R20

Software Engineering Lab Manual

Department of CSE

Vasireddy Venkatadri Institute of Technology
Nambur(V), Peda kakani(M), Guntur Dt.- 522508

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Vision of the Department
To facilitate quality education by focusing on assimilation, generation and dissemination of knowledge in the area of Computer Science & Engineering to transform students into socially responsible engineers.

Mission of the Department
○ Equip our graduates with the knowledge by student centric teaching-learning process and expertise to contribute significantly to the software industry and to continue to grow professionally. ○ To train socially responsible, disciplined engineers who work with good leadership skills and can contribute for nation building. ○ To make our graduates aware of cutting edge technologies and make them industry-ready engineers. ○ To shape the department into a centre of academic and research excellence.

Program Educational Objectives	
PEO-1	To provide the graduates with ***solid foundation in Computer Science and Engineering*** along with the fundamentals of Mathematics and Sciences with a view to impart in them high quality technical skills like modeling, analyzing, designing, programming and implementation with global competence and helps the graduates for life-long learning.
PEO-2	To prepare and motivate graduates with ***recent technological developments related to core subjects*** like Programming, Databases, Design of Compilers and Network Security aspects and future technologies so as to contribute effectively for Research & Development by participating in professional activities like publishing and seeking copy rights.
PEO-3	To train graduates to choose a ***decent career option either in high degree of employability/Entrepreneur or, in higher education*** by empowering students with ethical administrative acumen, ability to handle critical situations and training to excel in

	competitive examinations.
PEO-4	To train the graduates to have ***basic interpersonal skills and sense of social responsibility*** that paves them a way to become good team members and leaders.

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY, NAMBUR DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Course : B.Tech
Branch : CSE
Regulation : R20
Code : 20CSL401
Lab Name : software Engineering Lab
Total Marks 50
Internal Lab Marks 15
External Lab Marks 35

Evaluation of Internal Marks (15):

S. No.	Item	Marks
1	Day to Day Evaluation	5
2	Record	5
3	Experiment	5

Evaluation of External Marks (35):

S. No.	Item	Marks
1	Program	10
2	Procedure	10
3	Execution	10
4	Viva Voce	5

Course Objectives & Course Outcomes:

This course focuses on the use of data structures. Through the introduction of the most widely used data structures employed in solving commonly encountered problems (e.g. lists, trees, and graphs), students will learn different ways to organize data for easy access and efficient manipulation. Algorithms to solve classic problems (e.g. searching, sorting, hashing, graph algorithms, etc.) will be presented, as well as classic algorithm design strategies (e.g. divide-and-conquer and greedy algorithms).

Course Outcomes:

Students who complete this course will be able to:

Software Engineering Lab		
C328	Subject Code: RT32058	Year/Semester: II B.Tech./ I
C328.1	To apply requirement gathering techniques to create SRS for a defined problem.	
C328.2	To apply the cost, size, effort estimation techniques on a defined problem.	
C328.3	To assess the risk for a defined problem by applying Risk Assessment strategies like RMMM.	
C328.4	To model a real world problem using modern modeling tools.	
C328.5	To create test cases based on requirements and design.	
C328.6	To conduct FTRs as a measure of communication between him and the other stakeholders of the project.	

CO/ PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
C328.1	3	3	-	-	3	-	-	-	3	3	2	-
C328.2	3	3	-	-	-	-	-	-	3	3	3	-
C328.3	3	3	-	-	-	-	-	-	3	3	3	-
C328.4	3	3	3	3	-	-	-	-	3	3	3	-
C328.5	3	2	-	-	3	-	-	-	3	3	-	-
C328.6	-	-	-	-	-	-	-	-	3	3	3	-

LIST OF EXPERIMENTS

1. Do the Requirement Analysis and Prepare SRS
2. Using COCOMO model estimate effort.
3. Calculate effort using FP oriented estimation model.
4. Analyze the Risk related to the project and prepare RMMM plan.
5. Develop Time-line chart and project table using PERT or CPM project scheduling methods.
6. Draw E-R diagrams, DFD, CFD and structured charts for the project.
7. Design of Test cases based on requirements and design.
8. Prepare FTR
9. Prepare Version control and change control for software configuration items.

