LECTURE HANDOUTS

MCA **I / II**

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **I - Introduction**

Date of Lecture: 22.02.2021

Topic of Lecture: Software Engineering paradigms

Introduction :

- Software is more than just a program code.
- A program is an executable code, which servers some computational purpose.
- Software is the collection of computer programs, procedures rules and associated documentation and data.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Class and Object, Message, operation and method, Encapsulation, Abstraction, Inheritance, Polymorphism

Detailed content of the Lecture:

Software Engineering paradigms:

- Software is an information transformer- producing, managing, modifying, displaying or transforming information that can simple as a single bit or a complex as a multimedia application.

Software Products:

- Software products may be developed for a particular customer or may be developed for a general market.

Software products may be:

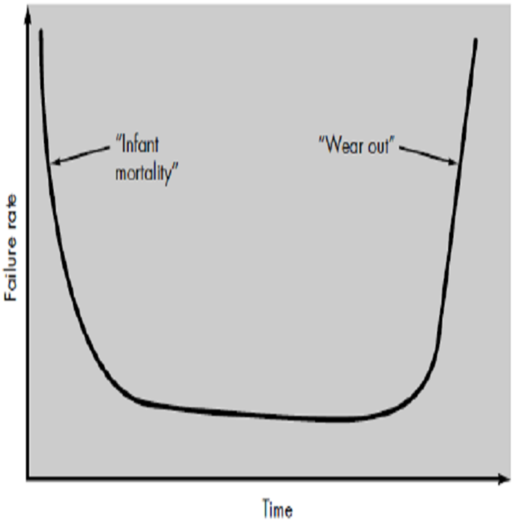
- Generic
- Bespoke

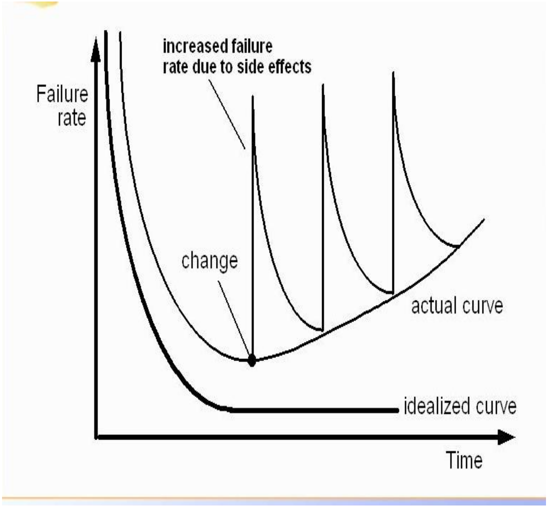
What are the attributes of good software?

- Maintainability.
- Dependability
- Efficiency
- Usability

Software Characteristics:

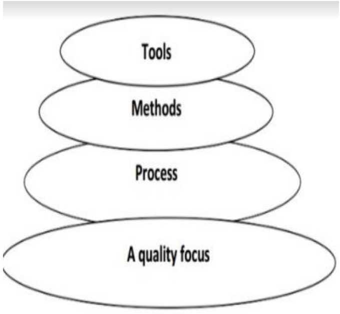
- Software is developed or engineered, it is not manufactured in the classical sence.
- Software doesn't "wear out".
- Although the industry is moving towards component based assembly, most software continues to be custom to built.





Software Engineering -A layered Technology:

- Application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software that is, the application of engineering software.



Software Engineering Layers

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=1MGNuW0HjFA

Important Books/Journals for further learning including the page nos.:
Software Engineering: A Practitioner Approach–Roger S. Pressman , McGraw Hil, 2010 **(Page No : 36 - 40)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs.R.Pavithra**

Unit : **I - Introduction**

Date of Lecture: 23.02.2021

Topic of Lecture: Waterfall Life cycle mode

Introduction :

- It is called classic life cycle or Linear model.
- Requirements are well defined and stable.
- It suggests a systematic, sequential approach to software development.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Class, Object, Attributes, Function

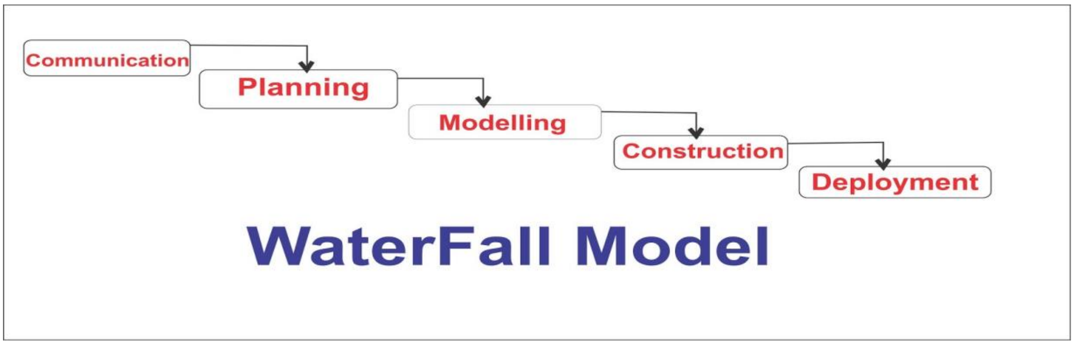
Detailed content of the Lecture:

Process Models:

- Every software engineering organization should describe a unique set of framework activities for the software process it adopts.
- Waterfall Life Cycle Model.
- Iterative Waterfall Life Cycle Model.
- Prototyping Model.
- Incremental Model.
- Sprial Model.
- RAD Model.
- Sprial Model.

Waterfall Life Cycle Model:

- It suggests a systematic, sequential approach to software development.
- It begins with customer specification of requirements and progresses.
- Planning.
- Modeling.
- Construction and
- Deployment.



Advantages:

- Easy to understand.
- Each phase has well defined input and output.
- Helps project manger in proper planning of project.
- Provides a templates into which methods of analysis, design, code and support can be placed.

Disadvantages:

- One way street.
- It lack overlapping and interactions among phases.
- Model doesn't support delivery of system in pieces.

Feasibility Study:

- It involves analysis of the problem and collection of all relevant information relating to the product.
- The collected data are analyzed.
- Requirement of the Customer.
- Formulations of the different strategies for solving the problem.
- Evaluation of different solution strategies.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=x-jqSXYE4S4

Important Books/Journals for further learning including the page nos.:
Software Engineering: A Practitioner Approach–Roger S. Pressman , McGraw Hil, 2010 **(Page No : 78 - 86)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code **: SOFTWARE ENGINEERING / 19CAB10**

Course Faculty **: Mrs. R.Pavithra**

Unit **: I -Introduction.**

Date of Lecture: 24.02.2021

Topic of Lecture: Spiral Model

Introduction :

- This Spiral model is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Classes, Objects, Methods, Data

Detailed content of the Lecture:

Spiral Model:

- This Spiral model is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis.
- The spiral model has four phases: Planning, Design, Construct and Evaluation.

Advantages :

- Risk Identification at early stage.
- Suitable for high rk projects.
- Flexibility for adding functionality.

Disadvantages:

- Costly.
- Risk dependent.
- Not suitable for smaller projects.
- Difficult to meeting budget.

Win-Win Sprial Model:

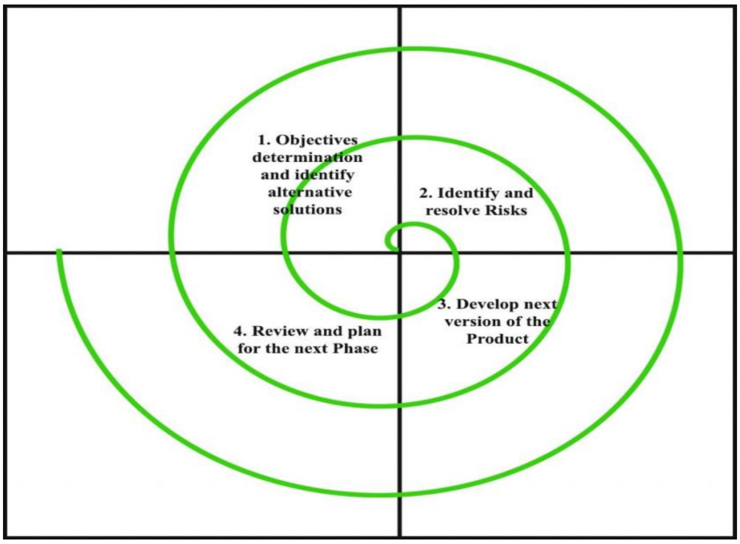
- The customer wins by getting the system satisfying most of thier requirements and developers wins by working on achievable budgets and deadlines.

Advantages:

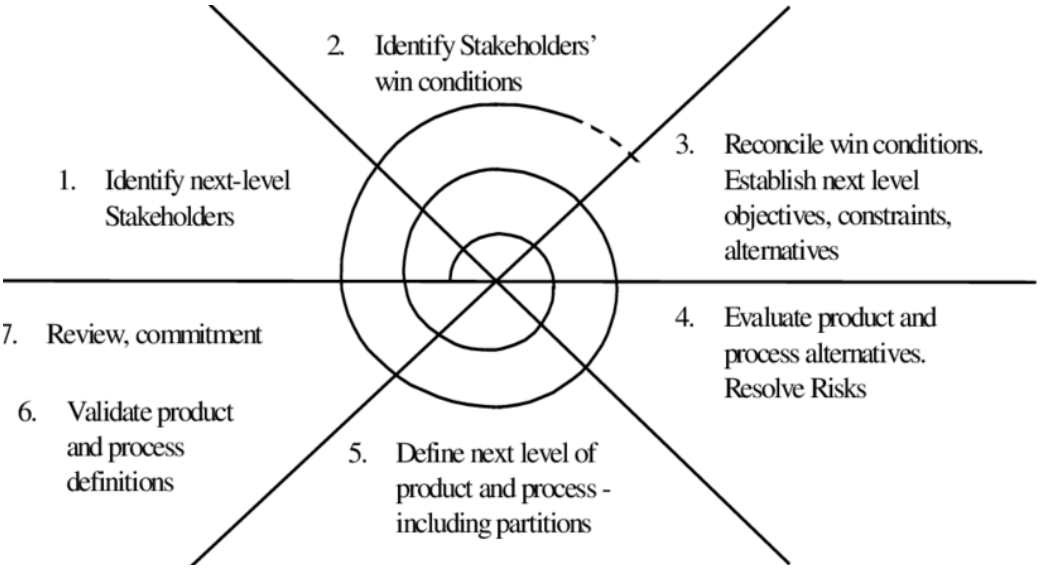
- Lieght weight methods suit small-medium size project.
- Produces good team.

- Test based approach to requirements and quality assurance

<u>Quadrants in sprial model</u>


1. Objectives determination and identify alternative solutions
2. Identify and resolve Risks
3. Develop next version of the Product
4. Review and plan for the next Phase

<u>Win-Win Sprial Model:</u>


1. Identify next-level Stakeholders
2. Identify Stakeholders' win conditions
3. Reconcile win conditions. Establish next level objectives, constraints, alternatives
4. Evaluate product and process alternatives. Resolve Risks
5. Define next level of product and process - including partitions
6. Validate product and process definitions
7. Review, commitment

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=YfGvIhPXz1A

Important Books/Journals for further learning including the page nos.:
Software Engineering: A Practitioner Approach–Roger S. Pressman , McGraw Hil, 2010 **(Page No : 98-103)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code	**: SOFTWARE ENGINEERING / 19CAB10**
Course Faculty	**: Mrs. R.Pavithra**
Unit	**: I - Introduction**

Date of Lecture:25.02.2021

Topic of Lecture: Prototype Model

Introduction :

- Prototyping Model is a software development model in which prototype is built, tested, and reworked until an acceptable prototype is achieved.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Building blocks of object-oriented modeling, Link, Dynamic structure of the system

Detailed content of the Lecture:

Prototype Model :

- The prototyping model is a systems development method in which a prototype is built, tested and then reworked as necessary until an acceptable outcome is achieved from which the complete system or product can be developed.

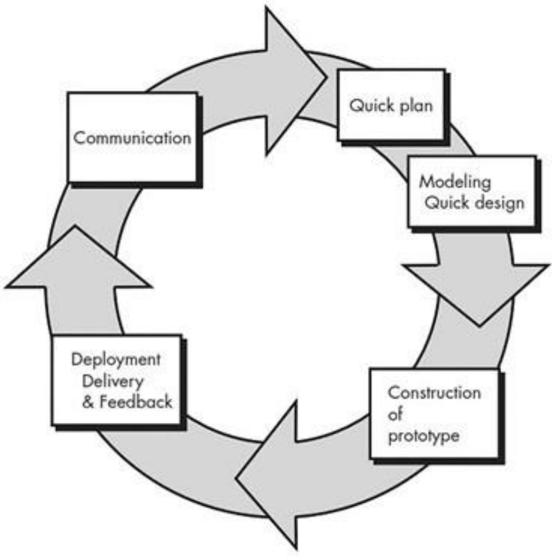


Figure: Prototype Model

Advantages:

- Clarity.
- Risk Identification.
- Good Environment.
- Take less time to complete.

Disadvantages:

- High cost.
- Slow process.
- Too many changes.

RAD Model:

- Rapid Application Development(RAD) is an incremental software model that a short development cycle.
- The RAD model is a “high-speed” of the waterfall model.
- The RAD process enables a development team to create a fully functional system within a very short time period.

Contents of RAD Packages:

- Graphical user development environment.
- Reusable Components.
- Code generator.
- Programming Language.

Advantages:

- Fast products.
- Efficient Documentation.
- Interaction with user.

Disadvantages:

- User may not like fast activities.
- Not suitable for technical risks.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=JhHkb7z5GzY

Important Books/Journals for further learning including the page nos.:
Software Engineering: A Practitioner Approach–Roger S. Pressman , McGraw Hil, 2010 **(Page No : 67-73)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **I - Introduction**

Date of Lecture:26.02.2021

Topic of Lecture: Fourth Generation Techniques

Introduction :

- The tools in automatically generate source code based on the developers specification.
- The term fourth generation techniques (4GT) encompasses a broad array of software tools that have one thing in common: each enables the software engineer to specify some characteristic of software at a high level.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Relationship, Class, Object, Interface

Detailed content of the Lecture:

Fourth Generation Techniques:

- fourth-generation language (4GL), computer programming language that is intended to be easier for users than machine languages (first-generation), assembly languages (second-generation), and the older high-level languages.

Software development environment that supports the 4GT paradigm includes some or all of the following tools:

- Non-procedural languages for database query
- Report generation
- Data manipulation
- Screen interaction and definition
- Code generation and High-level graphics capability
- Spreadsheet capability
- Automated generation of HTML and similar languages used for Web-site creation using advanced software tools.

Advantages:

- Reduction in software development.
- Improved productivity of software engineers.
- 4GT helped by CASE tools and code generators.

Disadvantages:

- Some 4GT are not at all easier than programming languages.
- Generated source code are sometimes inefficient.
- Time is reduced for only small and medium projects.

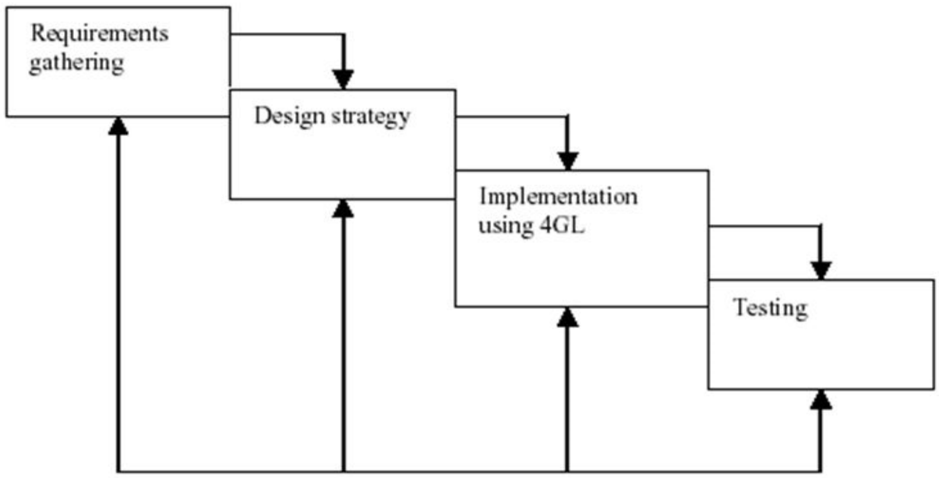


Figure 1.7: Fourth Generation Techniques

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=_Q5RWV8Cniw

Important Books/Journals for further learning including the page nos.:
Software Engineering: A Practitioner Approach–Roger S. Pressman , McGraw Hil, 2010 **(Page No : 153-161)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

I / II

MCA

Course Name with Code :**SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **I - Introduction**

Date of Lecture: 01.03.2021

Topic of Lecture: Planning

Introduction :

- Planning may be defined as deciding in advance what to be done in future.
- It is the process of thinking before doing.
- It involves determination of goals as well as the activities required to be undertaken to achieve the goals.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Behavior of the system, Role, Function, Abstraction

Detailed content of the Lecture:

Planning:

- Software planning process include steps to estimate the size of the software work products and the resources needed produces a schedule identify and access software risks.

During planning a project is split into several activities :

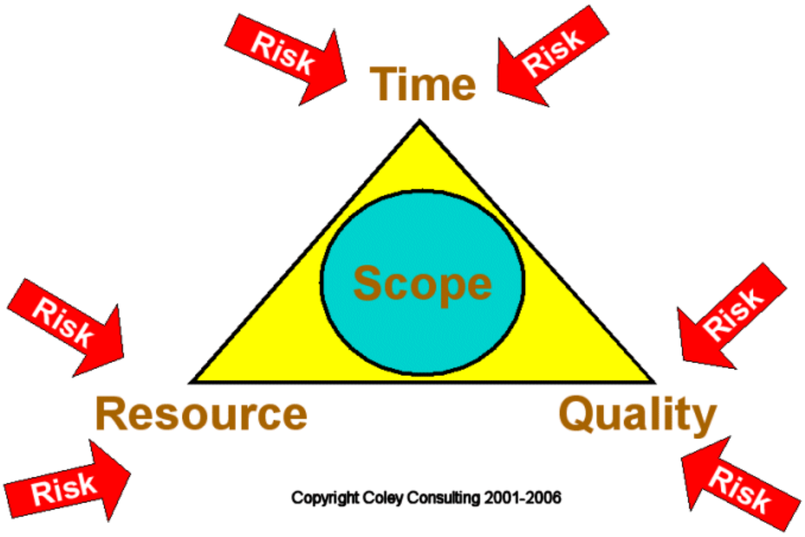
- How much efforts is required to complete each activities?
- How much calender time is needed?
- How much will the completed activity cost?

Planning Objectives:

- Understand the scope of the problem.
- Make use of past historical data.
- Estimate effort or function or size.
- Define a project schedule.

Characteristics of software project planning:

- Scope.
- Resource.
- Time.
- Quality.
- Risk.



Copyright Coley Consulting 2001-2006

Project Plan:

- The biggest single problem that afflicts software developing is that of underestimating resources required for a project.
- According to the project management body of knowledge.
- According to PRINCE(PRojects IN Controlled Environments).

Types of Project Plan:

- Software development plan.
- Quality Assurance Plan.
- Validation Plan.
- Configuration Management Plan.
- Maintenance Plan.
- Staff development plan.

Structure of a software project management plan:

- Project summary.
- Project planning.

Major issues in planning a software project:

- Software requirements are frequently incorrect and incomplete.
- Planning schedule and cost are not updated and are based on marketing needs, not system requirements.

Video Content / Details of website for further learning (if any):
https://backlinko.com/hub/youtube/planning

Important Books/Journals for further learning including the page nos.:
Software Engineering: A Practitioner Approach–Roger S. Pressman , McGraw Hil, 2010 **(Page No : 176-182)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

DESIGNING YOUR FUTURE
Estd. 2000

IQAC

LECTURE HANDOUTS

MCA

I / II

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **I - Introduction**

Date of Lecture: 02.03.2021

Topic of Lecture : Software Project Scheduling

Introduction :

- Software project scheduling is an action that distributes estimated effort across the planned project duration by allocating the effort to specific software engineering tasks. The schedule evolves over time.
- During early stages of project planning, a macroscopic schedule is developed.

