to software development. Activities and tasks occur sequentially with defined guidelines for progress Each of these models suggests a somewhat different process flow, but all perform the same set of generic framework activities: communication, planning, modeling, construction, and deployment

**The Waterfall Model**: suggests a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through planning, modeling, construction, and deployment, culminating in ongoing support of the completed software

**The V-Model:**A variation in the representation of the waterfall model is called the V-model.

As a software team moves down the left side of the V, basic problem requirements are refined into progressively more detailed and technical representations of the problem and its solution. Once code has been generated, the team moves up the right side of the V, essentially performing a series of tests (quality assurance actions) that validate each of the models created as the team moves down the left side.



**Incremental Process** Models are iterative in nature and produce working versions of software quite rapidly.Deliver software in small but usable pieces, each piece builds on pieces already deliveredThe plan addresses the modification of the core product to better meet the needs of the customer and the delivery of additional features and functionality. This process is repeated following the delivery of each increment, until the complete product is produced

**Evolutionary Process** Models recognize the iterative, incremental nature of most software engineering projects and are designed to accommodate change.For example

1. **Prototyping** .A quick design focuses on a representation of those aspects of the software that will be visible to end users.The quick design leads to the construction of a prototype. The prototype is deployed and evaluated by stakeholders, who provide feedback that is used to further refine requirements. Iteration occurs as the prototype is tuned to satisfy the needs of various stakeholders, while at the same time enabling you to better understand what needs to be done.
2. **The Spiral Model**. :This approach couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model.Using the spiral model, software is developed in a series of evolutionary releases. During early iterations, the release might be a model or prototype. During later iterations, increasingly more complete versions of the engineered system are produced.

**Concurrent Models** :Allows a software team to represent iterative and concurrent elements of any of the process models

**Specialized process** models include **Component-Based Development** that emphasizes component reuse and assembly, **The Formal** Methods Model that encourages a mathematically based approach to software development and verification; and the **aspect-oriented** model that accommodates crosscutting concerns spanning the entire system architecture

**Unified Process** is a “use case driven, architecture centric, iterative and incremental” software process designed as a framework for UML methods and tools. Phases of the Unified Process include 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 in an effort to overcome perceived and actual weaknesses in conventional software engineering .One of the most compelling characteristics of the agile  
approach is its ability to reduce the costs of change through the software process.

**Agile development** stresses continuous communication and collaboration among developers and customers.It embraces a philosophy that encourages customer satisfaction, incremental software delivery, small project teams (composed of software engineers and stakeholders), informal methods, and minimal software engineering work products.

**Extreme programming** is the most widely used approach to agile software development. s. Extreme Programming uses an object-oriented approach as its preferred development paradigm and encompasses a set of rules and practices that occur within the context of four framework activities: planning, design, coding, and testing

**XP suggests** a number of innovative and powerful techniques that allow an agile team to create frequent software releases that deliver features and functionality that have been described and then prioritized by stakeholders

**Industrial XP** incorporates six new practices that are designed to help ensure that an XP project works successfully for significant projects within a large organization which are Readiness assessment, Project community, Project chartering ,Test-driven management, Retrospectives, Continuous learning.

Other agile process models also stress human collaboration and team self organization, but define their own framework activities and select different points of emphasis.

**Scrum**:Within each framework activity, work tasks occur within a process pattern called a sprint. The work conducted within a sprint (the number of sprints required for each framework activity will vary depending on product complexity and size) is adapted to the problem at hand and is defined and often modified in real time by the Scrum team. Scrum emphasizes the use of a set of software process patterns that have proven effective for projects with tight timelines, changing requirements, and business criticality. Each of these process patterns defines a set of development activities (Backlog ,Sprints,Scrum meetings, Demos) and allows the Scrum team to construct a process that is adapted to the needs of the project.

**Dynamic Systems Development Method**: This approach provides a framework for building and maintaining systems which meet tight time constraints through the use of incremental prototyping in a controlled project environment, Each increment only delivers enough functionality to move to the next increment. DSDM then defines three different iterative cycles: Functional model iteration, Design and build iteration, Implementation

**Agile Modeling (AM)** is a practice-based methodology for effective modeling and documentation of software-based systems.Modeling principles include Model with a purpose, Use multiple models, Travel light (only keep models with long-term value), Content is more important than representation ,Know the models and tools you use to create them ,Adapt locally. Agile modeling suggests that modeling is essential for all systems, but that the complexity, type, and size of the model must be tuned to the software to be built.

**Agile Unified Process**: Adopts the classic UP phased activities : inception, elaboration, construction, 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 to 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 ,Environment management.

**Chapter 6: HUMAN ASPECTS OF SOFTWARE ENGINEERING**

A software engineer has to master the technical stuff, learn and apply the skills required to understand the problem, design an effective solution, build the software, and test it in an effort to develop the highest quality possible.

**Characteristics of a software engineer** include :Individual responsibility for his commitments, Acute awareness(able to adapt his behavior to both the environment and people),Brutally honest ,Resilient under pressure, Heightened sense of fairness(Gladly shares credit with others),Exhibits attention to detail, Is pragmatic(recognizes that software engineering is not a religion in which dogmatic rules must be followed, but rather a discipline that can be adapted based on the circumstances at hand)

**The psychology of software engineering** includes individual cognition and motivation, the group dynamics of a software team, and the organization behavior of the company. In order to improve communication and collaboration, members of a software team can take on boundary-spanning role

**To be effective, a software team** must have a sense of purpose, a sense of involvement, a sense of trust, and a sense of improvement. In addition the team must avoid “toxicity” that is characterized by a frenzied and frustrating work atmosphere, an inappropriate software process, high frustration that causes friction among team member , an unclear definition of roles on the software team, and continuous exposure to failure.

**Factors considered when planning** the team structure of software engineering teams include difficulty of the problem to be solved, “size” of the resultant program(s) in lines of code or function points ,time that the team will stay together (team lifetime) ,degree to which the problem can be modularized, required quality and reliability of the system to be built, rigidity of the delivery date, and degree of sociability (communication) required for the project.

**Organizational paradigms** for software engineering teams include closed paradigm structures (team along a traditional hierarchy of authority), random paradigm structures (prefer a loose structure that relies on individual initiative), as well as open paradigm and synchronous paradigm

Agile teams subscribe to the agile philosophy and generally have more autonomy than more conventional software teams. Agile teams emphasize communication, simplicity, feedback, courage, and respect.

**Social media** is becoming an integral part of many software projects. Blogs, microblogs, forums, and social networking capabilities help to form a software engineering community that communicates and coordinates more effectively.

**Cloud computing** has the potential to influence the manner in which software engineers organize their teams, the way they do their work, the manner in which they communicate and connect, and the way software projects are managed. In situations in which the cloud can enhance the social and collaborative aspects of software development, its benefits far outweigh its risks.

**Collaborative development environments** contain a number of services that enhance communication and collaboration for a software team. These environments are particularly useful for global software development where geographic separation can precipitate barriers to successful software engineering.