****

Experiment No-1

Aim: To Analyze requirements and Prepare SRS for Attendance

Maintenance System Software Requirements specification for Attendance

Maintenance System

Sample structure of SRS

1. Introduction
   - 1.1 Purpose
   - 1.2 Scope
   - 1.3 Overview
2. General Description
   - 2.1 User Manual
3. Functional Requirements
   - 3.1 Description
   - 3.2 Technical Issues
4. Interface Requirements
   - 4.1 GUI
   - 4.2 Hardware Interface
   - 4.3 Software Interface
5. Performance Requirements
6. Design Constraints
7. Other Non-functional Attributes
   - 7.1 Security
   - 7.2 Reliability
   - 7.3 Availability
   - 7.4 Maintainability
   - 7.5 Reusability
8. Operational Scenarios
9. Preliminary Schedule

Example: SRS Document for Attendance Management System

1. Introduction

1.1 Purpose

This document detailed functional and non-functional requirements for attendance maintenance system. The purpose of this document is that the requirements mentioned in it should be utilized by software developer to implement the system.

1.2 Scope

This system allows the teacher to maintain attendance record of the classes to which it is teaching. With the help of this system Teacher should be in a position to send e-mail to the students who remain absent for the class. The system provides a cumulative report at every month end for the corresponding class.

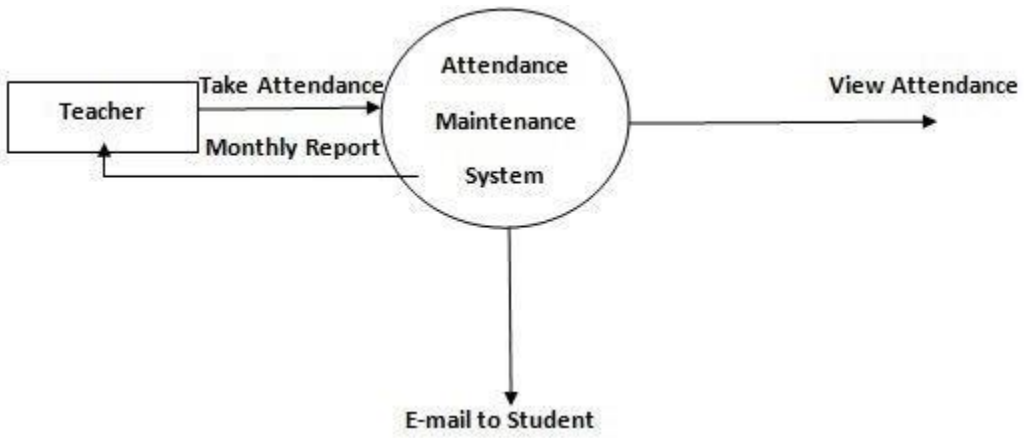
1.3 Overview:

This system provides an easy solution to the teacher to keep track of student attendance, and statistics.

2. General Description

This attendance maintenance system replaces the traditional, manual attendance system by which a lot of paper work will be reduced. The teacher should able to view photograph of student along with his attendance in his laptop. This is the primary feature of this system. Another feature is that Teacher can be allowed to edit particular record at desired time. The system should produce monthly attendance report. And there should be facility to send an e-mail/warning to the student remaining absent in the class.

Every Teacher should have laptop with wireless internet connection. A teacher may teach to different classes and a separate record for the corresponding classes should be maintained.



2.1 User manual:

The system should provide Help option in which how to operate the system should be explained. Also hard copy of this document should be given to the user in a booklet form.

3. Functional Requirements:

The identity of student is verified and then marked present at particular date and time. The system should display student photograph along with their names for the corresponding class. The student may be marked present or absent students. A statistical report should display individuals report or a cumulative report whenever required.

3.1 Description

The identity of student is verified and then marked present at particular date and time. The system should display student photograph along with their names for that corresponding class. The student may be marked present or absent depending upon his display individuals report or a cumulative report whenever requirement.

3.2 Technical Issues

The system should be implemented in VC++.

4. Interface Requirements

4.1 GUI

GUI 1: Main menu should provide option such as File, Edit, Report, help.

GUI 2: In File menu one can create a new record file or can open an existing record file.. GUI 3: The display of record

GUI 4: Report option should display statistical report. It may be particular student or for the whole class. GUI 5: Help option should describe functionality of the system. It should be written in simple HTML.

4.2 Hardware Interface

Hardware Interface 1: The system should be embedded in the laptops.

Hardware Interface 2: The laptop should use wireless Ethernet Card to send departmental database server.

4.3 Software Interface:

Software Interface 1: Attendance maintenance system.

Software Interface 2: The attendance database should be transmitted to departmental database server. Software Interface 3: E-mail message generator which generates standard message of absence.

Software Interface 4: Report generators.

5. Performance Requirements

This system should work concurrently on multiple processors between college hours. The system should support 50 users.

The email should be send within one hour after class gets over.

The system should support taking attendance of maximum 10 students per class.

6. Design Constraints

The system should be designed within 6 months.

7. Other Non Functional Attributes

7.1 Security

The teacher should provide password to log on to the system. He/she should be able to see the record of his/her class.