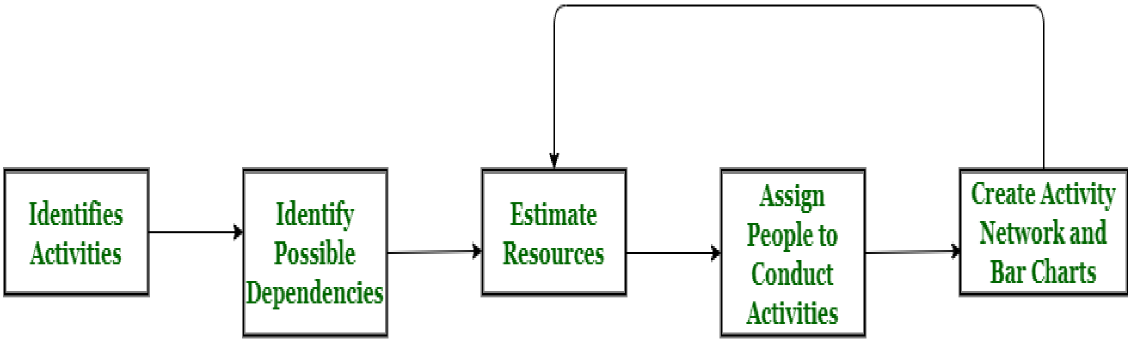
Prerequisite knowledge for Complete understanding and learning of Topic:

- Object, Memory, Databases, OODBMS

Detailed content of the Lecture:

<u>Software Project Scheduling :</u>

- Software project scheduling is an distributes estimated effort across the planned project.
- Project scheduling involves separating the total work involved in a project in separate activities and judging the time required to complete the activities.


Identifies Activities
Identify Possible Dependencies
Estimate Resources
Assign People to Conduct Activities
Create Activity Network and Bar Charts

Project Scheduling Process

<u>Basic principles of software project scheduling:</u>

- Compartmentalization.
- Interdependency.
- Time Allocation.
- Effort Validation.
- Defined Responsibilities.
- Defined outcomes.
- Defined Milestones.

<u>Relationship between people and effort:</u>

- The PNR curve provides an indication of the relationship between effort applied and delivery time for a software project.

Effort Distribution:

- A recommended distribution of effort the software process is often referred to as the 40-20-40 rule.

Defining a task set for the software project:

- A task set is a collection of software engineering work tasks, milestones, and work products that must be acomplished to complete a particular project.
- Concept Development projects.
- New application development projects.
- Application enhancement projects.
- Application maintenance projects.
- Re-Engineering projects.

Example of a task set:

- Concept Scoping: It determines the overall scope of the project.
- Preliminary concept planning: It establishes the organization ability to undertake the work implied by the project scope.
- Technology Risk Assessment: It evaluates the risk associated with the technology to be implemented as part of project scope.
- Concept Implementation: It implement the concept representation in a manner that can be reviewed by a customer and is used marketing purposes.
- Customer Reaction: Customer reaction to the concept feedback on a new technology concept and target specific customer applications.

Scheduling Techniques:

- Scheduling of a software project does not differ greatly from scheduling of any multitask engineering effort.
- Work Breakdown Structure(WBS).
- Activity Charts.
- Project Evaluation Review Techniques(PERT).
- Grant Charts.
- Critical Path Method(CPM).

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=SkQzQCAWf8M

Important Books/Journals for further learning including the page nos.:
Software Engineering: A Practitioner Approach–Roger S. Pressman , McGraw Hil, 2010 **(Page No : 134-139)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

MCA

I / II

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **I - Introduction**

Date of Lecture: 03.03.2021

Topic of Lecture: Risk analysis and management

Introduction :

- Risk Analysis is a proven way of identifying and assessing factors that could negatively affect the success of a business or project.
- It allows you to examine the risks that you or your organization face, and helps you decide whether or not to move forward with a decision.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Classes
- Object Oriented Programming
- Groups
- Relationship

Detailed content of the Lecture:

Risk analysis and management:

- Risk analysis and management are a series of steps that help a software team to understand and manage uncertainty during the development process..
- A risk is a potential problem.
- Managers, Software enginners and customers participate in risk analysis & management.

Software Risk:

- According to webster risk is the possibility of suffering loss.
- Risk in a project or program is a measure of the ability to achieve objectives within cost, schedule and constraints.

Types of software risk:

- Classification I
- Classification II

Classification I:

- Project Risks
- Technical Risks.
- Business Risks.

Classification II:

- Known Risks.
- Predictable Risks.
- Unperdictable Risks.

Classification I:

- **Project Risks:** The projct schedule will slip and that costs will increase.Project risks identify schedule, resource, customer and requirements problem.
- **Technical Risks:**The product quality and the timeliness of the schedule if a technical risks is real then implementation may become difficult or impossible.
- If identify potential design, implementation, interface, verification and maintenance problems.

Business Risks:

- Market Risk.
- Strategic Risk.
- Management Risk.
- Budget Risk.

Classification II:

- **Known Risks:** That can be uncovered after careful evaluation of the project plan.
- **Predictable Risks:**Predictable Risks are extrapolated from past project experience.

Unperdictable Risks: Unperdictable Risks are the joker in the desk, they can extremely difficult to identify in advance.

Risk Principles:

- Global Perspective.
- Forward looking view.
- Open communication.
- Integrated management.
- Continuous process.
- Shared product vision.
- Team work.

Risk Strategies:

- Reactive Risk Strategies.
- Proactive Risk Strategies.

Risks in software development projects:

- Poorly defined requirements.
- Client requirements changes.
- Poor techniques for cost estimation.
- Dependence on skills of individual developers.

Risk Management Process:

- The risk management activities includes identify, analysis, plan, track and control risks.
- Risk Assessment.
- Risk Control.

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=ku0DvsnsLCA

Important Books/Journals for further learning including the page nos.:

Software Engineering: A Practitioner Approach–Roger S. Pressman , McGraw Hil, 2010 **(Page No : 124-129)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

DESIGNING YOUR FUTURE
Estd. 2000

IQAC

LECTURE HANDOUTS

MCA

I / II

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **I - Introduction**

Date of Lecture: 04.03.2021

Topic of Lecture: Requirements and Specification

Introduction :

- A Requirement is a statement of one thing a product must do or a quality it must have.
- A Requirement Specification is a collection of the set of all requirements that are to be imposed on the design and verification of the product.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Analysis, Design, Implementation, Testing, Maintenance

Detailed content of the Lecture:

Requirements and Specification:

- Requirement engineering is the sub-discipline of software engineering that is concerned with determine the goal, functions and constraints of software system.

Requirements:

- Requirements management is a systematic approach to eliciting organizing and documenting the requirements of the systems.

Types of Requirements:

- System Requirements.
- User Requirements.

System Requirements:

- System requirements set out the systems functions, services and operational constraints in detail.
- It may be part of the construct between the system buyer and the software developer.

Types of system requirements:

- Functional requirements.
- Non-functional Requirements.
- Domain Requirements.

Functional Requirements:

- The customer should provide statement of service. It should be clear how the system should react to particular inputs and how a particular system.

Problem of Functional Requirements:

- User Intention.
- Developer Interpretation.
- Requirements completeness and consistency.

Non-Functional Requirements:

- The system properties and constraints various properties of a system can be: Reliability, response time, storage requirements.

Types of Non-Functional Requirements:

- Product Requirements.
- Organizational Requirements.
- External Requirements.

Domain Requirements:

- Requirements can be application domain of the system, reflecting, characteristics of the domain.

Problem of Domain Requements:

- Understandability.
- Implicitness.

User Requirements:

- User requirements are defined using natural language labels and diagrams because these are the representation that can be understood by all users.

Software Requirement Specification:

- Software Requirements document is the specification of the system.
- It is not a design document.
- Requirements document is called SRS.

Users of SRS:

- Users, Customer and marketing Personnel.
- Software Developers.
- Test Engineers.
- Project Managers.
- Maintenance Engineers.

Video Content / Details of website for further learning (if any):

https://support.google.com/youtube/answer/78358?hl=en

Important Books/Journals for further learning including the page nos.:
Software Engineering: A Practitioner Approach–Roger S. Pressman , McGraw Hil, 2010 **(Page No : 224-229)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

DESIGNING YOUR FUTURE
Estd. 2000

IQAC

LECTURE HANDOUTS

MCA

I / II

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **II - SOFTWARE DESIGN**

Date of Lecture: 05.03.2021

Topic of Lecture: Abstraction, Modularity

Introduction :

- Design is a meaningful representation of something that is to be built it can traced to a customer requirements and at the same time assessed for quality against a set of predefined for “ good

Prerequisite knowledge for Complete understanding and learning of Topic:

- Object, Model, Tool, Software

Detailed content of the Lecture:

Design Principles:

- Process should not suffer tunnel vision.
- Traceable to analysis model
- Not reinvent the wheel.
- Minimize the intellectual distance between the software problem.
- Design is not a coding.
- Reviewed to minimize conceptual errors.

Objectives of Software:

- Correctness.
- Understandability.
- Modularity.
- Completeness.
- Efficiency.
- Consistency.
- Verifiability.

Types of Design:

- Preliminary Design.
- Detailed Design.

Abstraction, Modularity :

Abstraction:

Focus on solving a problem without being concerned about lower level details.

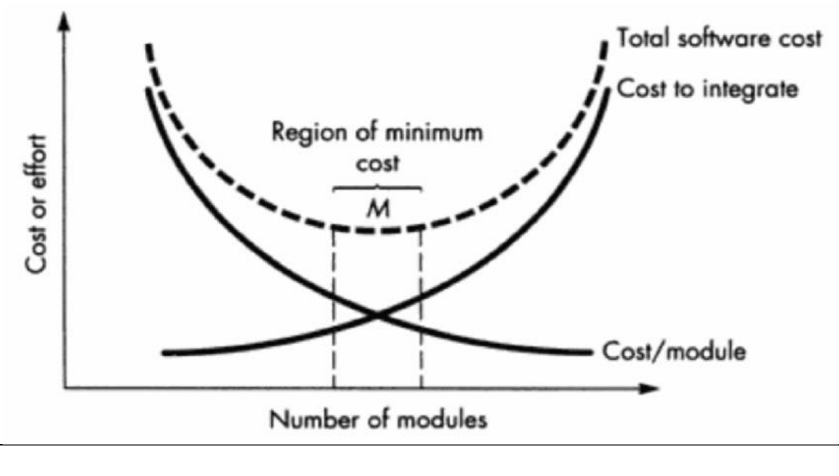
At the highest level of abstraction, a solution is stated in board teams using the language of the problem environment.

Three types of Abstraction:

- Procedural Abstraction.
- Data Abstraction.
- Control Abstraction.

Modularity:

- A system is a modular if it is composed of well defined conceptually simple and independent units interacting well defined interfaces.



Function Independence:

- It is a achieve by the developing modules with single minded function and an a version to excessive interaction with other modules.

Reasons:

- Error Isolation.
- Scope for reuse.
- Understand ability.

Stepwise Refinement:

- It is a top-down design strategy.
- A program developed by procedural details.
- A hierarchy is developed by programming language statements.

Control Hierarchy:

- Control Hierarchy also called program structure.
- The software such as sequence of processes.
- It is a expression of control relationship among module super-ordinate and sub-ordinate module.

Inforamation Hiding:

- Information hiding is a fundamental design concept for software.
- The software system is designed using the information hiding approaches each module in the system hides the internal details for the processing activities.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=Bv84KQGYtr8

Important Books/Journals for further learning including the page nos.:
Software Engineering–Sommerville ,Addison Wesley-Longman, 2004 **(Page No : 24-29)**

Course Faculty

Verified by HOD

DESIGNING YOUR FUTURE
Estd. 2000

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

IQAC

LECTURE HANDOUTS

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs.R.Pavithra**

Unit : **II - SOFTWARE DESIGN**

Date of Lecture: 06.03.2021

Topic of Lecture: Software Architecture

Introduction :

- Software architecture is, simply, the organization of a system.
- This organization includes all components, how they interact with each other, the environment in which they operate, and the principles used to design the software. In many cases, it can also include the evolution of the software into the future.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Object Oriented Concepts
- Modeling
- Software
- Analysis

Detailed content of the Lecture:

Software Architecture:

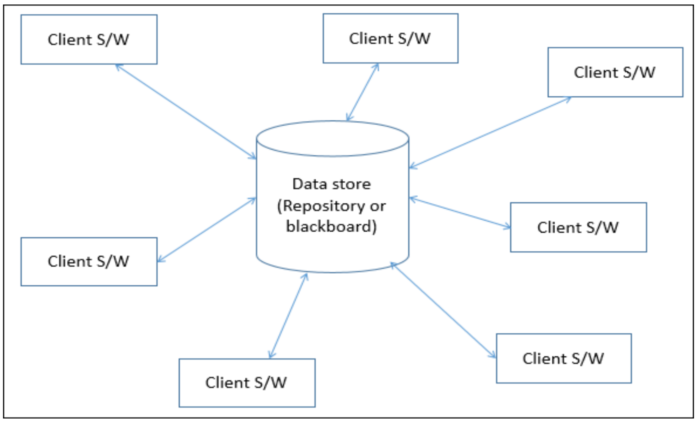
- The overall structure of the components software and the way in which that structure provides conceptual integrity for a system.
- Structural Properties.
- Extra-functional Properties.
- Families of Related Systems.

Architecture Styles and Patterns:

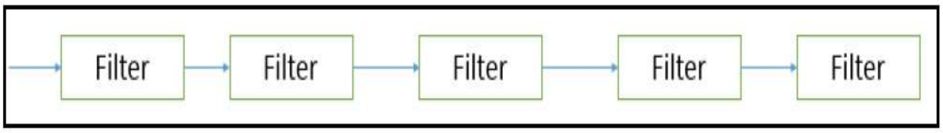
- A set of components that perform a function required by a system.
- A set of connectors that cable communication, coordination and cooperation among components.
- Data-centered architectures.
- Data flow architectures.
- Call and return architectures.
- Object-oriented architectures.
- Layered architectures.

Data-centered architectures:

- A data store resides at the center of this architecture and is accessed frequently by other components that updates, add, delete, or otherwise modify data within the store.



Data Flow Architectures:
The architectures is applied when input data a transformed through a series of computational or manipulative components into output data.



Call and return architectures:

- This architectures style enables a software designer to achieve a program structure that is relatively easy to modify and scale.
- A number of substyles exists within this category.

Remote procedure call architectures:

- The components of a main program/ subprogram architecture are distributed across multiple computers on a network.

Object-oriented Architectures:

- The components of system encapsulate data and the operations that must be applied to manipulate the data.
- Communication and coordination between components is accomplished message passing.

Layered Architectures:

- The basic structure of a layered architecture.
- A number of different layers are defined each operation that the machine instruction set,
- At the outer layer components services user interface operations.
- At the inner layer components perform operating system interfacing intermediate layer provide utility services and application software function.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=lTkL1oIMiaU

Important Books/Journals for further learning including the page nos.:
Software Engineering–Sommerville ,Addison Wesley-Longman, 2004 **(Page No : 124-129)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

MCA

I / II

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **II - SOFTWARE DESIGN**

Date of Lecture: 08.03.2021

Topic of Lecture: Cohesion , Coupling

Introduction :

- The “Single-Mindedness” of a module.
- Cohesion is a measure of functional strength of a module.
- The degree to which a componentt is connected to other components and to the external world.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Modeling, SDLC, Object, Class, System

Detailed content of the Lecture:

Cohesion :

- Cohesion means sticking together. If your group of friends heads to the lunchroom as a team and sits all together, you're demonstrating strong cohesion.
- Cohesion is a word that comes to us through physics, where cohesion describes particles that are the same and tend to stick together
- The “Single-Mindedness” of a module.
- Cohesion is a measure of functional strength of a module.

Types of Cohesion:

- Communicational cohesion.
- Sequential cohesion.
- Procedural cohesion.
- Temporal cohesion.
- Utility cohesion.

Communicational cohesion:

- All operations that access the same data are defined within one class.The data in question, accessing and storing.
-

Sequential Cohesion:

- Components or operations are grouped in a manner that allow the first to provide input to the next.

Procedural Cohesion:

- That allows one to be invoked immediately after the preceding one was invoked even when there is no data.

Temporal Cohesion:

- Operations that are performed to reflect a speecifc behavior or state.

Utility Cohesion:

- Classes or operations that exist within the same category but are otherwise unrelatedare grouped together.

Coupling:

- The degree to which a componentt is connected to other components and to the external world.

Types of Coupling:

- Data Coupling.
- Stamp Coupling.
- Control Coupling.
- Common Coupling.
- Content Coupling.

Data Coupling:

- Operations pass long strings of data argument.

Stamp Coupling:

- Class B is declared as a type for an arguments of an operation of class A.

Control Coupling:

- Operation A() invokes operation B() and passes a control flag to B. The control flag then directs logical flow within B.

Common Coupling:

- Number of components all make use of a global variable.

Content Coupling:

- One components secretly modifies data that is internal to another components.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=bKYlMcyuvcM

Important Books/Journals for further learning including the page nos.:
Software Engineering–Sommerville ,Addison Wesley-Longman, 2004 **(Page No : 159-162)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

MCA

I / II

Course Name with Code : SOFTWARE ENGINEERING / 19CAB10

Course Faculty : **Mrs. R.Pavithra**

Unit : **II - SOFTWARE DESIGN**

Date of Lecture:09.03.2021

Topic of Lecture: Various Design Concepts and notations

Introduction :

- A design of the software must be modular i.e the software must be logically partitioned into elements.
- In design, the representation of data , architecture, interface and components should be distinct. A design must carry appropriate data structure and recognizable data patterns.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Modeling, Object, Class, System

Detailed content of the Lecture:

Various Design Concepts and notations:

Patterns:

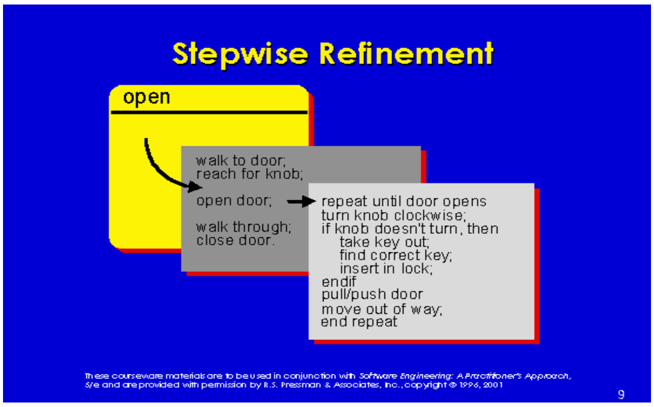
- A Pattern is a named of insight which conveys the proven solution to a recurring problem within a certain computing.
- A design pattern describes a design structure that solves a particular design problem within a specific context and forces that may have a impact on the manner.

Information Hiding:

- Information contained within a module is inaccessible to modules that have no need for such information.

Refinement:

- Refinement begins with a statement of function.
- That is defined at a high level of abstraction.
- Abstraction enables a designer to specify procedure and data and support low level.



Stepwise Refinement

```
open
```

```
walk to door;
reach for knob;

open door;

walk through;
close door.
```

```
repeat until door opens
turn knob clockwise;
if knob doesn't turn, then
    take key out;
    find correct key;
    insert in lock;
endif
pull/push door
move out of way;
end repeat
```

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach,* 5/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001

9

Notations:

Development Diagram:

- Each 3D box a hardware element that is part of the physical architecture of the system.

Class Diagram:

- Represent system level elements in terms of the data that describe the element and the operations that manipulate the data.

Use-case diagram:

- A use-case tells a story about how an end-user interacts with the system under a specific set of circumstances.

Sequence Diagram:

- It indicates how events cause transitions from object to object.
- Each of the arrow represent an event and indicates how the event channel.

- Time is measured vertically and thw narrow vertical rectangle represent time spend in processing an activity.

Graphical Design Notation:

- The foundation of components level design for conventional software components were formed in the early 1960s.
- The constructs are sequence, condition and repetition.