7.2 Reliability

Due to wireless connectivity, reliability cannot be guaranteed.

7.3 Availability

The system should be available during college hours.

7.4 Maintainability

There should be a facility to add or delete or update teachers and students for each semester

7.5 Reusability

There should be a facility to add or delete or update teachers and students for each semester.

8. Operational Scenarios:

There will be student database, teacher database. The student database will contain students name, class, attendance, email address, address, and phone number.

The teacher database will contain teachers name, class taught, e-mail address, phone number.

9. Preliminary Schedule

The system has to be implemented within 6 months.

Experiment No-2

Aim: To Calculate Effort Estimation using COCOMO Model

Effort Estimation using COCOMO Model for Attendance Maintenance System

BASIC COCOMO MODEL:

The basic COCOMO model gives an approximate estimate of the project parameters. The basic COCOMO estimation model is given by the following expressions:

Effort = a_1x (KLOC) ^{a2} pm

Tdev = b_1x (Effort) ^{b2}

Months P= Effort/ Tdev

- Where KLOC is the estimated size of the software product expressed in Kilo Lines of Code, and P is the no of persons required to complete the work

- **a_1, a_2 , b_1, b_2** are constants for each category of software products,

- Tdev is the estimated time to develop the software, expressed in months,

- Effort is the total effort required to develop the software product, expressed in person-months (PMs).

The coefficients **a_1, a_2 , b_1, b_2** for various types of software projects

Software Projects	**a_1**	**a_2**	**b_1**	**b_2**
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Estimation of development effort for the three classes of software products, the formulas for estimating the effort based on the code size are shown below:

Organic : Effort = $2.4(KLOC)^{1.05}$ PM

Semi-detached: Effort = $3.0(KLOC)^{1.12}$ PM

Embedded : Effort = $3.6(KLOC)^{1.20}$ PM

Example:

Effort Calculation for Attendance Maintenance System Consider Lines of Code = 10000

i.e value of KLOC is 10

Organic : Effort = $2.4(KLOC)^{1.05}$ PM

$= 2.4*(10)^{1.05}$

$= 2.4 * 11.220$

$= 26.92$ pm

Semi-detached: Effort = $3.0(10)^{1.12}$ PM

$=3.0* 13.18$

$=39.5$ pm

Embedded : Effort = $3.6(10)^{1.20}$ PM

$= 3.6*15.84$

$= 57.02$

Experiment No-3

Aim: To Calculate Effort using FP oriented estimation model

Effort estimation using FP oriented estimation model for Attendance Maintenance System

FP Points Computation

To compute function points (FP), the following relationship is used:

FP = count total [0.65 + 0.01 Σ (F_i)] where count total is the sum of all FP entries .

The Fi (i = 1 to 14) are "complexity adjustment values" based on responses to the following questions :

1. Does the system require reliable backup and recovery?

2. Are data communications required?

3. Are there distributed processing functions?

4. Is performance critical?

5. Will the system run in an existing, heavily utilized operational environment?

6. Does the system require on-line data entry?

7. Does the on-line data entry require the input transaction to be built over multiple screens or operations?

8. Are the master files updated on-line?

9. Are the inputs, outputs, files, or inquiries complex?

10. Is the internal processing complex?

11. Is the code designed to be reusable?

12. Are conversion and installation included in the design?

13. Is the system designed for multiple installations in different organizations?

14. Is the application designed to facilitate change and ease of use by the user?

Each of these questions is answered using a scale that ranges from 0 (not important or applicable) to 5 (absolutely essential). The constant values in Equation and the weighting factors that are applied to information domain counts are determined empirically.

Once function points have been calculated, they are used in a manner analogous to LOC as a way to normalize measures for software productivity, quality, and other attributes:

Effort = FP/ Person-Month

Count Total can be obtained using the following table

Domain Characteristics	**Count**		**Weighting factor**			**Count**
			Simple	**Average**	**Complex**	
No of user Input		*	3	4	6	
No of user output		*	4	5	7	
No of user enquiries		*	3	4	6	
No of files		*	7	10	15	
No of external interfaces		*	5	7	10	
Count Total						

<u>Example</u>:

Assumptions

: Number of user input : 5

Number of user output : 5

Number of user enquiries

:6 Number of files :5

Number of external interfaces : 5

Apply these assumptions on a **simple project** and calculate the **Count Total**

Domain Characteristics	**Count**		**Weighting factor**			**Count**
			Simple	**Average**	**Complex**	
No of user Input	**5**	*	3	4	6	**15**
No of user output	**5**	*	4	5	7	**20**
No of user Enquiries	**6**	*	3	4	6	**18**
No of files	**5**	*	7	10	15	**35**
No of external Interfaces	**5**	*	5	7	10	**25**
No of external Interfaces	**5**	*	5	7	10	**25**
Count Total						**113**