Tabular Design Notation:

- Decision tables provide a notation that translate actions and conditions into a tabular form.
- List all condition during execution of the procedure.

HIPO Diagram:

- HIPO diagram (Hierarchy-Process-Input-Output) were developed at IBM as design representation schemes for top-down software development and as external documentation.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=95nIM9vZSAQ

Important Books/Journals for further learning including the page nos.:
Software Engineering–Sommerville ,Addison Wesley-Longman, 2004 **(Page No : 169-173)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

MCA

I / II

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **II - SOFTWARE DESIGN**

Date of Lecture:10.03.2021

Topic of Lecture: Real time and Distributed System Design

Introduction :

- A distributed real-time system (DRTS) consists of autonomous computing nodes connected by a real-time network.
- Nodes in such a system cooperate to achieve a common goal within specified deadlines.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Tools
- Packages
- Modeling
- Open-Source Software

Detailed content of the Lecture:

Real time and Distributed System Design:

- Real time system is any system that responds in a timely manner.
- Activities must be scheduled and executed to meet their timeliness requirements.

Characteristics of real time system:

- Timeliness.
- Embedded System.
- Concurrency.
- Dependability.

Distributed System Design:

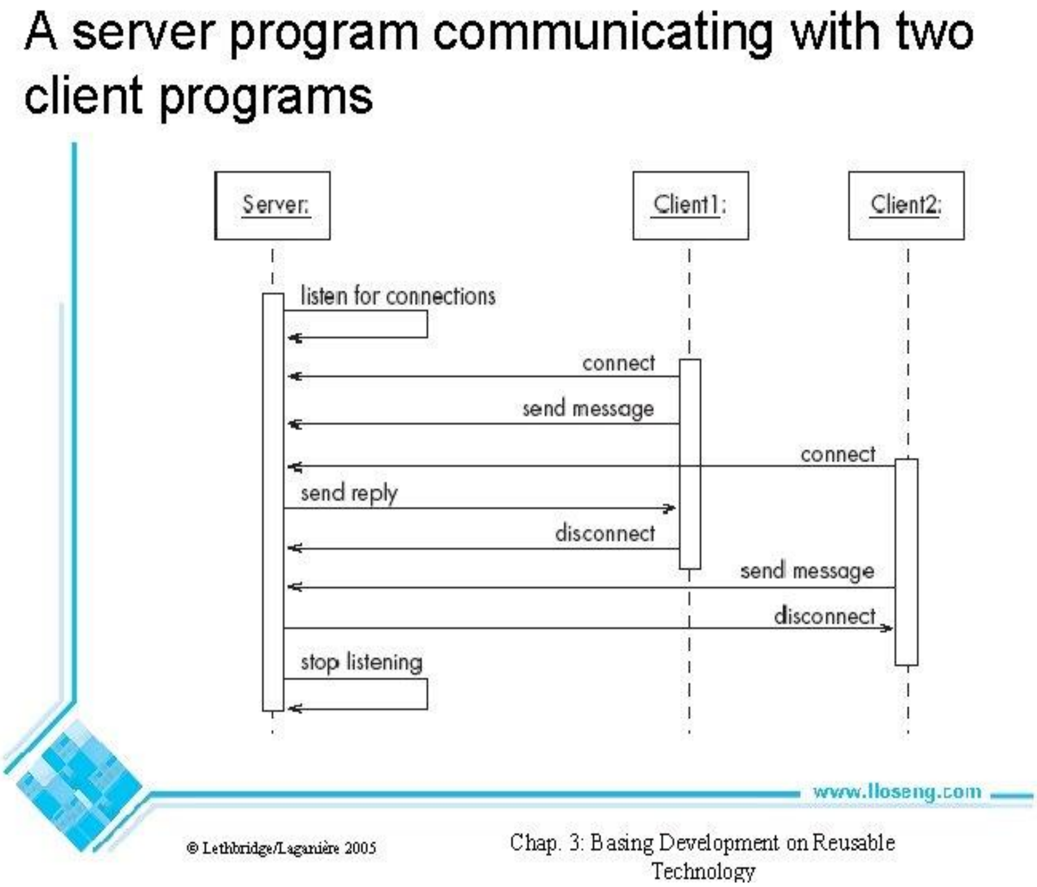
- A distributed system is a collection of computational and storage devices connected through a communication network.
- Network Topology.
- Communication Mode.

Advantages:

- Resource Sharing.
- Enhanced Performance.
- Improved Reliability and availability.
- Modular Expandability.

Disadvantages:

- Complexity.
- Manageability.
- Security.
- Unpredictability.



A server program communicating with two client programs

© Lethbridge/Laganière 2005

Chap. 3: Basing Development on Reusable Technology

Video Content / Details of website for further learning (if any): https://www.youtube.com/watch?v=dZ3swmtR1As
Important Books/Journals for further learning including the page nos.: Software Engineering–Sommerville ,Addison Wesley-Longman, 2004 **(Page No : 189-193)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

I / II

Course Name with Code **: SOFTWARE ENGINEERING / 19CAB10**

Course Faculty **: Mrs. R.Pavithra**

Unit **: II - SOFTWARE DESIGN**

Date of Lecture: 11.03.2021

Topic of Lecture: Documentation , Dataflow Oriented design

Introduction :

- Computer software include the source code for a system and all the supporting.
- A data flow diagram is a graphical representation of the flow of data through an information system.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Class
- Object
- Methods
- Link
- Relationship

Detailed content of the Lecture:

Documentation :

- Material that provides official information or evidence or that serves as a record."you will have to complete the relevant documentation"
- Documents generate during analysis, design, implementation, testing and maintenance.

Writing Good Design Document:

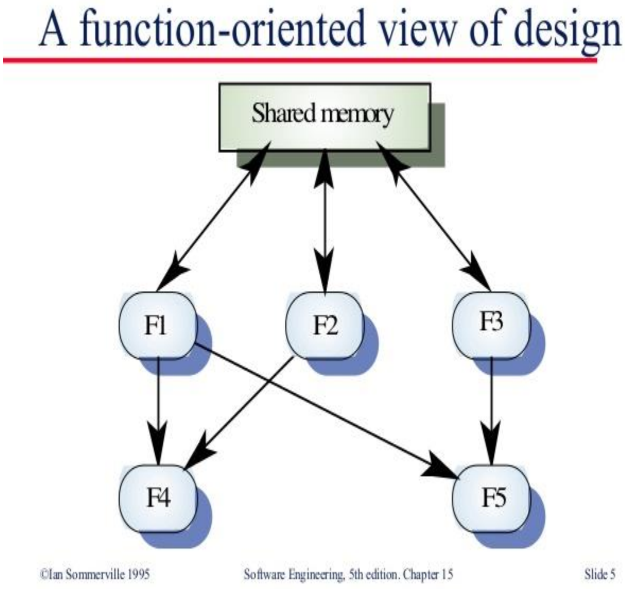
- Design document as a means of communication.
- Purpose.
- General Priorities.
- Outline of the design.
- Major design issues.
- Other details of the design

Some useful documents are,

- Process and product documentation.
- User documentation.
- System documentation.

Data Flow Oriented Modeling:

- A data flow oriented design approach relies on decomposing the system into a set of interacting functions with a centralized system state shared by these function.



A function-oriented view of design

©Ian Sommerville 1995 Software Engineering, 5th edition. Chapter 15 Slide 5

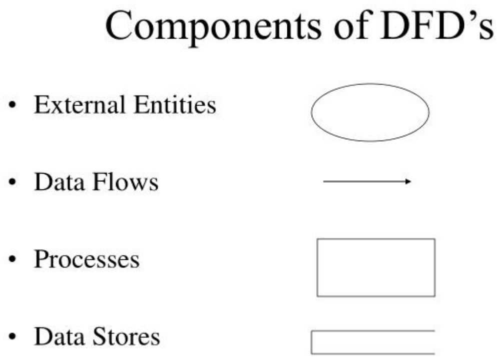
Function Oriented Design Techniques:

- Data flow Diagram.
- Data Dictionary.
- Data Structure Code.
- Pseudo-code.
- Flow Chart.

Data flow Diagram:

- A data flow diagram is a graphical representation of the flow of data through an information system.

Components of Data flow diagram:



Components of DFD's

- External Entities
- Data Flows
- Processes
- Data Stores

Data Dictionary:

- The data dictionary is an organized listing of all data elements .
- That both user and system analyst will have a common understanding of all inputs, components of

store and intermediate calculation.

Pseudo-Code:

- It is another program analysis tool that is used for planning program logic.
- “Pseudo” means imitation or flase and “Code” refers to thr written a programming language.

Logical Structures:

- Sequence.
- Selection.
- Iteration.

Flow chart:

- The flow chart use standard symbols are write operations.

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=6VGTvgaJllM

Important Books/Journals for further learning including the page nos.:
Software Engineering–Sommerville ,Addison Wesley-Longman, 2004 **(Page No : 212-217)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code **: SOFTWARE ENGINEERING / 19CAB10**

Course Faculty **: Mrs. R.Pavithra**

Unit **: II - SOFTWARE DESIGN**

Date of Lecture: 12.03.2021

Topic of Lecture: Jackson System development

Introduction :

- Jackson System Development (JSD) is a method of system development that covers the software life cycle either directly or by providing a framework into which more specialized techniques can fit.
- JSD can start from the stage in a project when there is only a general statement of requirements.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Behavior of a system, Interaction and concurrency,Node

Detailed content of the Lecture:

Jackson System development:

- A JSD model describe the real world in terms of entities, actions, and ordering of actions.
- JSD is a method for specifying and designing systems.
- JSD is a structured systems analysis and design method.

There are three major phases and steps:

- The modeling phase.
- The Network phase.
- The implementation phase.

JSD Entity Structure:

Modeling Stage:

- Events and entities are identified.
- Entity structures and entity life cycle are formed.
- Result of this stage is a set of,
- Tables
- Definitions.
- Diagram.
- Entity / Action step.
- Entity Structure Step.

Network Phase:

- Input and output added to the derived model.
- Network of communication sequential processes.
- Represent interaction between process.
- Initial Model step.
- Function Step.
- System timing Step.

Implementation Phases:

- Detailed Coding and design.
- Results of this stage is the final system.

Video Content / Details of website for further learning (if any):

https://www.youtube.com/user/jacksonsystems

Important Books/Journals for further learning including the page nos.:
Software Engineering–Sommerville ,Addison Wesley-Longman, 2004 **(Page No : 223-228)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

MCA

I / II

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **II - SOFTWARE DESIGN**

Date of Lecture:13.03.2021

Topic of Lecture: Designing for Reuse

Introduction :

- Information technology, design reuse is the inclusion of previously designed components (blocks of logic or data) in software and hardware.
- Design reuse makes it faster and cheaper to design and build a new product, since the reused components will not only be already designed but also tested for reliability.

Prerequisite knowledge for Complete understanding and learning of Topic:

- System
- Modeling
- Behaviour
- Software Engineering

Detailed content of the Lecture:

Designing for Reuse:

- Developers can reuse a component in both similar and completely different applications: for example, a component used as part of a central processing unit (CPU) for a PC could be reused in a handheld device or a set-top box.
- In hardware development, components in design reuse are called IP cores (intellectual property cores).
- Many different software reuse is importance aspect is whether software is reused may be modified.

Approaches of software Reuse:

- Design pattern.
- Component-based development.
- Application Frameworks.
- Services oriented system.
- Application product lines.
- COTS Integration.
- Configurable Vertical Application.
- Program libraries.

Problems with COTS System Integration:

- Lack of control over functionally and performance.
- Problem with COTS system Interoperability.
- No control over system evolution.
- Support from COTS vendors.

The data reuse process is as follows:

1. **Gathering Information:** This involves the collection of information, processing and modeling to fetch related data.
2. **Information Reuse:** This involves the effective use of data.

The design reuse process has four major issues:

- Retrieve
- Reuse
- Repair
- Recover

- These are generally referred to as the four Rs. In spite of these challenges, companies have used the design reuse concept as a successfully implemented concept in the software field at different levels, ranging from low level code reuse to high level project reuse.

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=L3rdMsHLgIE

Important Books/Journals for further learning including the page nos.:
Software Engineering–Sommerville ,Addison Wesley-Longman, 2004 **(Page No : 278-281)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

I / II

MCA

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **II - SOFTWARE DESIGN**

Date of Lecture: 15.03.2021

Topic of Lecture: Programming standards , Case Study for Design of any Application Project.

Introduction :

- A coding standard gives a uniform appearance to the codes written by different engineers.
- It improves readability, and maintainability of the code and it reduces complexity also.
- It helps in code reuse and helps to detect error easily.

Prerequisite knowledge for Complete understanding and learning of Topic:

- System
- Modeling
- Behaviour
- Software Engineering

Detailed content of the Lecture:

Programming standards :

- Their programmers to maintain to some well-defined and standard style of coding called coding standards.
- They usually make their own coding standards and guidelines depending on what suits their organization best and based on the types of software they develop.
- It is very important for the programmers to maintain the coding standards otherwise the code will be rejected during code review.

Purpose of Having Coding Standards:

- A coding standard gives a uniform appearance to the codes written by different engineers.
- It improves readability, and maintainability of the code and it reduces complexity also.
- It helps in code reuse and helps to detect error easily.
- It promotes sound programming practices and increases efficiency of the programmers.

Case Study for Design of any Application Project:

- The case study of Genie allowed researchers to study whether language could be taught even after critical periods for language development had been missed. Her case also served as an example of how scientific research may interfere with treatment and lead to further abuse of vulnerable individuals.

- A case study is an in-depth study of one person, group, or event. In a case study, nearly every aspect of the subject's life and history is analyzed to seek patterns and causes of behavior.
- Case studies can be used in a variety of fields including psychology, medicine, education, anthropology, political science, and social work.

Benefits and Limitations:

- A case study is an in-depth study of one person, group, or event.
- In a case study, nearly every aspect of the subject's life and history is analyzed to seek patterns and causes of behavior.
- Case studies can be used in a variety of fields including psychology, medicine, education, anthropology, political science, and social work.

Types:

- **Collective case studies**: These involve studying a group of individuals. Researchers might study a group of people in a certain setting or look at an entire community of people.
- **Descriptive case studies**: These involve starting with a descriptive theory. The subjects are then observed and the information gathered is compared to the pre-existing theory.
- **Explanatory case studies**: These are often used to do causal investigations. In other words, researchers are interested in looking at factors that may have actually caused certain things to occur.
- **Exploratory case studies**: These are sometimes used as a prelude to further, more in-depth research. This allows researchers to gather more information before developing their research questions and hypotheses.
- **Instrumental case studies**: These occur when the individual or group allows researchers to understand more than what is initially obvious to observers.
- **Intrinsic case studies**: This type of case study is when the researcher has a personal interest in the case. Jean Piaget's observations of his own children are good examples of how an intrinsic cast study can contribute to the development of a psychological theory.

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=Tx-lh4yIM_k

Important Books/Journals for further learning including the page nos.:
Software Engineering–Sommerville ,Addison Wesley-Longman, 2004 **(Page No : 134-140)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **III - SOFTWARE TESTING AND MAINTENANCE**

Date of Lecture: 16.03.2021

Topic of Lecture: Software Testing Fundamentals

Introduction :

- SOFTWARE TESTING Fundamentals (STF) is a platform to gain (or refresh) basic knowledge in the field of Software Testing. If we are to 'cliche' it, the site.
- Software testing can be stated as the process of verifying and validating that software or application is bug-free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently with handling all the exceptional and boundary cases.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Object
- Relationship
- Packages
- Analysis

Detailed content of the Lecture:

Software Testing Fundamentals:

- Software testing is the process os testing the software product.
- Effective software testing will to the delivery of higher quality software products, more satisfied userd, lower maintenance costs, more accurate, and reliable results.
- Manual Testing.
- Automated Testing.

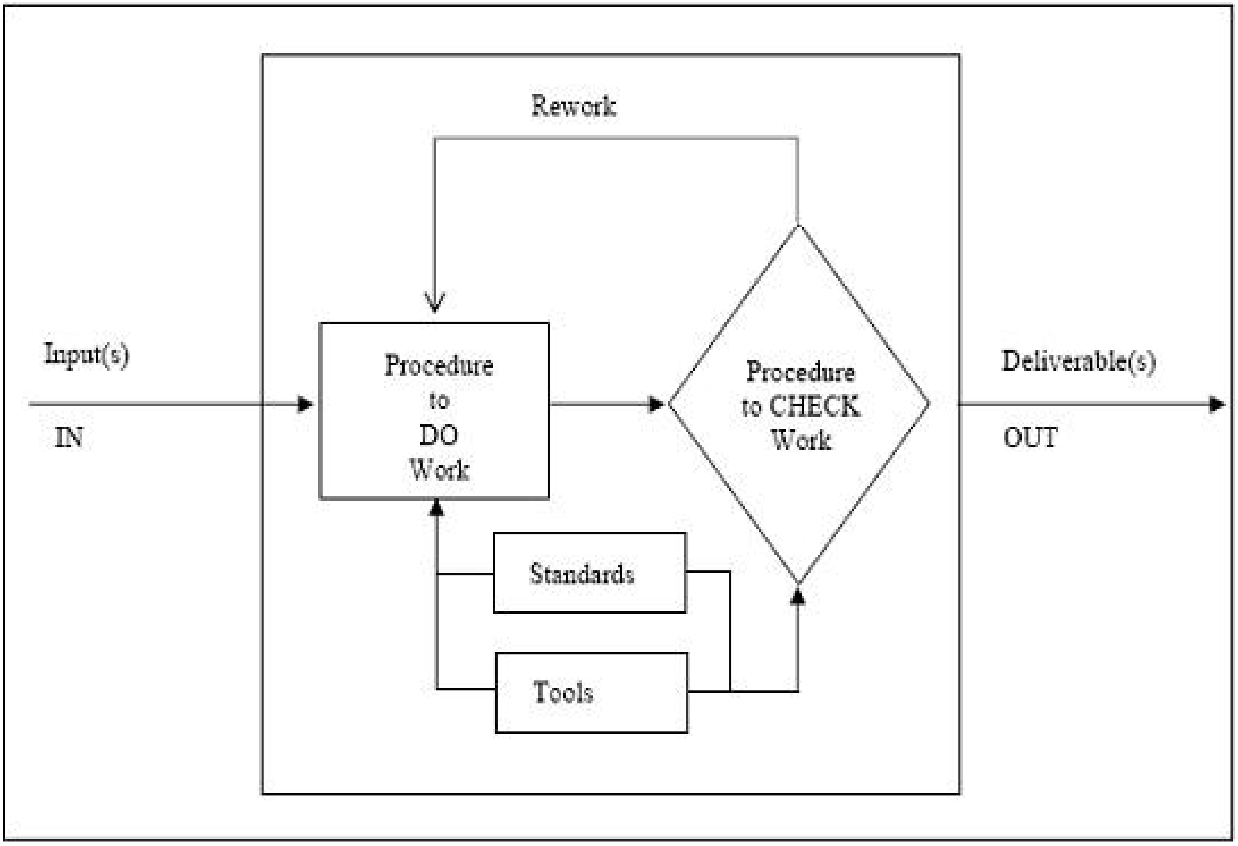
Terminologies:

- Error.
- Fault.
- Failure.

Verification and Validation(V&V) Testing:

- Verification means “Are we building the product Right”.
- Verification occurs of the end of every step of the life cycle.
- Validation means “Are we building the Right Product”.
- To matching with the customer requirements.
- It verifies that the software being developed implements all the requirements specified in the SRS document.

- Inputs.
- Outputs.
- Validation Process.
- Check Process.
- Standard, Tools, and Guidelines.



Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=goaZTAzsLMk

Important Books/Journals for further learning including the page nos.:
Software Engineering–Jibitesh Mishra, Ashok Mohanty ,Pearson Education, First Edition, 2012 **(Page No : 134-139)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

I / II

MCA

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs.R.Pavithra**

Unit :**III - SOFTWARE TESTING AND MAINTENANCE**

Date of Lecture: 17.03.2021

Topic of Lecture: Software testing strategies , Black Box Testing

Introduction :

- Software Testing is a type of investigation to find out if there is any default or error present in the software so that the errors can be reduced or removed to increase the quality of the software
- Black box testing checks scenarios where the system can break.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Use case, Process, Links, Methods, System

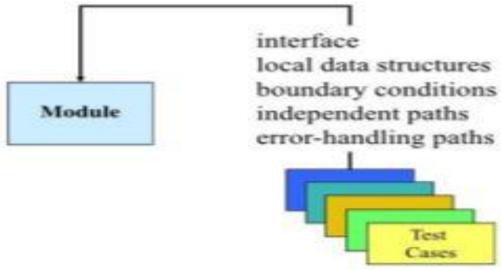
Detailed content of the Lecture:

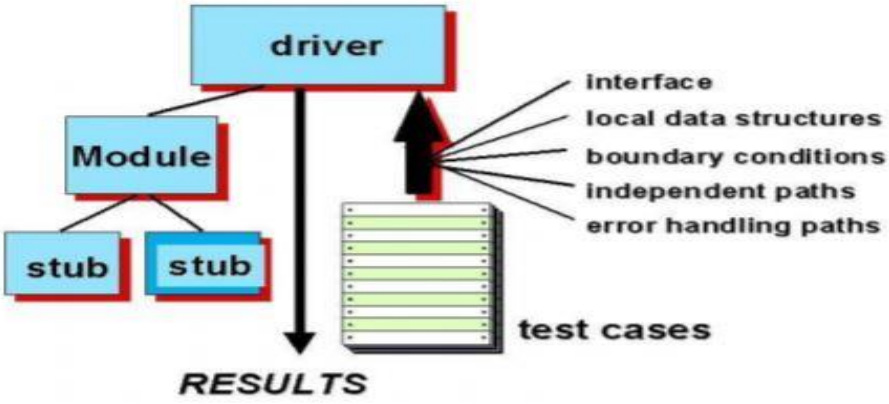
Software testing strategies:

- System Testing.
- Validation Testing.
- Integration Testing.
- Unit Testing.

Unit Testing:

- Unit testing is begin at the vertex of the spiral and concentrates each unit of the software as implemented in software code.





Integration Testing:

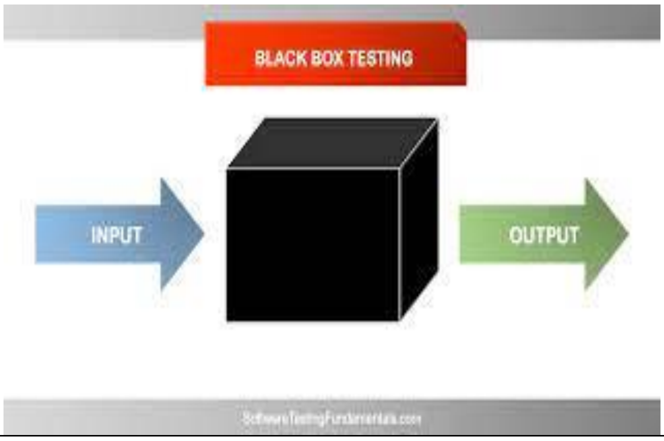
- The focus is on design and construction of the software architecture.

Types of Integration Testing:

- Big-Bang Integration Testing.
- Top-Down Integration Testing.
- Bottom-Up Integration Testing.
- Mixed Integration Testing.
- Regression Integration Testing.
- Somke Integration Testing.

Black-box testing:

- Testing of all the interfaces and should involve the clients in the process.
- Black box testing is also called behavioral testing.
- Specification testing.
- Behavioral testing.
- Data-driven testing.
- Functional testing.
- Input / Output driven testing.



Types of Functional testing:

- Positive functional testing.
- Negative functional testing.

Black-Box testing Techniques:

Equivalence Partitioning:

- It divides the input domain of a program into classes of data from which test casescn be derived.

Boundary Value Analysis:

- A great number of error tend to occur at the boundaries of the input domain rather than in the center.

Guidelines for performing Boundary value Analysis:

- Range.
- Maximum and minimum number.
- Output conditions.
- Internal data structures.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=9sVcvyCaCU8

Important Books/Journals for further learning including the page nos.:
Software Engineering–Jibitesh Mishra, Ashok Mohanty ,Pearson Education, First Edition, 2012 **(Page No : 67-72)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **III - SOFTWARE TESTING AND MAINTENANCE**

Date of Lecture: 18.03.2021

Topic of Lecture: White Box Testing , System Testing

Introduction :

- White Box Testing is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security.
- System testing is defined as testing of a complete and fully integrated software product.
- This testing falls in black-box testing wherein knowledge of the inner design of the code is not a pre-requisite and is done by the testing team.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Object
- Relationship
- Packages
- Analysis

Detailed content of the Lecture:

White Box Testing :

- White box testing, also known as clear box testing, happens when you have insight into the code and/or general knowledge about the architecture of the software.
- The structural testing test case are generate based on the actual code of the program or module to be tested.
- This structural approach is sometimes called “glass box testing”.
- Structural testing is concerned with testing the implementation of the program.
- Glass-box testing, Structural testing, Clear box testing, Open-box testing, Logic-driven testing and Path oriented testing.

White box testing technique:

- Statement Coverage testing.
- Branch Coverage testing.
- Condition Coverage testing.
- Loop coverage testing.
- Path Coverage testing.
- Data flow testing.

System Testing:

- System testing is a series of different tests whose primary purpose is to fully exercised the computer based system.

Types of system testing:

- Security Testing.
- Reliability Testing.
- Performance testing.
- Recovery testing.
- Thread testing.

Security Testing:

- Security testing involves testing the system in order to make sure that unauthorized personnel or other system.
- Program that check for access to the system in password are tested along eith any organizational security .

Reliability Testing:

- Software reliability refers to the probability of failure free operation of a system.
- Directly estimating software reliability by quantifying its related factors can be difficult.

Performance Testing:

- Performance testing is designed to test the run-time performance of software within the context of an integrated system.

Recovery Testing:

- Recovery testing uses test case esigned to examine how easily and completely the system can recover from a disaster.

Thread Testing:

- Thread testing is a popular testing techniques suitable for testing real-time system.
- It test the system response to events as processing threads through the system.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=3bJcvBLJViQ

Important Books/Journals for further learning including the page nos.:
Software Engineering–Jibitesh Mishra, Ashok Mohanty ,Pearson Education, First Edition, 2012
(Page No :67 -72)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **III - SOFTWARE TESTING AND MAINTENANCE**

Date of Lecture:19.03.2021

Topic of Lecture: Object Orientation Testing , State based Testing

Introduction :

- The object-oriented model, interaction errors can be uncovered by scenario-based testing.
- This form of Object oriented-testing can only test against the client's specifications, so interface errors are still missed.
- State-based testing is a new method for testing object-oriented programs.
- State-based testing validates the expected transformations that can occur within a class.
- Classes are modelled using physical values assigned to the attributes of the class.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Object
- Relationship
- Packages
- Analysis

Detailed content of the Lecture:

<u>Object Orientation Testing:</u>

- Testing is the process of examining something with the intention of finding errors.
- Testing may reveal a symptom of error, but it may not uncover .

<u>Factors effecting Object Oriented Testing:</u>

- Information Hiding.
- Encapsulation.
- Inheritance.

<u>Partition Testing at the Class Level:</u>

- Partition testing reduces the number of test cases required to exercise the class in much the same manner as equivalence partitioning for conventional software. Input and output are categorized and test cases are designed to exercise each category

- **<u>State-based partitioning:</u>** categorizes class operations based on their ability to change the state of the class. Again considering the account class, state operations include deposit and withdraw, whereas non state operations include balance, summarize, and credit Limit.

- **Attribute-based partitioning:** categorizes class operations based on the attributes that they use. For the account class, the attributes balance and creditLimit can be used to define partitions.
- **Category-based partitioning:** categorizes class operations based on the generic function that each performs.

State based Testing :

- State Transition Testing is a type of software testing which is performed to check the change in the state of the application under varying input.
- In this type of testing, both positive and negative input values are provided and the behavior of the system is observed.

Testing Tools:

- Software testing tools are develop quality software.

Types of Testing Tools:

- Tools for test management and control.
- Test specification Tools.
- Static Analysis Tools.
- Dynamic Testing Tools.

Tools for test management and control:

- Test management tools are used to store information on how testing is to be done, plan testing activities and report the status of quality assurance activities.

Test specification Tools:

- Test specifications are iterative, generative blueprints of test design.
- They are written at the item level, and allow test developers or item writers to produce new versions of a test for different test-taking populations.

Static Analysis Tools:

- Static analysis is used in software engineering by software development and quality assurance teams. Automated tools can assist programmers and developers in carrying out static analysis.

Dynamic Testing Tools:

- Dynamic Testing is a kind of software testing technique using which the dynamic behaviour of the code is analysed.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=tyALVEAUgc0

Important Books/Journals for further learning including the page nos.:
Software Engineering–Jibitesh Mishra, Ashok Mohanty ,Pearson Education, First Edition, 2012 **(Page No :123 -126)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **III - SOFTWARE TESTING AND MAINTENANCE**

Date of Lecture:20.03.2021

Topic of Lecture : Testing Tools

Introduction :

- Software Testing tools are the tools which are used for the testing of software. Software testing tools are often used to assure firmness, thoroughness and performance in testing software products.
- These tools are used to fulfill all the requirements of planned testing activities.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Class
- Object Oriented Concepts
- Link
- Interaction

Detailed content of the Lecture:

Testing Tools:

- Testing Improves Accuracy.
- Automation Does What Manual Testing Cannot.
- Automated Testing Helps Developers and Testers.
- QA and Dev Team Morale Improves.

Types of Testing Tools:

Test management tool:

- Test management tools are used to keep track of all the testing activity, fast data analysis, manage manual and automation test cases, various environments, and plan and maintain manual testing as well.

Performance testing tool:

- Performance or Load testing tools are used to check the load, stability, and scalability of the application.

- When n-number of the users using the application at the same time, and if the application gets crashed because of the immense load, to get through this type of issue, we need load testing tools.
- Volume Testing.
- Load Testing.
- Stress Testing.
- Network Testing.

Security testing tool:

- The security testing tool is used to ensure the security of the software and check for the security leakage.
- If any security loophole is there, it could be fixed at the early stage of the product.
- We need this type of the tool when the software has encoded the security code which is not accessible by the unauthorized users.
- Intrusion Testing.
- Hardening Testing.

Test Specification Tools:

- Test specifications are iterative, generative blueprints of test design.
- They are written at the item level, and allow test developers or item writers to produce new versions of a test for different test-taking populations.

Dynamic Testing Tools :

- Dynamic analysis is the process of testing and evaluating a program while software is running.
- Also referred to as dynamic code scanning, dynamic analysis improves the diagnosis and correction of bugs, memory issues, and crashes of an application during its execution.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=CqPq7EAw1o4

Important Books/Journals for further learning including the page nos.:
Software Engineering–Jibitesh Mishra, Ashok Mohanty ,Pearson Education, First Edition, 2012 **(Page No :143 -145)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code : **SOFTWARE ENGINEERING /19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **III - SOFTWARE TESTING AND MAINTENANCE**

Date of Lecture:22.03.2021

Topic of Lecture : Test Case Management

Introduction :

- Test case management tools have great data entry, tracking, and automation capabilities, but they also give you really powerful reporting tools as well. You can easily see all sorts of data, trends, and graphs related to the ongoing testing status of the software.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Attributes
- Methods
- Relationships
- Class

Detailed content of the Lecture:

Test Case Management :

- Test management tools allow teams to manage test case environments, automated tests, defects and project tasks.
- Some applications include advanced dashboards and detailed tracking of key metrics, allowing for easy tracking of progress and bug management.
- Test management is basically everything that testers and QA teams do to manage the software testing process or life cycle.
- Test case management tools enable software testers and Quality Assurance (QA) teams to manage test case environments, automated tests, bugs and project tasks.

Test Management Tools:

QMetry Test Management:

- QMetry Test Management by QMetry is an enterprise test management tool that helps companies achieve their goal of shifting left and achieving continuous quality.

Consistently ranked as one of the top tools for test case management, QMetry offers the following

features:

- Test case authoring
- Reusable test cases
- Test execution
- Test Case coverage reports with customization options
- Seamless integrations with Jira, CI/CD tools like Bamboo or Jenkins and automation frameworks
- Exploratory testing for smarter and automated test case documentation
- Comprehensive test coverage and full traceability
- Easy migration and import/export capability
- Easy cloning, auto fill, and auto suggests of test runs

T-Plan Professional:

- A test plan is a detailed document that outlines the test strategy, objectives, resources needed, schedule, and success criteria for testing a specific new feature or piece of software.
- The main goal, of course, is to discover defects, errors, and any other gap that might cause the software to not act as intended or provide a bad experience for your users. More specifically, a test plan ensures your software:
- Meets the requirements that guided its design and development (In other words, does it do what it's supposed to do when it's supposed to do it?)
- Responds correctly to all kinds of inputs

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=MBQoGvGDGqU

Important Books/Journals for further learning including the page nos.:
Software Engineering–Jibitesh Mishra, Ashok Mohanty ,Pearson Education, First Edition, 2012 **(Page No :172 -176)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

DESIGNING YOUR FUTURE
Estd. 2000

IQAC

LECTURE HANDOUTS

I / II

MCA

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **III - SOFTWARE TESTING AND MAINTENANCE**

Date of Lecture: 23.03.2021

Topic of Lecture: Software Maintenance Organization

Introduction :

- Software maintenance is the process of changing, modifying, and updating software to keep up with customer needs. Software maintenance is done after the product has launched for several reasons including improving the software overall, correcting issues or bugs, to boost performance.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Object
- Classes
- Methods
- Response

Detailed content of the Lecture:

<u>Software Maintenance Organization:</u>

- Software maintenance is a vast activity which includes optimization, error correction, deletion of discarded features and enhancement of existing features.
- Since these changes are necessary, a mechanism must be created for estimation, controlling and making modifications.
- Software maintenance is a natural part of SDLC (software development life cycle). Software developers don't have the luxury of launching a product and letting it run, they constantly need to be on the lookout to both correct and improve their software to remain competitive and relevant.
- Using the right software maintenance techniques and strategies is a critical part of keeping any software running for a long period of time and keeping customers and users happy.

Why is software maintenance important?

- Creating a new piece of software and launching it into the world is an exciting step for any company. A lot goes into creating your software and its launch including the actual building and coding, licensing models, marketing, and more.

- This means monitoring and maintaining properly.
- As technology is changing at the speed of light, software must keep up with the market changes and demands.

Software as Evolutionary Entity:

- Software Evolution is a term which refers to the process of developing software initially, then timely updating it for various reasons, i.e., to add new features or to remove obsolete functionalities etc.
- The evolution process includes fundamental activities of change analysis, release planning, system implementation and releasing a system to customers.

Law of continuing change:

- This law states that any software system that represents some real-world reality undergoes continuous change or become progressively less useful in that environment.

1. **Law of increasing complexity:**
   As an evolving program changes, its structure becomes more complex unless effective efforts are made to avoid this phenomenon.
2. **Law of conservation of organization stability:**
   Over the lifetime of a program, the rate of development of that program is approximately constant and independent of the resource devoted to system development.
3. **Law of conservation of familiarity:**
   This law states that during the active lifetime of the program, changes made in the successive release are almost constant.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=ZsSPBsl1Ld8

Important Books/Journals for further learning including the page nos.:
Software Engineering–Jibitesh Mishra, Ashok Mohanty ,Pearson Education, First Edition, 2012
(Page No :193 -196)

Course Faculty

Verified by HOD

LECTURE HANDOUTS

I / II

MCA

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **III - SOFTWARE TESTING AND MAINTENANCE**

Date of Lecture: 24.03.2021

Topic of Lecture: Maintenance Report

Introduction :

- A maintenance report is a document that contains specific information about your past maintenance actions and their effect on cost, assets, and business performance.
- In general, maintenance reports are used to track KPIs and performance indicators which the department identified as worthy

Prerequisite knowledge for Complete understanding and learning of Topic:

- Generalization
- Class Hierarchy
- Relationship
- Link

Detailed content of the Lecture:

Maintenance Report:

- A maintenance report is a document that contains specific information about your past maintenance actions and their effect on cost, assets, and business performance. In general, maintenance reports are used to track.

1. Maintenance is applicable to software developed using any software life cycle model.
   - The system changes and hence maintenance must be performed in order to:
   - Correct Faults.
   - Improve the design.
   - Implement enhancement.
   - Interface with other system.
   - Adoption of environment.
   - Migrate legacy software.
   - Replacement of old software by new software.
2. In software maintenance report four key characteristics should be mentioned:

- Maintaining control over the software day to day functions.
- Maintaining control over software modification.
- Repairing of function.
- Performance degradation should be avoided.

Maintenance report:

- A maintenance report is a document that contains specific information about your past maintenance actions and their effect on cost, assets, and business performance.

1. **Concise**: the report should be brief but comprehensive; presented in a way that makes complex data easy to digest.
2. **Fact-based**: accurate; up-to-date; objective, based on hard data rather than opinions and speculations.

Reviewed in the proper context: was downtime prolonged because a supplier delayed the shipment of an important spare part or because we didn't order it in time – context is crucial for decision making.

- **Preventive maintenance report:** includes an overview of preventive maintenance activities and maintenance schedules (number of PM tasks open, scheduled, closed, and deferred, planned maintenance percentage..). It can be broken down by location, equipment type, vendor, etc.
- **Work order report:** an overview of work order management activities – received maintenance tickets and created, open, closed, and deferred WOs.
- **Asset history report:** it can include the list of purchase orders, PMs, WOs, spare parts, and other resources spent on a specific asset or group of assets.
- **Purchase order report:** list of created, sent, scheduled, received, and closed purchase orders; can be broken down by vendor, facility, priority, order type, order status, and more.
- **Vendor history report:** an overview of the work performed by a maintenance vendor and associated costs.

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=4yEOCUHoZ54

Important Books/Journals for further learning including the page nos.:
Software Engineering–Jibitesh Mishra, Ashok Mohanty ,Pearson Education, First Edition, 2012 **(Page No :203 -206)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code	**: SOFTWARE ENGINEERING / 19CAB10**
Course Faculty	**: Mrs. R.Pavithra**
Unit	**: III - SOFTWARE TESTING AND MAINTENANCE**

Date of Lecture: 25.03.2021

Topic of Lecture: Types of Maintenance

Introduction :

- Software maintenance is the process of changing, modifying, and updating software to keep up with customer needs.
- Software maintenance is done after the product has launched for several reasons including improving the software overall, correcting issues or bugs, to boost performance

Prerequisite knowledge for Complete understanding and learning of Topic:

- Standards, Quality, Processes, Tool, Services

Detailed content of the Lecture:

Types of Maintenance:

- Software maintenance is a natural part of SDLC (software development life cycle).
- Software developers don't have the luxury of launching a product and letting it run, they constantly need to be on the lookout to both correct and improve their software to remain competitive and relevant.

- Corrective Software Maintenance.
- Adaptive Software Maintenance.
- Perceptive Software Maintenance.
- Preventive Software Maintenance.

Corrective Software Maintenance :

- Corrective software maintenance is the typical, classic form of maintenance (for software and anything else for that matter).
- Corrective software maintenance is necessary when something goes wrong in a piece of software including faults and errors.
- These can have a widespread impact on the functionality of the software in general and therefore must be addressed as quickly as possible.

- Many times, software vendors can address issues that require corrective maintenance due to bug reports that users send in.
- If a company can recognize and take care of faults before users discover them, this is an added advantage that will make your company seem more reputable and reliable (no one likes an error message after all).

Adaptive Software Maintenance :

- Adaptive software maintenance has to do with the changing technologies as well as policies and rules regarding your software.
- These include operating system changes, cloud storage, hardware, etc.
- When these changes are performed, your software must adapt in order to properly meet new requirements and continue to run well.

Perceptive Software Maintenance:

- As with any product on the market, once the software is released to the public, new issues and ideas come to the surface.
- Users may see the need for new features or requirements that they would like to see in the software to make it the best tool available for their needs.
- This is when Perceptive software maintenance comes into play.

Preventive Software Maintenance:

- Preventative software maintenance is looking into the future so that your software can keep working as desired for as long as possible.
- This includes making necessary changes, upgrades, adaptations and more.
- Preventative software maintenance may address small issues which at the given time may lack significance but may turn into larger problems in the future

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=nA4ev7jM6ek

Important Books/Journals for further learning including the page nos.:
Software Engineering–Jibitesh Mishra, Ashok Mohanty ,Pearson Education, First Edition, 2012
(Page No :214 -217)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

DESIGNING YOUR FUTURE
Estd. 2000

IQAC

LECTURE HANDOUTS

MCA

I / II

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **III - SOFTWARE TESTING AND MAINTENANCE**

Date of Lecture: 16.03.2021

Topic of Lecture: Software Testing Fundamentals

Introduction :

- SOFTWARE TESTING Fundamentals (STF) is a platform to gain (or refresh) basic knowledge in the field of Software Testing. If we are to 'cliche' it, the site.
- Software testing can be stated as the process of verifying and validating that software or application is bug-free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently with handling all the exceptional and boundary cases.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Object
- Relationship
- Packages
- Analysis

Detailed content of the Lecture:

<u>Software Testing Fundamentals:</u>

- Software testing is the process os testing the software product.
- Effective software testing will to the delivery of higher quality software products, more satisfied userd, lower maintenance costs, more accurate, and reliable results.
- Manual Testing.
- Automated Testing.

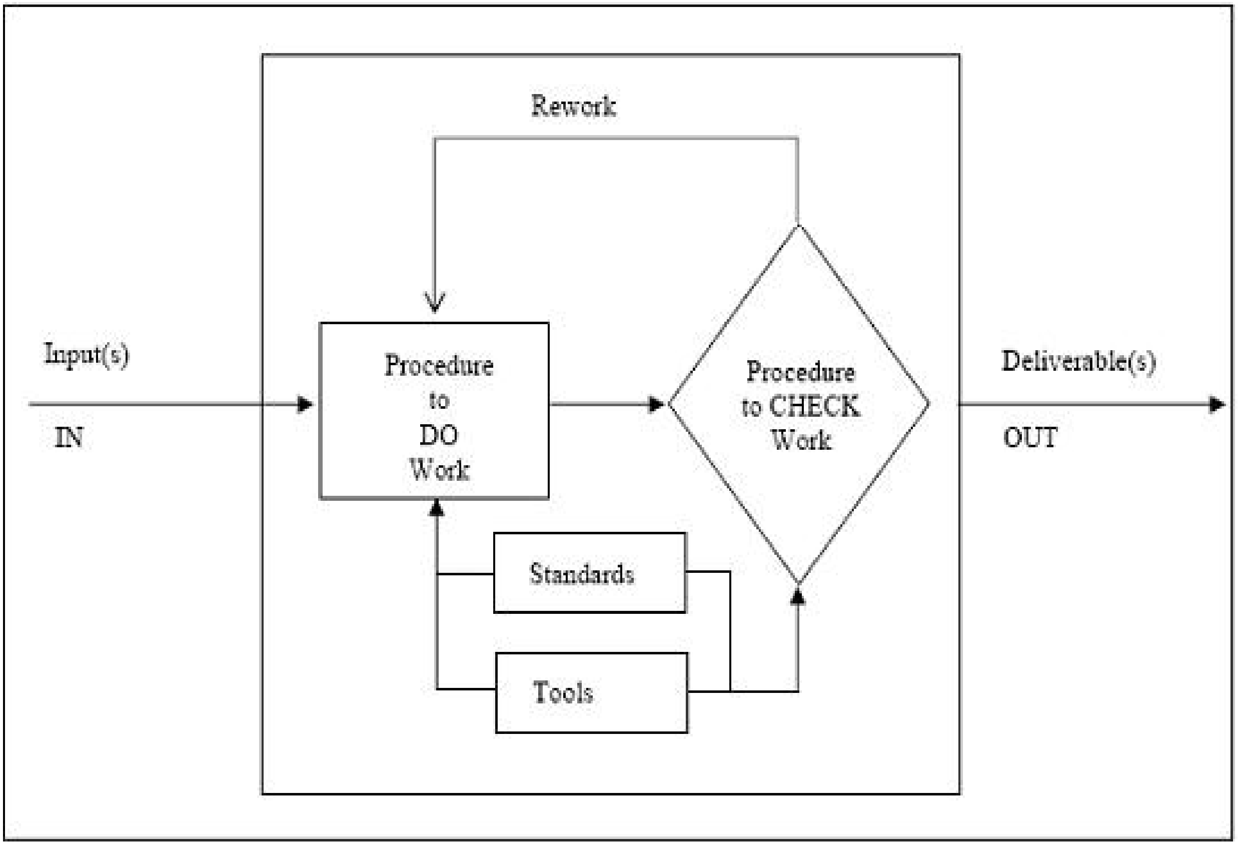
<u>Terminologies:</u>

- Error.
- Fault.
- Failure.

<u>Verification and Validation(V&V) Testing:</u>

- Verification means “Are we building the product Right”.
- Verification occurs of the end of every step of the life cycle.
- Validation means “Are we building the Right Product”.
- To matching with the customer requirements.
- It verifies that the software being developed implements all the requirements specified in the SRS document.

- Inputs.
- Outputs.
- Validation Process.
- Check Process.
- Standard, Tools, and Guidelines.



Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=goaZTAzsLMk

Important Books/Journals for further learning including the page nos.:
Software Engineering–Jibitesh Mishra, Ashok Mohanty ,Pearson Education, First Edition, 2012
(Page No : 134-139)

Course Faculty

Verified by HOD

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs.R.Pavithra**

Unit :**III - SOFTWARE TESTING AND MAINTENANCE**

Date of Lecture: 17.03.2021

Topic of Lecture: Software testing strategies , Black Box Testing

Introduction :

- Software Testing is a type of investigation to find out if there is any default or error present in the software so that the errors can be reduced or removed to increase the quality of the software
- Black box testing checks scenarios where the system can break.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Use case, Process, Links, Methods, System

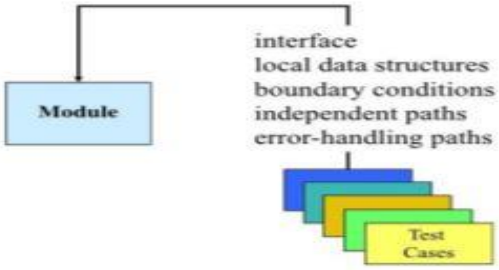
Detailed content of the Lecture:

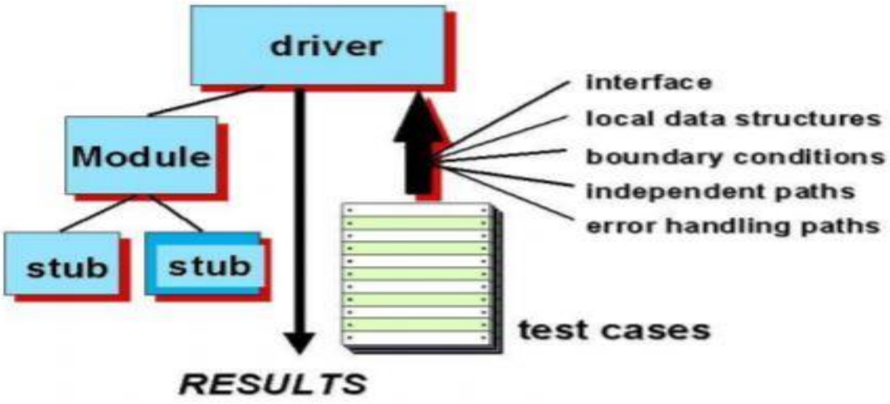
Software testing strategies:

- System Testing.
- Validation Testing.
- Integration Testing.
- Unit Testing.

Unit Testing:

- Unit testing is begin at the vertex of the spiral and concentrates each unit of the software as implemented in software code.





Integration Testing:

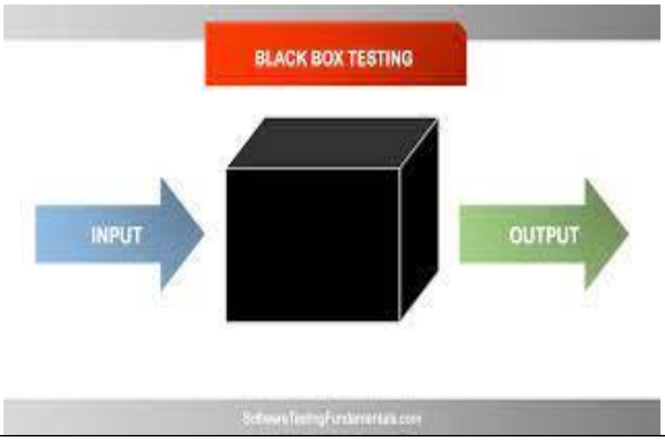
- The focus is on design and construction of the software architecture.

Types of Integration Testing:

- Big-Bang Integration Testing.
- Top-Down Integration Testing.
- Bottom-Up Integration Testing.
- Mixed Integration Testing.
- Regression Integration Testing.
- Somke Integration Testing.

Black-box testing:

- Testing of all the interfaces and should involve the clients in the process.
- Black box testing is also called behavioral testing.
- Specification testing.
- Behavioral testing.
- Data-driven testing.
- Functional testing.
- Input / Output driven testing.



Types of Functional testing:

- Positive functional testing.
- Negative functional testing.

Black-Box testing Techniques:

Equivalence Partitioning:

- It divides the input domain of a program into classes of data from which test casescn be derived.

Boundary Value Analysis:

- A great number of error tend to occur at the boundaries of the input domain rather than in the center.

Guidelines for performing Boundary value Analysis:

- Range.
- Maximum and minimum number.
- Output conditions.
- Internal data structures.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=9sVcvyCaCU8

Important Books/Journals for further learning including the page nos.:
Software Engineering–Jibitesh Mishra, Ashok Mohanty ,Pearson Education, First Edition, 2012 **(Page No : 67-72)**

Course Faculty

Verified by HOD

Estd. 2000

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **III - SOFTWARE TESTING AND MAINTENANCE**

Date of Lecture: 18.03.2021

Topic of Lecture: White Box Testing , System Testing

Introduction :

- White Box Testing is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security.
- System testing is defined as testing of a complete and fully integrated software product.
- This testing falls in black-box testing wherein knowledge of the inner design of the code is not a pre-requisite and is done by the testing team.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Object
- Relationship
- Packages
- Analysis

Detailed content of the Lecture:

<u>White Box Testing :</u>

- White box testing, also known as clear box testing, happens when you have insight into the code and/or general knowledge about the architecture of the software.
- The structural testing test case are generate based on the actual code of the program or module to be tested.
- This structural approach is sometimes called “glass box testing”.
- Structural testing is concerned with testing the implementation of the program.
- Glass-box testing, Structural testing, Clear box testing, Open-box testing, Logic-driven testing and Path oriented testing.

<u>White box testing technique:</u>

- Statement Coverage testing.
- Branch Coverage testing.
- Condition Coverage testing.
- Loop coverage testing.
- Path Coverage testing.
- Data flow testing.

System Testing:

- System testing is a series of different tests whose primary purpose is to fully exercised the computer based system.

Types of system testing:

- Security Testing.
- Reliability Testing.
- Performance testing.
- Recovery testing.
- Thread testing.

Security Testing:

- Security testing involves testing the system in order to make sure that unauthorized personnel or other system.
- Program that check for access to the system in password are tested along eith any organizational security .

Reliability Testing:

- Software reliability refers to the probability of failure free operation of a system.
- Directly estimating software reliability by quantifying its related factors can be difficult.

Performance Testing:

- Performance testing is designed to test the run-time performance of software within the context of an integrated system.

Recovery Testing:

- Recovery testing uses test case esigned to examine how easily and completely the system can recover from a disaster.

Thread Testing:

- Thread testing is a popular testing techniques suitable for testing real-time system.
- It test the system response to events as processing threads through the system.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=3bJcvBLJViQ

Important Books/Journals for further learning including the page nos.:
Software Engineering–Jibitesh Mishra, Ashok Mohanty ,Pearson Education, First Edition, 2012
(Page No :67 -72)

Course Faculty

Verified by HOD

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **III - SOFTWARE TESTING AND MAINTENANCE**

Date of Lecture:19.03.2021

Topic of Lecture: Object Orientation Testing , State based Testing

Introduction :

- The object-oriented model, interaction errors can be uncovered by scenario-based testing.
- This form of Object oriented-testing can only test against the client's specifications, so interface errors are still missed.
- State-based testing is a new method for testing object-oriented programs.
- State-based testing validates the expected transformations that can occur within a class.
- Classes are modelled using physical values assigned to the attributes of the class.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Object
- Relationship
- Packages
- Analysis

Detailed content of the Lecture:

Object Orientation Testing:

- Testing is the process of examining something with the intention of finding errors.
- Testing may reveal a symptom of error, but it may not uncover .

Factors effecting Object Oriented Testing:

- Information Hiding.
- Encapsulation.
- Inheritance.

Partition Testing at the Class Level:

- Partition testing reduces the number of test cases required to exercise the class in much the same manner as equivalence partitioning for conventional software. Input and output are categorized and test cases are designed to exercise each category

- **State-based partitioning:** categorizes class operations based on their ability to change the state of the class. Again considering the account class, state operations include deposit and withdraw, whereas non state operations include balance, summarize, and credit Limit.

- **Attribute-based partitioning:** categorizes class operations based on the attributes that they use. For the account class, the attributes balance and creditLimit can be used to define partitions.
- **Category-based partitioning:** categorizes class operations based on the generic function that each performs.

State based Testing :

- State Transition Testing is a type of software testing which is performed to check the change in the state of the application under varying input.
- In this type of testing, both positive and negative input values are provided and the behavior of the system is observed.

Testing Tools:

- Software testing tools are develop quality software.

Types of Testing Tools:

- Tools for test management and control.
- Test specification Tools.
- Static Analysis Tools.
- Dynamic Testing Tools.

Tools for test management and control:

- Test management tools are used to store information on how testing is to be done, plan testing activities and report the status of quality assurance activities.

Test specification Tools:

- Test specifications are iterative, generative blueprints of test design.
- They are written at the item level, and allow test developers or item writers to produce new versions of a test for different test-taking populations.

Static Analysis Tools:

- Static analysis is used in software engineering by software development and quality assurance teams. Automated tools can assist programmers and developers in carrying out static analysis.

Dynamic Testing Tools:

- Dynamic Testing is a kind of software testing technique using which the dynamic behaviour of the code is analysed.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=tyALVEAUgc0

Important Books/Journals for further learning including the page nos.:
Software Engineering–Jibitesh Mishra, Ashok Mohanty ,Pearson Education, First Edition, 2012 **(Page No :123 -126)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

MCA

I / II

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **III - SOFTWARE TESTING AND MAINTENANCE**

Date of Lecture:20.03.2021

Topic of Lecture : Testing Tools

Introduction :

- Software Testing tools are the tools which are used for the testing of software. Software testing tools are often used to assure firmness, thoroughness and performance in testing software products.
- These tools are used to fulfill all the requirements of planned testing activities.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Class
- Object Oriented Concepts
- Link
- Interaction

Detailed content of the Lecture:

Testing Tools:

- Testing Improves Accuracy.
- Automation Does What Manual Testing Cannot.
- Automated Testing Helps Developers and Testers.
- QA and Dev Team Morale Improves.

Types of Testing Tools:

Test management tool:

- Test management tools are used to keep track of all the testing activity, fast data analysis, manage manual and automation test cases, various environments, and plan and maintain manual testing as well.

Performance testing tool:

- Performance or Load testing tools are used to check the load, stability, and scalability of the application.

- When n-number of the users using the application at the same time, and if the application gets crashed because of the immense load, to get through this type of issue, we need load testing tools.
- Volume Testing.
- Load Testing.
- Stress Testing.
- Network Testing.

Security testing tool:

- The security testing tool is used to ensure the security of the software and check for the security leakage.
- If any security loophole is there, it could be fixed at the early stage of the product.
- We need this type of the tool when the software has encoded the security code which is not accessible by the unauthorized users.
- Intrusion Testing.
- Hardening Testing.

Test Specification Tools:

- Test specifications are iterative, generative blueprints of test design.
- They are written at the item level, and allow test developers or item writers to produce new versions of a test for different test-taking populations.

Dynamic Testing Tools :

- Dynamic analysis is the process of testing and evaluating a program while software is running.
- Also referred to as dynamic code scanning, dynamic analysis improves the diagnosis and correction of bugs, memory issues, and crashes of an application during its execution.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=CqPq7EAw1o4

Important Books/Journals for further learning including the page nos.:
Software Engineering–Jibitesh Mishra, Ashok Mohanty ,Pearson Education, First Edition, 2012 **(Page No :143 -145)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

MCA

I / II

Course Name with Code : **SOFTWARE ENGINEERING /19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **III - SOFTWARE TESTING AND MAINTENANCE**

Date of Lecture:22.03.2021

Topic of Lecture : Test Case Management

Introduction :

- Test case management tools have great data entry, tracking, and automation capabilities, but they also give you really powerful reporting tools as well. You can easily see all sorts of data, trends, and graphs related to the ongoing testing status of the software.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Attributes
- Methods
- Relationships
- Class

Detailed content of the Lecture:

Test Case Management :

- Test management tools allow teams to manage test case environments, automated tests, defects and project tasks.
- Some applications include advanced dashboards and detailed tracking of key metrics, allowing for easy tracking of progress and bug management.
- Test management is basically everything that testers and QA teams do to manage the software testing process or life cycle.
- Test case management tools enable software testers and Quality Assurance (QA) teams to manage test case environments, automated tests, bugs and project tasks.

Test Management Tools:

QMetry Test Management:

- QMetry Test Management by QMetry is an enterprise test management tool that helps companies achieve their goal of shifting left and achieving continuous quality.

Consistently ranked as one of the top tools for test case management, QMetry offers the following

features:

- Test case authoring
- Reusable test cases
- Test execution
- Test Case coverage reports with customization options
- Seamless integrations with Jira, CI/CD tools like Bamboo or Jenkins and automation frameworks
- Exploratory testing for smarter and automated test case documentation
- Comprehensive test coverage and full traceability
- Easy migration and import/export capability
- Easy cloning, auto fill, and auto suggests of test runs

T-Plan Professional:

- A test plan is a detailed document that outlines the test strategy, objectives, resources needed, schedule, and success criteria for testing a specific new feature or piece of software.
- The main goal, of course, is to discover defects, errors, and any other gap that might cause the software to not act as intended or provide a bad experience for your users. More specifically, a test plan ensures your software:
- Meets the requirements that guided its design and development (In other words, does it do what it's supposed to do when it's supposed to do it?)
- Responds correctly to all kinds of inputs

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=MBQoGvGDGqU

Important Books/Journals for further learning including the page nos.:
Software Engineering–Jibitesh Mishra, Ashok Mohanty ,Pearson Education, First Edition, 2012 **(Page No :172 -176)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

I / II

MCA

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **III - SOFTWARE TESTING AND MAINTENANCE**

Date of Lecture: 23.03.2021

Topic of Lecture: Software Maintenance Organization

Introduction :

- Software maintenance is the process of changing, modifying, and updating software to keep up with customer needs. Software maintenance is done after the product has launched for several reasons including improving the software overall, correcting issues or bugs, to boost performance.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Object
- Classes
- Methods
- Response

Detailed content of the Lecture:

<u>Software Maintenance Organization:</u>

- Software maintenance is a vast activity which includes optimization, error correction, deletion of discarded features and enhancement of existing features.
- Since these changes are necessary, a mechanism must be created for estimation, controlling and making modifications.
- Software maintenance is a natural part of SDLC (software development life cycle). Software developers don’t have the luxury of launching a product and letting it run, they constantly need to be on the lookout to both correct and improve their software to remain competitive and relevant.
- Using the right software maintenance techniques and strategies is a critical part of keeping any software running for a long period of time and keeping customers and users happy.

Why is software maintenance important?

- Creating a new piece of software and launching it into the world is an exciting step for any company. A lot goes into creating your software and its launch including the actual building and coding, licensing models, marketing, and more.

- This means monitoring and maintaining properly.
- As technology is changing at the speed of light, software must keep up with the market changes and demands.

Software as Evolutionary Entity:

- Software Evolution is a term which refers to the process of developing software initially, then timely updating it for various reasons, i.e., to add new features or to remove obsolete functionalities etc.
- The evolution process includes fundamental activities of change analysis, release planning, system implementation and releasing a system to customers.

Law of continuing change:

- This law states that any software system that represents some real-world reality undergoes continuous change or become progressively less useful in that environment.

4. **Law of increasing complexity:**
   As an evolving program changes, its structure becomes more complex unless effective efforts are made to avoid this phenomenon.
5. **Law of conservation of organization stability:**
   Over the lifetime of a program, the rate of development of that program is approximately constant and independent of the resource devoted to system development.
6. **Law of conservation of familiarity:**
   This law states that during the active lifetime of the program, changes made in the successive release are almost constant.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=ZsSPBsl1Ld8

Important Books/Journals for further learning including the page nos.:
Software Engineering–Jibitesh Mishra, Ashok Mohanty ,Pearson Education, First Edition, 2012
(Page No :193 -196)

Course Faculty

Verified by HOD

LECTURE HANDOUTS

I / II

MCA

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **III - SOFTWARE TESTING AND MAINTENANCE**

Date of Lecture: 24.03.2021

Topic of Lecture: Maintenance Report

Introduction :

- A maintenance report is a document that contains specific information about your past maintenance actions and their effect on cost, assets, and business performance.
- In general, maintenance reports are used to track KPIs and performance indicators which the department identified as worthy

Prerequisite knowledge for Complete understanding and learning of Topic:

- Generalization
- Class Hierarchy
- Relationship
- Link

Detailed content of the Lecture:

Maintenance Report:

- A maintenance report is a document that contains specific information about your past maintenance actions and their effect on cost, assets, and business performance. In general, maintenance reports are used to track.

3. Maintenance is applicable to software developed using any software life cycle model.

- The system changes and hence maintenance must be performed in order to:
- Correct Faults.
- Improve the design.
- Implement enhancement.
- Interface with other system.
- Adoption of environment.
- Migrate legacy software.
- Replacement of old software by new software.

4. In software maintenance report four key characteristics should be mentioned:

- Maintaining control over the software day to day functions.
- Maintaining control over software modification.
- Repairing of function.
- Performance degradation should be avoided.

Maintenance report:

- A maintenance report is a document that contains specific information about your past maintenance actions and their effect on cost, assets, and business performance.

3. **Concise**: the report should be brief but comprehensive; presented in a way that makes complex data easy to digest.
4. **Fact-based**: accurate; up-to-date; objective, based on hard data rather than opinions and speculations.

Reviewed in the proper context: was downtime prolonged because a supplier delayed the shipment of an important spare part or because we didn't order it in time – context is crucial for decision making.

- **Preventive maintenance report:** includes an overview of preventive maintenance activities and maintenance schedules (number of PM tasks open, scheduled, closed, and deferred, planned maintenance percentage..). It can be broken down by location, equipment type, vendor, etc.
- **Work order report:** an overview of work order management activities – received maintenance tickets and created, open, closed, and deferred WOs.
- **Asset history report:** it can include the list of purchase orders, PMs, WOs, spare parts, and other resources spent on a specific asset or group of assets.
- **Purchase order report:** list of created, sent, scheduled, received, and closed purchase orders; can be broken down by vendor, facility, priority, order type, order status, and more.
- **Vendor history report:** an overview of the work performed by a maintenance vendor and associated costs.

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=4yEOCUHoZ54

Important Books/Journals for further learning including the page nos.:
Software Engineering–Jibitesh Mishra, Ashok Mohanty ,Pearson Education, First Edition, 2012 **(Page No :203 -206)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **III - SOFTWARE TESTING AND MAINTENANCE**

Date of Lecture: 25.03.2021

Topic of Lecture: Types of Maintenance

Introduction :

- Software maintenance is the process of changing, modifying, and updating software to keep up with customer needs.
- Software maintenance is done after the product has launched for several reasons including improving the software overall, correcting issues or bugs, to boost performance

Prerequisite knowledge for Complete understanding and learning of Topic:

- Standards, Quality, Processes, Tool, Services

Detailed content of the Lecture:

Types of Maintenance:

- Software maintenance is a natural part of SDLC (software development life cycle).
- Software developers don't have the luxury of launching a product and letting it run, they constantly need to be on the lookout to both correct and improve their software to remain competitive and relevant.

- Corrective Software Maintenance.
- Adaptive Software Maintenance.
- Perceptive Software Maintenance.
- Preventive Software Maintenance.

Corrective Software Maintenance :

- Corrective software maintenance is the typical, classic form of maintenance (for software and anything else for that matter).
- Corrective software maintenance is necessary when something goes wrong in a piece of software including faults and errors.
- These can have a widespread impact on the functionality of the software in general and therefore must be addressed as quickly as possible.

- Many times, software vendors can address issues that require corrective maintenance due to bug reports that users send in.
- If a company can recognize and take care of faults before users discover them, this is an added advantage that will make your company seem more reputable and reliable (no one likes an error message after all).

Adaptive Software Maintenance :

- Adaptive software maintenance has to do with the changing technologies as well as policies and rules regarding your software.
- These include operating system changes, cloud storage, hardware, etc.
- When these changes are performed, your software must adapt in order to properly meet new requirements and continue to run well.

Perceptive Software Maintenance:

- As with any product on the market, once the software is released to the public, new issues and ideas come to the surface.
- Users may see the need for new features or requirements that they would like to see in the software to make it the best tool available for their needs.
- This is when Perceptive software maintenance comes into play.

Preventive Software Maintenance:

- Preventative software maintenance is looking into the future so that your software can keep working as desired for as long as possible.
- This includes making necessary changes, upgrades, adaptations and more.
- Preventative software maintenance may address small issues which at the given time may lack significance but may turn into larger problems in the future

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=nA4ev7jM6ek

Important Books/Journals for further learning including the page nos.:
Software Engineering–Jibitesh Mishra, Ashok Mohanty ,Pearson Education, First Edition, 2012
(Page No :214 -217)

Course Faculty

Verified by HOD

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **IV -SOFTWARE METRICS**

Date of Lecture: 26.03.2021

Topic of Lecture: Scope

Introduction :

- A measurement is an manifestation of the size, quantity, amount or dimension of a particular attributes of a product or process.
- Software measurement is a titrate impute of a characteristic of a software product or the software process.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Design
- Model
- Relationship
- Rules
- Object

Detailed content of the Lecture:

Scope :

- Software measurement is a quantified attribute (see also: measurement) of a characteristic of a software product or the software process. It is a discipline within software engineering.
- The process of software measurement is defined and governed by ISO Standard ISO.
- Make General Predictions about a system.
- Identify Anomalous Components.
- According to IEEE standards Glossary, " A quantitative measure of the degree in which a system, component, or process possesses given attributes.

Need of Software Measurement:

- Create the quality of the current product or process.
- Anticipate future qualities of the product or process.
- Enhance the quality of a product or process.
- Regulate the state of the project in relation to budget and schedule.

Measurement of Principle :

- The unit of measure concept is a standard convention used in accounting, under which all transactions must be consistently recorded using the same currency.
- If a transaction involves receipts or payments in a different currency, the amount is converted

to the home currency used by an organization before being recorded.

- To measure a physical quantity a convenient and a fixed part or portion of that is considered as standard and the quantity is measured with reference to that standard and everywhere that fixed portion or part is used.

Measurement Principles:

- **Formulation:** The derivation of software measures and metrics appropriate for the representation of the software that is begin considered.
- **Collection :** The mechanism used to accumulate data required to derive the formulated metrics.
- **Analysis :** The computation of metrics and the application of mathematical tools.
- **Interpretation :** The evaluation of metrics in an effort to gain insight into the quality of the representation.
- **Feedback :** Recommendations derived from the interpretation of product metrics transmitted to the software team.

Scope of Software Metrics:

- Cost and effort estimation.
- Productivity measures and model.
- Data collection.
- Quantity models and measures.
- Reliability models.
- Performance and evaluation models.
- Structural and complexity metrics.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=bnydxXPN_rI

Important Books/Journals for further learning including the page nos.:
Software Engineering: A Practitioner Approach–Roger S. Pressman , McGraw Hil, 2010 **(Page No : 87 - 90)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

MCA

I / II

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs.R.Pavithra**

Unit : **IV -SOFTWARE METRICS**

Date of Lecture: 27.03.2021

Topic of Lecture: Classification of Metrics

Introduction :

- Estimation of the size of the software is an essential part of Software Project Management.
- It helps the project manager to further predict the effort and time which will be needed to build the project.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Class, Object, Access Specifier, Software, Operation

Detailed content of the Lecture:

Classification of Metrics :

- It helps the project manager to further predict the effort and time which will be needed to build the project. Various measures are used in project size estimation. Some of these are:

Product Metrics:

- Product metrics are quantifiable data points that a business tracks and analyzes to gauge the success of its product.
- Examples of product metrics include conversion rate, churn rate, and monthly recurring revenue.
- **Size Metrics:** Size Oriented Metrics are also used for measuring and comparing productivity of programmers. It is a direct measure of a Software. The size measurement is based on lines of code computation
- **Lines of Code :** As the name suggests, LOC count the total number of lines of source code in a project.
- **Complexity Metrics :** A complexity measure is a cyclomatic complexity in which the complexity of a module is the number of independent cycles in the flow graph of the module.

Product Metrics:

- In software development process, a working product is developed at the end of each successful phase. Each product can be measured at any stage of its development.
- Metrics are developed for these products so that they can indicate whether a product is developed according to the user requirements.
- If a product does not meet user requirements, then the necessary actions are taken in the respective phase.
- Product metrics help software engineer to detect and correct potential problems before they result in catastrophic defects. In addition, product metrics assess the internal product attributes in order to know the efficiency of the following.
- Analysis, design, and code model
- Potency of test cases
- Overall quality of the software under development.

Various metrics formulated for products in the development process are listed below.

- **Metrics for analysis model:** These address various aspects of the analysis model such as system functionality, system size, and so on.
- **Metrics for design model:** These allow software engineers to assess the quality of design and include architectural design metrics, component-level design metrics, and so on.
- **Metrics for source code:** These assess source code complexity, maintainability, and other characteristics.
- **Metrics for testing:** These help to design efficient and effective test cases and also evaluate the effectiveness of testing.
- **Metrics for maintenance:** These assess the stability of the software product.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=aWAnNHXIKww

Important Books/Journals for further learning including the page nos.:
Software Engineering: A Practitioner Approach–Roger S. Pressman , McGraw Hil, 2010 **(Page No : 145 - 147)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code **: SOFTWARE ENGINEERING / 19CAB10**

Course Faculty **: Mrs. R.Pavithra**

Unit **: IV -SOFTWARE METRICS**

Date of Lecture: 29.03.2021

Topic of Lecture: Measuring Process and Product attributes

Introduction :

- Software measurement is a quantified attribute (see also: measurement) of a characteristic of a software product or the software process. It is a discipline within software engineering.
- Product attributes define the characteristics of products, enabling you to uniquely describe a product.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Object, Classes, Attributes, CASE Tools, Methods

Detailed content of the Lecture:

Measuring Process:

- A measurement is an manifestation of the size, quantity, amount or dimension of a particular attributes of a product or process.
- Software measurement is a titrate impute of a characteristic of a software product or the software process.
- It is an authority within software engineering. Software measurement process is defined and governed by ISO Standard.

Need of Software Measurement:

Software is measured to:

- Create the quality of the current product or process.
- Anticipate future qualities of the product or process.
- Enhance the quality of a product or process.
- Regulate the state of the project in relation to budget and schedule.

Classification of Software Measurement:

There are 2 types of software measurement:

1. **Direct Measurement:**
   In direct measurement the product, process or thing is measured directly using standard scale.

2. **Indirect Measurement:**
   In indirect measurement the quantity or quality to be measured is measured using related parameter i.e. by use of reference.

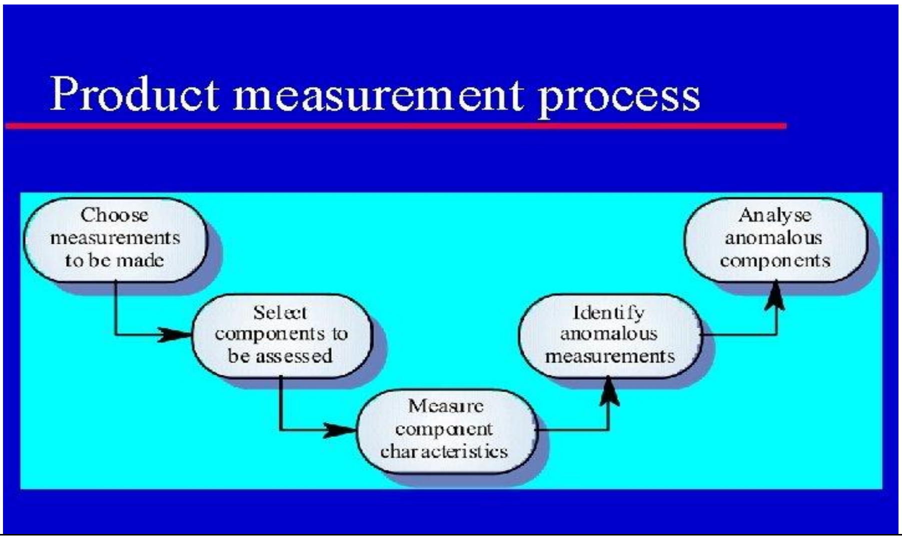
Metrics:

A metrics is a measurement of the level that any impute belongs to a system product or process. There are 4 functions related to software metrics:

- Planning
- Organizing
- Controlling
- Improving

Characteristics of software Metrics:

1. **Quantitative:**
   Metrics must possess quantitative nature.It means metrics can be expressed in values.
2. **Understandable:**
   Metric computation should be easily understood ,the method of computing metric should be clearly defined.
3. **Applicability:**
   Metrics should be applicable in the initial phases of development of the software.
4. **Repeatable:**
   The metric values should be same when measured repeatedly and consistent in nature.
5. **Economical:**
   Computation of metric should be economical.
6. **Language Independent:**
   Metrics should not depend on any programming language.



Product measurement process

Choose measurements to be made
Select components to be assessed
Measure component characteristics
Identify anomalous measurements
Analyse anomalous components

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=460wei5M1hM

Important Books/Journals for further learning including the page nos.:
Software Engineering: A Practitioner Approach–Roger S. Pressman , McGraw Hil, 2010 **(Page No : 149 - 153)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

DESIGNING YOUR FUTURE
Estd. 2000

IQAC

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **IV -SOFTWARE METRICS**

Date of Lecture:30.03.2021

Topic of Lecture: Direct and Indirect measures , Software Metrics.

Introduction :

- Direct Measurement: In direct measurement the product, process or thing is measured directly using standard scale.
- Indirect Measurement: In indirect measurement the quantity or quality to be measured is measured using related parameter i.e. by use of reference

Prerequisite knowledge for Complete understanding and learning of Topic:

- Object
- Classes
- Attributes
- CASE Tools
- Methods

Detailed content of the Lecture:

Direct and Indirect Measures:

- Direct measures include software processes like cost and effort applied and products like lines of code produced, execution speed, and other defects that have been reported.
- Indirect measures include products like functionality, quality, complexity, reliability, maintainability, and many more.

Classification of Software Metrics:

There are 2 types of software metrics:

1. **Product Metrics:**
   Product metrics are used to evaluate the state of the product, tracing risks and under covering prospective problem areas. The ability of team to control quality is evaluated.
2. **Process Metrics:**
   Process metrics pay particular attention on enhancing the long term process of the team or organization.
3. **Project Metrics:**
   Project matrix is describes the project characteristic and execution process.
   - Number of software developer

- Staffing pattern over the life cycle of software
- Cost and schedule
- Productivity

Software Metrics:

- A software metric is a measure of software characteristics which are measurable or countable.
- Software metrics are valuable for many reasons, including measuring software performance, planning work items, measuring productivity, and many other uses.

Types of Metrics:

- **Internal metrics:** Internal metrics are the metrics used for measuring properties that are viewed to be of greater importance to a software developer. For example, Lines of Code (LOC) measure.
- **External metrics:** External metrics are the metrics used for measuring properties that are viewed to be of greater importance to the user, e.g., portability, reliability, functionality, usability, etc.
- **Hybrid metrics:** Hybrid metrics are the metrics that combine product, process, and resource metrics. For example, cost per FP where FP stands for Function Point Metric.
- **Project metrics:** Project metrics are the metrics used by the project manager to check the project's progress.
- Data from the past projects are used to collect various metrics, like time and cost; these estimates are used as a base of new software.
- That as the project proceeds, the project manager will check its progress from time-to-time and will compare the effort, cost, and time with the original effort, cost and time.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=eVYtuvIFpoA

Important Books/Journals for further learning including the page nos.:
Software Engineering: A Practitioner Approach–Roger S. Pressman , McGraw Hil, 2010 **(Page No : 179 - 183)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

DESIGNING YOUR FUTURE
Estd. 2000

IQAC

LECTURE HANDOUTS

MCA

I / II

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **IV -SOFTWARE METRICS**

Date of Lecture:31.03.2021

Topic of Lecture: Cost Estimation

Introduction :

- Software cost estimation is the process of predicting the effort required to develop a software system.
- We then highlight the cost estimation models that have been proposed and used successfully. Models may be classified into 2 major categories: algorithmic and non-algorithmic.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Database
- Object
- Methods
- Messages

Detailed content of the Lecture:

Cost Estimation:

- Cost estimation simply means a technique that is used to find out the cost estimates.
- The cost estimate is the financial spend that is done on the efforts to develop and test software in Software Engineering.
- Cost estimation models are some mathematical algorithms or parametric equations that are used to estimate the cost of a product or a project.

Models may be classified into 2 major categories :

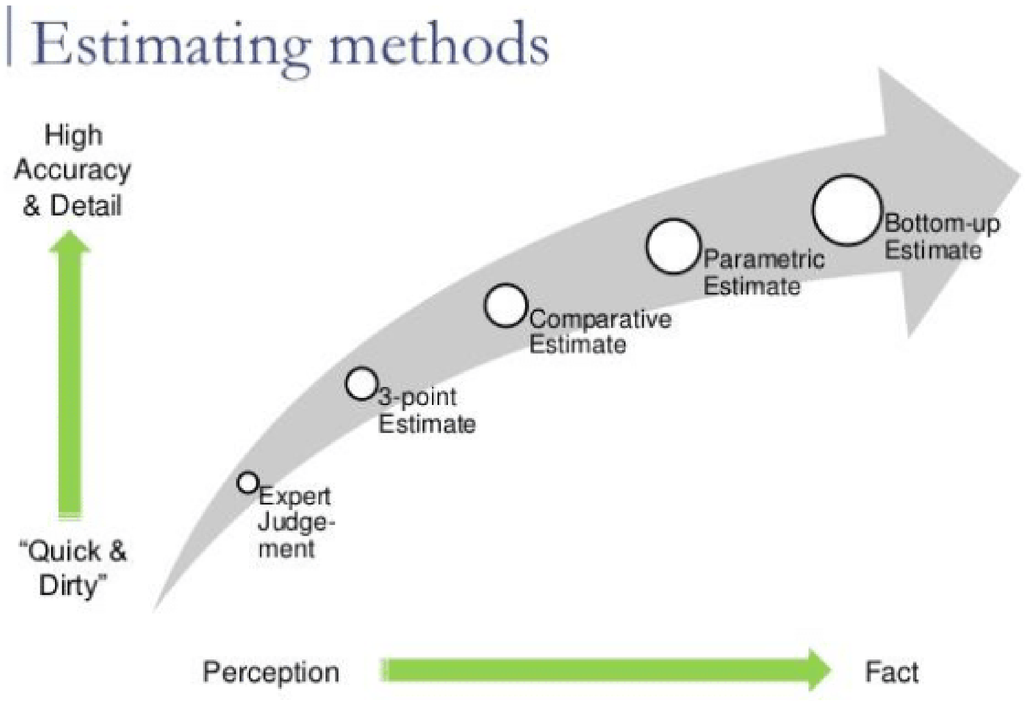
- Algorithmic Methods.
- Non-Algorithmic Methods .

\

Non-Algorithmic Methods:

Methods of Cost Estimation in Projects

- Expert Judgement
- Analogous Estimating
- Parametric Estimating
- Bottom-up Estimating
- Three-point Estimating
- Data Analysis (Alternative analysis/Reserve analysis)
- Project Management Information system
- Decision making (voting)



Algorithmic Methods:

- Linear Models.
- Multiplicative Models.
- Power Function Models.
- Model Calibration Using Linear Regression.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=dWr399eZXfE

Important Books/Journals for further learning including the page nos.:
Software Engineering: A Practitioner Approach–Roger S. Pressman , McGraw Hil, 2010 **(Page No : 199 - 203)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

DESIGNING YOUR FUTURE
Estd. 2000

IQAC

LECTURE HANDOUTS

MCA

I / II

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **IV -SOFTWARE METRICS**

Date of Lecture: 01.04.2021

Topic of Lecture: Reliability

Introduction :

- Software reliability is also defined as the probability that a software system fulfills its assigned task in a given environment for a predefined number of input cases, assuming that the hardware and the input are free of error.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Process Model
- Object
- UML Diagram

Detailed content of the Lecture:

Reliability:

- Software reliability engineering (SRE) assesses how well software-based products and services meet users' operational needs.
- SRE uses quantitative methods based on reliability measures to do this assessment.
- SRE uses such quantitative methods as statistical estimation and prediction, measurement, and modeling.
- A software reliability model indicates the form of a random process that defines the behavior of software failures to time.
- Software reliability models have appeared as people try to understand the features of how and why software fails, and attempt to quantify software reliability.
- Over 200 models have been established since the early 1970s, but how to quantify software reliability remains mostly unsolved.
- There is no individual model that can be used in all situations. No model is complete or even representative.

Most software models contain the following parts:

- Assumptions
- Factors
- A mathematical function that includes the reliability with the elements. The mathematical function is generally higher-order exponential or logarithmic.

Reliability Models

- A reliability growth model is a numerical model of software reliability, which predicts how software reliability should improve over time as errors are discovered and repaired.
- These models help the manager in deciding how much efforts should be devoted to testing. The objective of the project manager is to test and debug the system until the required level of reliability is reached.
- Reliability metrics are used to quantitatively expressed the reliability of the software product. The option of which parameter is to be used depends upon the type of system to which it applies & the requirements of the application domain.
- Measuring software reliability is a severe problem because we don't have a good understanding of the nature of software.
- It is difficult to find a suitable method to measure software reliability and most of the aspects connected to software reliability.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=m0W8nnupcUk

Important Books/Journals for further learning including the page nos.:
Software Engineering: A Practitioner Approach–Roger S. Pressman , McGraw Hil, 2010 **(Page No : 218 - 221)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **IV -SOFTWARE METRICS**

Date of Lecture: 03.04.2021

Topic of Lecture: Software Quality Assurance

Introduction :

- Software quality assurance (SQA) is a means and practice of monitoring the software engineering processes and methods used in a project to ensure proper quality of the software.
- The role of QA is exactly that – to make sure that the software gives your customers exactly what they expect.
- The QA team would define the features of the deliverable and then work through each step of the development process to ensure that they are being delivered

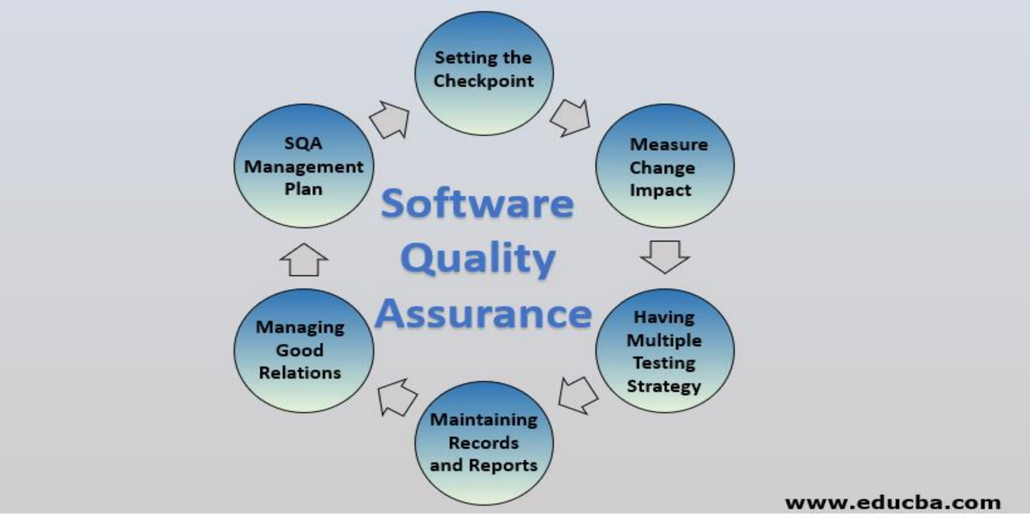
Prerequisite knowledge for Complete understanding and learning of Topic:

- Object
- Database
- Interface
- Design

Detailed content of the Lecture:

Software Quality Assurance:

- Software Quality Assurance (SQA) is simply a way to assure quality in the software. It is the set of activities which ensure processes, procedures as well as standards are suitable for the project and implemented correctly.



Software Quality Assurance has:

1. A quality management approach
2. Formal technical reviews
3. Multi testing strategy
4. Effective software engineering technology
5. Measurement and reporting mechanism

Major Software Quality Assurance Activities:

1. **SQA Management Plan:**
   Make a plan for how you will carry out the sqa through out the project. Think about which set of software engineering activities are the best for project. check level of sqa team skills.

2. **Set The Check Points:**
   SQA team should set checkpoints. Evaluate the performance of the project on the basis of collected data on different check points.

3. **Multi testing Strategy:**
   Do not depend on a single testing approach. When you have a lot of testing approaches available use them.

4. **Measure Change Impact:**
   The changes for making the correction of an error sometimes re introduces more errors keep the measure of impact of change on project. Reset the new change to change check the compatibility of this fix with whole project.

5. **Manage Good Relations:**
   In the working environment managing good relations with other teams involved in the project development is mandatory. Bad relation of sqa team with programmers team will impact directly and badly on project. Don’t play politics.

<u>Software Quality Metrics:</u>

- Product Metrics.
- Process Metrics.
- Project Metrics.

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=tj2LwVZ6NX4

Important Books/Journals for further learning including the page nos.:
Software Engineering: A Practitioner Approach–Roger S. Pressman , McGraw Hil, 2010 **(Page No : 232 - 236)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

I / II

MCA

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **IV -SOFTWARE METRICS**

Date of Lecture: 05.04.2020

Topic of Lecture: Standards

Introduction :

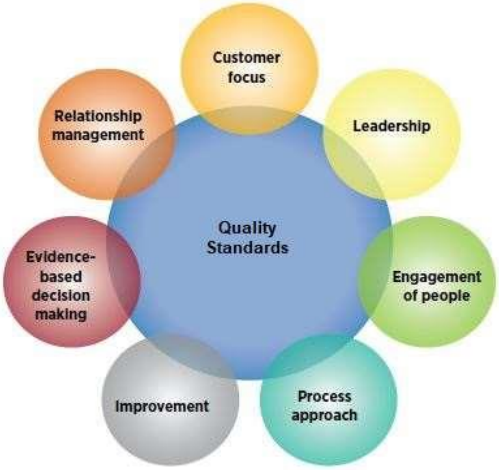
- The quality assurance system in which quality components can be organizational structure, responsibilities, procedures, processes, and resources for implementing quality management.
- Quality standards are defined as documents that provide requirements, specifications, guidelines, or characteristics that can be used consistently to ensure that materials, products, processes, and services are fit for their purpose.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Model
- Pattern
- User Interface
- Controller

Detailed content of the Lecture:

- Standards provide organizations with the shared vision, understanding, procedures, and vocabulary needed to meet the expectations of their stakeholders.
- Because standards present precise descriptions and terminology, they offer an objective and authoritative basis for organizations and consumers around the world to communicate and conduct business.



<u>Importance of International Standards:</u>

- Worldwide Progress in Trade Liberalization.
- Interpenetration of Sectors.
- Worldwide Communications Systems.
- Global Standards for Emerging Technologies.
- Developing Countries.

ISO 9126 Quality Standard:

Quality Models:

- The quality model determines which quality characteristics will be taken into account when evaluating the properties of a software product are precisely what is represented in the quality model, which categorizes the product quality into characteristics and sub-characteristics.
- Functionality.
- Reliability.
- Usability.
- Efficiency.
- Maintainability.
- Portability.

External Metrics:

- External metrics are applicable to running software.

Internal Metrics:

- Internal metrics are those which do not rely on software execution .

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=DfBnData5-0

Important Books/Journals for further learning including the page nos.:
Software Engineering: A Practitioner Approach–Roger S. Pressman , McGraw Hil, 2010 **(Page No : 239 - 242)**

Course Faculty

Verified by HOD

DESIGNING YOUR FUTURE
Estd. 2000

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

IQAC

LECTURE HANDOUTS

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **IV -SOFTWARE METRICS**

Date of Lecture: 06.04.2021

Topic of Lecture: COCOMO model

Introduction :

- The Constructive Cost Model (COCOMO) is a procedural software cost estimation model developed by Barry W. Boehm.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Object
- Records
- UML Diagrams
- Class

Detailed content of the Lecture:

COCOMO model:

- The Basic COCOMO model helps to obtain approximate estimate of parameters related to project.
- The main assumption of basic COCOMO model is that both the effort and development time are function of the software product size alone.
- Boehm after studying historical data collected from lots of real time project derived an expression.

- The expression derived by Boehm is represented as,
- Effort = a1 * (KLOC)a2 PM
- Timedev = b1 * (Effort)b2 Months

- Here,
- KLOC – Estimated size of software product expressed in terms of Kilo Lines Of Code
- Timedev – Estimated time in months required to develop the software product
- Effort – It represents the total effort required in terms of PM (Person Months)to develop the software.
- a1, a2, b1,b2 are constants for each category of software products.
- According to Boehm, every line of source text should be calculated as one LOC irrespective of the actual number of instructions on that line.

Effort estimation:

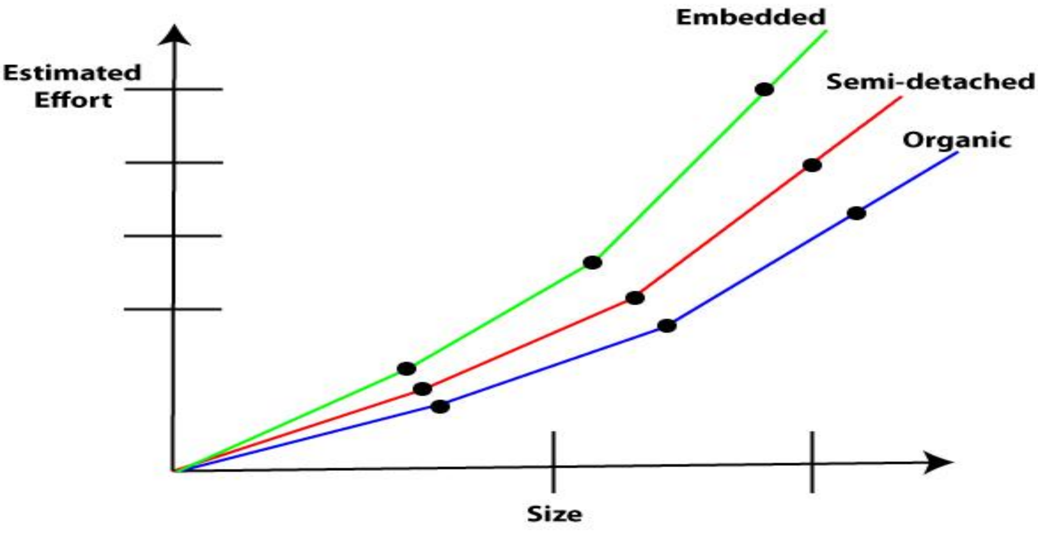
- Estimated effort required for different types of software products can be calculated based on below basic cocomo model formula derived by Boehm,
- Effort for Organic type project = 2.4*(KLOC)

- Effort for Semidetached type project = 3.0*(KLOC)
- Effort for Embedded type project = 3.6*(KLOC)

Time estimation:

- Development time required for different types of software products can be calculated based on below basic COCOMO model formula derived by Boehm,
- Time for Organic type project development = 2.5*(KLOC)
- Time for Semidetached type project development = 2.5*(KLOC)
- Time for Embedded type project development = 2.5*(KLOC)

- The time and effort value calculated using COCOMO are the representation of doing the development work in the shortest possible time without unduly increasing manpower cost.
- Effort and time duration estimation calculated using COCOMO model says:
- if we try to complete the project in shorter time than the estimated one, the development cost will increase to great extent.
- if we try to complete the project in longer time than the estimated one, then there appears almost no decrement in estimated cost.



Effort versus product size

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=6ySF9wNDAZo

Important Books/Journals for further learning including the page nos.:
Software Engineering: A Practitioner Approach–Roger S. Pressman , McGraw Hil, 2010 **(Page No : 249 - 253)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

Course Name with Code : SOFTWARE ENGINEERING / 19CAB10

Course Faculty : Mrs. R.Pavithra

Unit : V -SCM

Date of Lecture: 07.04.2021

Topic of Lecture: Introduction

Introduction :

- Software engineering, software configuration management (SCM or S/W CM) is the task of tracking and controlling changes in the software, part of the larger cross-disciplinary field of configuration management.
- SCM practices include revision control and the establishment of baselines.

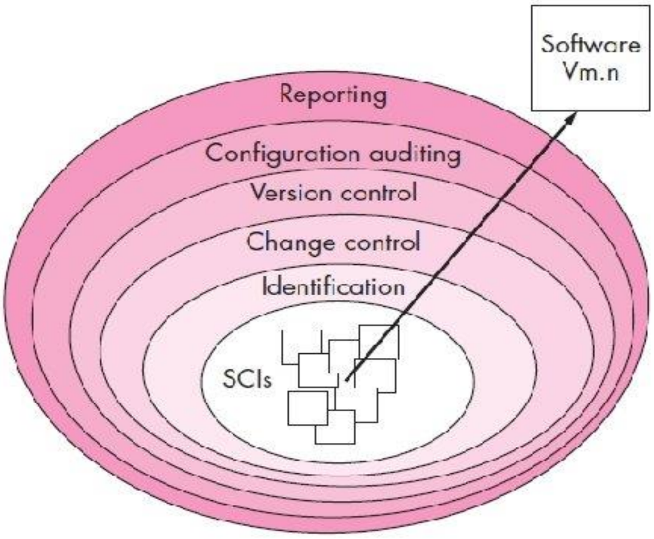
Prerequisite knowledge for Complete understanding and learning of Topic:

- Object, Records, UML Diagrams, Class

Detailed content of the Lecture:

SCM:

- There are multiple people working on software which is continually updating
- It may be a case where multiple version, branches, authors are involved in a software config project, and the team is geographically distributed and works concurrently
- Changes in user requirement, policy, budget, schedule need to be accommodated.



- Software should able to run on various machines and Operating Systems
- Helps to develop coordination among stakeholders
- SCM process is also beneficial to control the costs involved in making changes to a system.

The SCM Process are,

- Configuration identification,
- Change control,

- Version control,
- Configuration auditing
- Reporting.

- These tasks relate to software configuration items (SCIs) and can be seen as concentric layers that apply to SCIs as the project progresses.
- The process of delivering a product from raw material to the consumer.
- It includes supply planning, product planning, demand planning, sales and operations planning, and supply management.

Tasks in SCM process:

- Configuration Identification
- Baselines
- Change Control
- Configuration Status Accounting
- Configuration Audits and Reviews

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=mPwfYxxvVEk

Important Books/Journals for further learning including the page nos.:
Software Engineering–Sommerville ,Addison Wesley-Longman, 2004 **(Page No : 44-49)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

I / II

MCA

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs.R.Pavithra**

Unit : **V -SCM**

Date of Lecture: 08.04.2021

Topic of Lecture: Need for SCM

Introduction :

- In Software Engineering, Software Configuration Management(SCM) is a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle.
- The primary goal is to increase productivity with minimal mistakes.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Use cases
- Generalization
- Abstraction

Detailed content of the Lecture:

Need for SCM :

- Software Configuration Management is a set of activities that have been developed to manage change throughout the life cycle of computer software.
- Computer programs.
- Documents that describe the program, and
- Data.

Elements of configuration Management System:

- **Process Elements:** There are a collection of procedures and tasks that define an effective approach to change management for all constituencies involved in the management, engineering, and use of computer software.

- **Component Elements:** These are a set of tools coupled within a file management system.

- **Construction Elements:** These are a set of tools that automate the construction of software by ensuring that the proper set of validate components.

- **Human Elements:** These are used to implement effective SCM the software team uses a set of tools and process features.

Terminology:

- **Configuration Item :** An entity within a configuration that satisfies an end use function and that can be uniquely identified at a given reference point.

- **Baseline:** A formally approved version of a item, regardless of media, formally designed and fixed at a specific time during the configuration items life cycle.

- **SCM Directories :** Programmers Directory.
  Master Directory.
  Software Repository.

- **Version :** An initial release or re-release of a configuration item associated with a complete compilation or recompilation of the item.

- **Revision :** Change to a version that corrects only errors in the design / code, but does not affect the documented functionality.

- **Release :** The fprmal distributed of an approved version.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=IYT_ZsPeQyI

Important Books/Journals for further learning including the page nos.:
Software Engineering–Sommerville ,Addison Wesley-Longman, 2004 **(Page No : 123-127)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **V -SCM**

Date of Lecture: 09.04.2021

Topic of Lecture: Version Control

Introduction :

- Version Control System (VCS), also known as Source Control Management (SCM), is a way to manage and document changes developers make to software code.
- In this blog post, I will explain why we need VCS, provide best practices and recommend a few VCS tools

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Analysis
- Use case
- Classes
- Interaction Diagram (UML Diagram)

Detailed content of the Lecture:

Version Control:

Frequent Code Regression :

- Discarding the works of other developers can happen very frequently, resulting in a high risk of regression each time a new change is made.
- Even if all developers start with the same version of the software and even if they all work on different features at a time, as soon as they commit their revision they will be prompted to make a choice between their current revision and the ones in the repository.

Reduces Development Velocity :

- Since it is hard to tell which revision should be kept, the chances of deleting others' work are enormous.
- Without VCS, a software development team is very likely to redo the work over again and often. In the attempt to advance faster, by involving additional developers, for instance, the risk factor increases.
- Without a way to track changes, any change made has a great risk to be discarded by the following update.

Reduces System Maintainability :

- No tracking of code changes makes it hard to tell what may have caused a bug or a regression, making it difficult to fix the system and advance it.

Advantages:

- All changes are attributable
- Everything can be tracked and reverting is made easier
- Better conflict resolution
- Easier code maintenance and code quality monitoring
- Less software regression
- Better organization and a better communication

Version Control Software Features & capabilities:

- Centralized reviewable, retrievable version history
- Digital asset (e.g. binary files) storage
- Pull request, code review, collaboration tools
- Parallel development streams & branches
- Isolated code branches
- Branch creation or deletion
- Compare and merge version branches
- Revert code to previous versions
- Code version management (e.g. conflict resolution)
- Role-based access and control

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=Yc8sCSeMhi4

Important Books/Journals for further learning including the page nos.:
Software Engineering–Sommerville ,Addison Wesley-Longman, 2004 **(Page No : 142-145)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **V -SCM**

Date of Lecture:10.04.2021

Topic of Lecture: SCM Process

Introduction :

- The process of delivering a product from raw material to the consumer.
- It includes supply planning, product planning, demand planning, sales and operations planning, and supply management.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Dependencies between packages
- Dataflow Diagram
- UML
- Reservation

Detailed content of the Lecture:

SCM Process:

- Supply chain management (SCM) is the centralized management of the flow of goods and services and includes all processes that transform raw materials into final products.
- By managing the supply chain, companies can cut excess costs and deliver products to the consumer faster.

Configuration Identification:

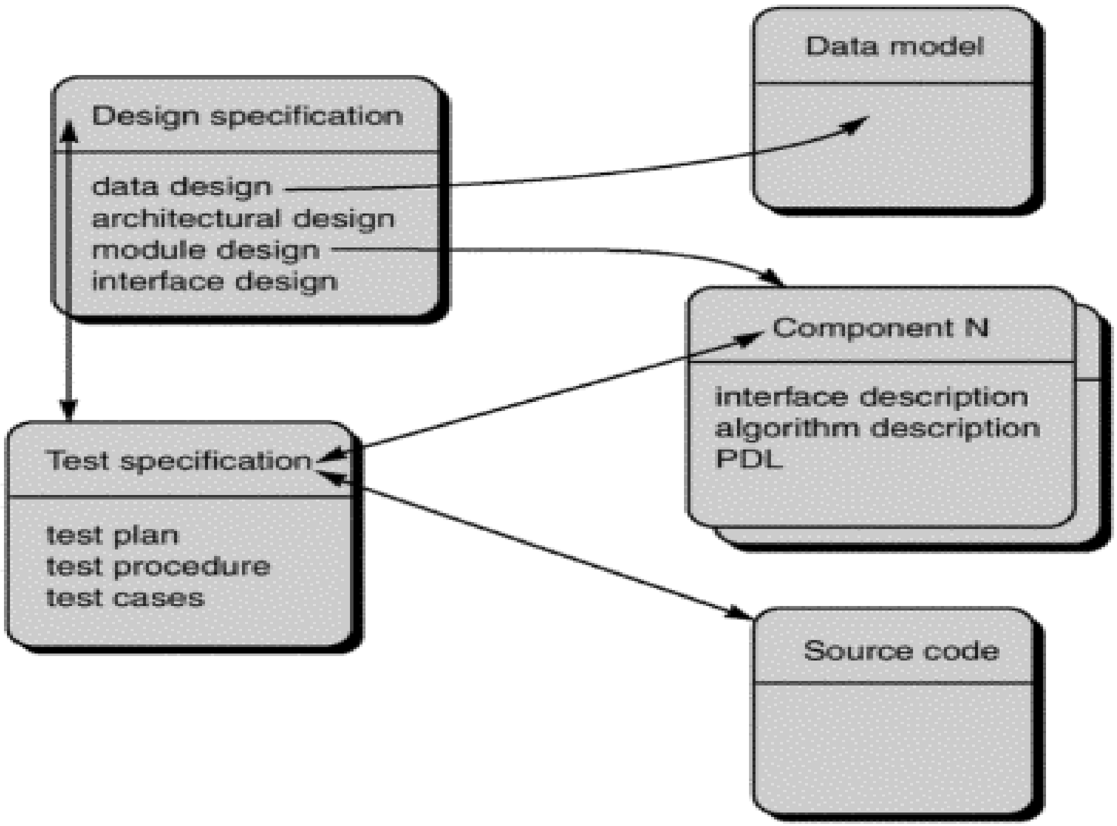
- Configuration identification is the process of identifying the attributes that define every aspect of a configuration item.
- A configuration item is a product (hardware and/or software) that has an end-user purpose.
- These attributes are recorded in configuration documentation and base lined.
- Identifying Item to be Controlled.
- Software Configuration.
- Software Configuration Items.
- Software Version.
- Baseline.

Categories of Objects Associated with Software Development:

- Controlled Objects.
- Pre-controlled Objects.
- Uncontrolled Objects.

Configuration Control:

- Configuration control is an important function of the configuration management discipline.
- Its purpose is to ensure that all changes to a complex system are performed with the knowledge and consent of management.



Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=AaHaLjuzUm8

Important Books/Journals for further learning including the page nos.:
Software Engineering–Sommerville ,Addison Wesley-Longman, 2004 **(Page No : 134-137)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

MCA

I / II

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **V -SCM**

Date of Lecture:12.04.2021

Topic of Lecture: Software Configuration Items

Introduction :

- An aggregation of hardware, software, or both, that is designated for configuration management and treated as a single entity in the configuration management process.-

Prerequisite knowledge for Complete understanding and learning of Topic:

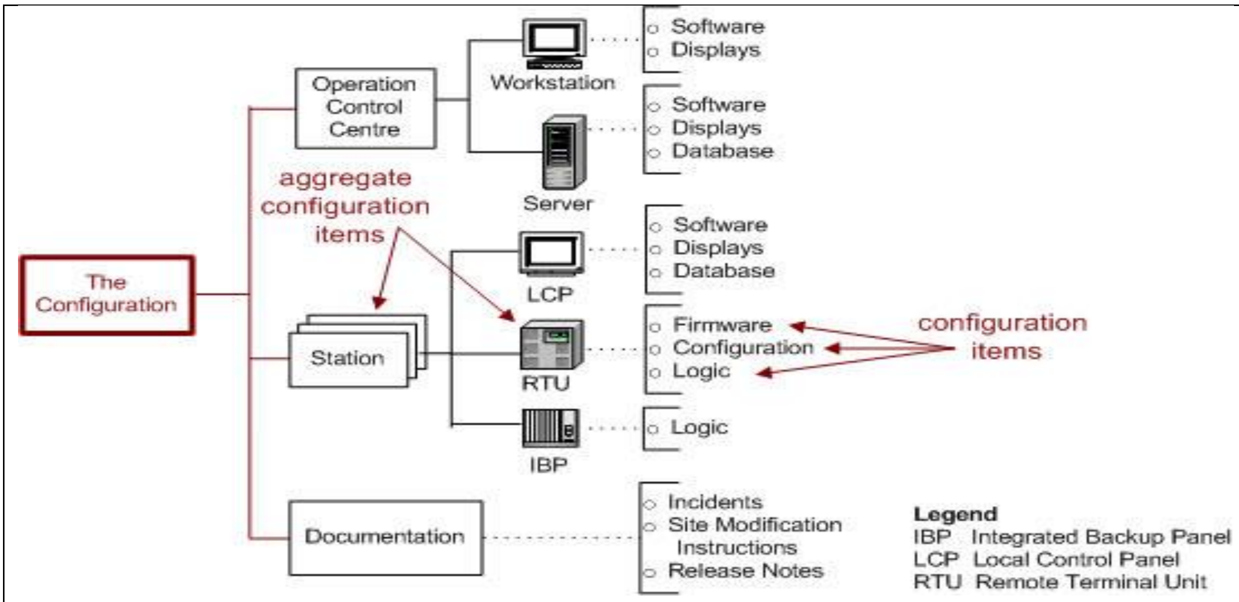
- Client
- Server
- Network
- UML Diagrams

Detailed content of the Lecture:

Software Configuration Items :

- A component of a system that is treated as a self contained unit for the purposes of identification and change control.
- All configuration items (CIs) are uniquely identified by CI registration codes and version numbers.

Examples of Configuration Items:-



Top-level configuration items:

- A top-level configuration item is a CI used as the starting point for organizing and promoting a set of related configuration items.
- An example of a top-level CI is a computer system. It can have many child CIs, such as an operating system, application software, and hardware components.
- When viewing a list of CIs in the Configuration Items or Actual Configuration Items application, you can see which ones are top-level CIs by checking the Top-level column.
- Knowing which CIs are top-level is important because when you create authorized CIs by promoting actual CIs, you must choose actual CIs that correspond to the top-level authorized CIs for which you have defined promotion templates.

Discovering configuration items:

- The configuration items in your environment are discovered by sensors or operational management products.

Refining configuration items:

- As configuration items are discovered and imported into Control Desk as actual configuration items, further reduction of the amount of data can take place.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=beA6sp0CBCc

Important Books/Journals for further learning including the page nos.:
Software Engineering–Sommerville ,Addison Wesley-Longman, 2004 **(Page No : 184-188)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **V -SCM**

Date of Lecture: 15.04.2021

Topic of Lecture: Taxonomy

Introduction :

- Taxonomy software can analyze a text and automatically assign it to a place in the taxonomy, with the option for users to manually override or modify the resulting classification.
- Taxonomy software can also integrate with or send output to content management, portal, and other enterprise management systems.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Encryption
- Decryption
- Security
- Hashing

Detailed content of the Lecture:

Taxonomy:

- A taxonomy is a hierarchical framework, or schema, for the organization of organisms, inanimate objects, events and/or concepts.
- From a computing standpoint, however, there are some important differences between a classic Linnaean taxonomy and a taxonomy that is useful in application logic.
-

Business process engineering tools:

- By modeling the strategic information requirements of an organization, business process engineering tools provide a "meta-model" from which specific information systems are derived. Rather than focusing on the requirements of a specific application.

Process modeling and management tools:

- If an organization works to improve a business (or software) process, it must first understand it. Process modeling tools (also called process technology tools) are used to represent the key elements of a process so that it can be better understood.

Project planning tools:

- Tools in this category focus on two primary areas: software project effort and cost estimation and project scheduling.
- Estimation tools compute estimated effort, project duration, and recommended number of people for a project.

Risk analysis tools:

- Identifying potential risks and developing a plan to mitigate, monitor, and manage them is of paramount importance in large projects.

Project management tools:

The project schedule and project plan must be tracked and monitored on a continuing basis. In addition, a manager should use tools to collect metrics that will ultimately provide an indication of software product quality. Tools in the category are often extensions to project planning tools.

Requirements tracing tools:

- When large systems are developed, things "fall into the cracks." That is, the delivered system does not fully meet customer specified requirements.

Metrics and management tools:

- Software metrics improve a manager's ability to control and coordinate the software engineering process and a practitioner's ability to improve the quality of the software that is produced.

Documentation tools:

- Document production and desktop publishing tools support nearly every aspect of software engineering and represent a substantial "leverage" opportunity for all software developers.

System software tools:

- CASE is a workstation technology. Therefore, the CASE environment must accommodate high-quality network system software, object management services, distributed component support, electronic mail, bulletin boards, and other communication capabilities.

Video Content / Details of website for further learning (if any):
https://www.youtube.com/watch?v=xv7EovHEGAI

Important Books/Journals for further learning including the page nos.:
Software Engineering–Sommerville ,Addison Wesley-Longman, 2004 **(Page No : 123-126)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

MCA **I / II**

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **V -SCM**

Date of Lecture: 17.04.2021

Topic of Lecture: CASE Repository

Introduction :

- Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored.
- Lower Case Tools - Lower CASE tools are used in implementation, testing and maintenance.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Classes
- Models
- Extensibility
- Package

Detailed content of the Lecture:

CASE Repository:

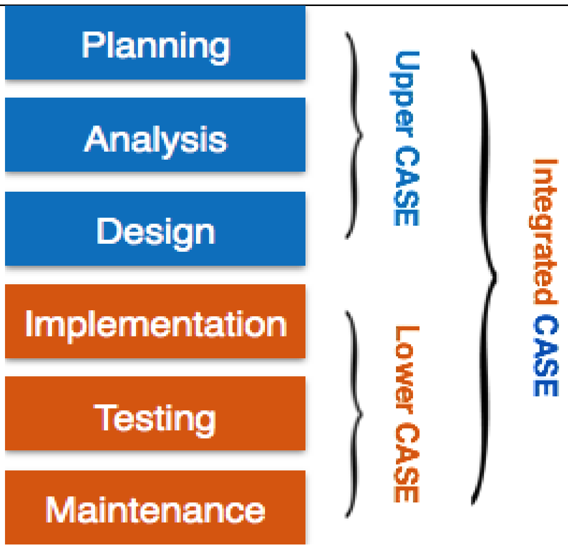
- A CASE Repository should be the representation, in data, of all relevant information about the system under development, in a consistent, complete form which is independent of its mode of entry and modification or subsequent use.

Components of CASE Tools:

- CASE tools can be broadly divided into the following parts based on their use at a particular SDLC stage:

Central Repository :

- CASE tools require a central repository, which can serve as a source of common, integrated and consistent information.
- Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored. Central repository also serves as data dictionary.



Upper Case Tools :

- Upper CASE tools are used in planning, analysis and design stages of SDLC.

Lower Case Tools :

- Lower CASE tools are used in implementation, testing and maintenance.

Integrated Case Tools:

- Integrated CASE tools are helpful in all the stages of SDLC, from Requirement gathering to Testing and documentation.
- CASE tools can be grouped together if they have similar functionality, process activities and capability of getting integrated with other tools.

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=rTvpPut8D8U

Important Books/Journals for further learning including the page nos.:
Software Engineering–Sommerville ,Addison Wesley-Longman, 2004 **(Page No : 234-237)**

Course Faculty

Verified by HOD

LECTURE HANDOUTS

I / II

MCA

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **V -SCM**

Date of Lecture: 19.04.2021

Topic of Lecture: Features

Introduction :

- The most common features of supply chain management software include: Inventory management - for tracking and managing the availability of raw materials, stocked goods or spare parts.
- This feature can also help with asset management, bar code integration and future inventory and price forecasting.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Case Tools
- UML
- Model
- Open Source

Detailed content of the Lecture:

Features :

- Software Configuration Management (SCM) Tools handle the task of tracking and controlling changes in the software.
- This includes identifying individual elements and configurations, tracking changes, and version selection, control, and base lining.
- Some products also include defect tracking capabilities.

SCM Tools Features & Capabilities:

- Version management / Single source of truth
- Concurrency management
- Flexible branching
- Regulatory compliance
- Integration with other development tools
- Workflow and process automation
- Rapid reliable rebuilds
- Automated cleanup of dead code branches
- Rapid hotfix and feature release.

Software Configuration Management vs. Configuration Management:

- Software configuration management is primarily focused on revision control.
- This category is part of a broader category called Configuration Management
- Configuration Management goes beyond revision control to include capabilities such as monitoring and asset discovery, patch management, software deployment and distribution, and automated administration and maintenance.
- This broader category has become important with the advent of DevOps which brings development and IT closer together through continuous deployment.
- Software Configuration Management is the process of tracking and controlling the software changes.

The basic features provided by any SCM tools are as follows:

- Concurrency Management
- Version Control
- Synchronization

Concurrency Management :

- When two or more tasks are happening at same time it is known as concurrent operation.
- If we talk concurrency in context to SCM it means that the same file being edited by multiple persons at the same time.
- If concurrency is not managed properly with SCM tools then it may lead to very severe problems.

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=6-_GddtUbMA

Important Books/Journals for further learning including the page nos.:
Software Engineering–Sommerville ,Addison Wesley-Longman, 2004 **(Page No : 254-256)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

DESIGNING YOUR FUTURE
Estd. 2000

IQAC

LECTURE HANDOUTS

MCA

I / II

Course Name with Code : **SOFTWARE ENGINEERING / 19CAB10**

Course Faculty : **Mrs. R.Pavithra**

Unit : **V -SCM**

Date of Lecture: 20.04.2021

Topic of Lecture: Web Engineering

Introduction :

- Automation of the software configuration management functions provides, among other things, for the automatic requirement of approved change requests for every baseline once the initial software baseline has been achieved, the mapping of baselines to change requests at release control reviews

Prerequisite knowledge for Complete understanding and learning of Topic:

- Case Tools, UML, Model, Open Source

Detailed content of the Lecture:

Web Engineering:

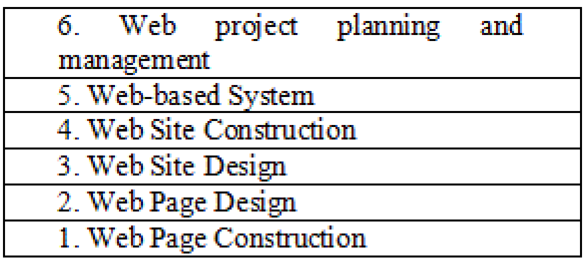
- Configuration management helps engineering teams build robust and stable systems through the use of tools that automatically manage and monitor updates to configuration data.
- Complex software systems are composed of components that differ in granularity of size and complexity.

Need for Web Engineering:

- The need for Web Engineering is felt (or dismissed) according to perceptions of the developers and managers, their experiences in creating applications made feasible by the new technologies, and the complexity of Web applications.
- In the early stages of Web development, White10 and Powell11, identified and emphasized the need for engineering as in Web Document Engineering and Web Site Engineering.
- Web Engineering, more generally, explicitly recognize the fact that good Web development requires multidisciplinary efforts and does not fit neatly into any of the existing disciplines.

Perceptions of Web Development:

- Levels of perception in Web Development For someone relatively new to Web development, be they developers, users or managers, the Web is manifested through the Web pages, the outcome of the simplest and most visible level



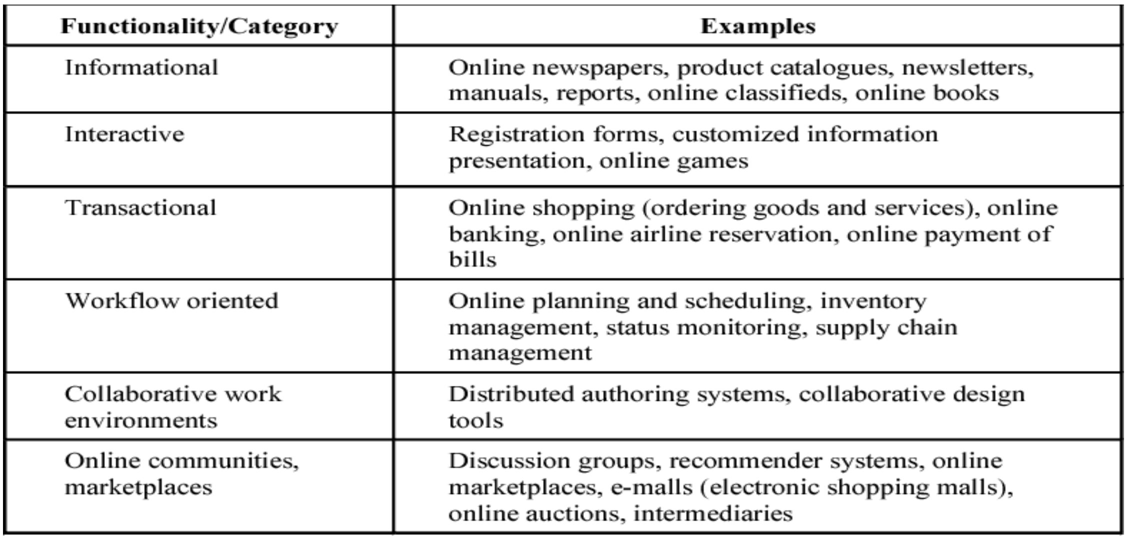
6. Web project planning and management
5. Web-based System
4. Web Site Construction
3. Web Site Design
2. Web Page Design
1. Web Page Construction

Multidisciplinary Nature of Web Development:

- Web applications handle information in its myriad forms (text, graphics, video, audio).
- Information sciences, multimedia, hypermedia and graphic design deal with structuring, processing, storing and presenting this information.
- Human-computer Interaction (HCI) and requirements engineering are essential to understand users and their requirements.
- Network management, general computing and simulation and modelling are required to deliver the information and desired functionality with an acceptable performance level.
- Software engineering, including new development methodologies, is 8 Web engineering essential for project and process management.

Evolution and Taxonomy of Web Applications :

- Web development within an organisation depends upon several factors. The motivation depends upon the initial purpose of using the Web (Web 'presence' or becoming a Web-based organisation), the customers expectations and the competitive environment.
- The drive to systematise development is subject to overall perception of the Web, as depicted in figure 1, and conscious policy decisions within the organisation. For example, a low level perception of the Web is likely to lead to ad hoc, sporadic efforts.



Functionality/Category	Examples
Informational	Online newspapers, product catalogues, newsletters, manuals, reports, online classifieds, online books
Interactive	Registration forms, customized information presentation, online games
Transactional	Online shopping (ordering goods and services), online banking, online airline reservation, online payment of bills
Workflow oriented	Online planning and scheduling, inventory management, status monitoring, supply chain management
Collaborative work environments	Distributed authoring systems, collaborative design tools
Online communities, marketplaces	Discussion groups, recommender systems, online marketplaces, e-malls (electronic shopping malls), online auctions, intermediaries

Video Content / Details of website for further learning (if any):

https://www.youtube.com/watch?v=8VWu_8c_7NE

Important Books/Journals for further learning including the page nos.:
Software Engineering–Sommerville ,Addison Wesley-Longman, 2004 **(Page No : 264-267)**

Course Faculty

Verified by HOD