Therefore Count Total =113

Now calculate the Functional Points using FP = count total [0.65 + 0.01 Σ (F_i)]

FP = count total [0.65 + 0.01 Σ (F_i)]

= 113 * (0.65 +0.01 * 25) where Σ (F_i) =25

i.e the questions answered using a scale that ranges from 0 (not important or applicable) to 5 (absolutely essential) in total 14 questions

=113 *(0.65 + 0.25)

=113 * 0.9 = 101.7

FP = 101.7

Effort = FP / person-month

Experiment No-4

Aim: To Analyze the Risk involved and Prepare

RMMM Plan RMMM Plan for Attendance Maintenance

System

Types Of Risks Involved

People risks are associated with the availability, skill level, and retention of the people on the development team.

Size risks are associated with the magnitude of the product and the product team. Larger products are generally more complex with more interactions. Larger teams are harder to coordinate.

Process risks are related to whether the team uses a defined, appropriate software development process and to whether the team members actually follow the process.

Technology risks are derived from the software or hardware technologies that are being used as part of the system being developed. Using new or emerging or complex technology increases the overall risk.

Tools risks, similar to technology risks, relate to the use, availability, and reliability of support software used by the development team, such as development environments and other Computer-Aided Software Engineering (CASE) tools.

Organizational and managerial risks are derived from the environment where the software is being developed. Some examples are the financial stability of the company and threats of company reorganization and the potential of the resultant loss of support by management due to a change in focus or a change in people.

Customer risks are derived from changes to the customer requirements, customers' lack of understanding of the impact of these changes, the process of managing these requirements changes, and the ability of the customer to communicate effectively with the team and to accurately convey the attributes of the desired product.

Sales and support risks involve the chances that the team builds a product that the sales force does not understand how to sell or that is difficult to correct, adapt, or enhance.

RMMM PLAN:

The RMMM plan is a document in which all the risk analysis activities are described. Sometimes project manager includes this document as a part of overall project plan. Sometimes specific RMMM Plan is not created, however each risk can be described individually using risk information sheet.

Typical template for RMMM Plan.

Risk Information sheet			
Project Name < Enter Name of the project for which risk has to be identified>			
Risk id < # >	**Date** <date at which risk is identified>	**Probability** <Risk Probability % >	**Impact** <low/medium/high>
Origin **<** The person who have identified the risk>		**Assigned to** <Who is responsible for mitigating the risk>	
Description <Description of risk identified>			
Refinement/Context <Associated information for risk refinement>			
Mitigation/ Monitoring <Enter the mitigation / Monitoring steps taken>			
Trigger/Contingency Plan <if Risk Mitigation fails then the pan for handling the risk>			
Status <Running status that provides a history of what is being done for the risk and changes in the risk. Include the date the status entry was made>			
Approval < Name & Signature of person approving closure>	**Closing Date** <Date>		

Example: RMMM Plan for Attendance Maintenance system

Risk Information sheet			
Project Name < Attendance Maintenance system>			
Risk id < 1 >	**Date** < 13/11/15 >	**Probability** < 20 % >	**Impact** <medium>
Origin < Yajat Surya>		**Assigned to** <Bhakti Shelar>	
Description <Summation of attendance is not working after each month>			
Refinement/Context <Update the software program and check for the summation function>			
Mitigation/ Monitoring <See that every module is in working condition or not>			
Trigger/Contingency Plan <Call the developer of the software and ask him/her to correct the module which is not working properly>			
Status <1. Contacted the developer of the software purchased : 15/11/15 2 Attendance is taken manually till the module gets worked : 17/11/15 3 Developer corrects the module : 18/11/15>			
Approval < Pradeep >		**Closing Date** <18/11/15>	

Experiment No-5

Aim: To Develop Time-line chart and project table using PERT

or CPM Project table using PERT or CPM for Attendance

Maintenance System

The program evaluation and review technique (PERT) chart is used to schedule, organize, and coordinate tasks within the project. The objective of PERT chart is to determine the critical path, which comprises critical activities that should be completed on schedule. This chart is prepared with the help of information generated in project planning activities, such as estimation effort, selection of suitable process model for software development, and decomposition of tasks into subtasks. The advantages of using PERT chart are:

- It represents the project in graphical form.
- It provides information about the expected completion time of the project.
- It describes the probability of completion of project before the specified date.
- It specifies the activities that form the critical path.
- It specifies the start and end dates of activities involved in project.
- It describes the dependencies of one or more tasks on each other.

STEPS FOR CREATING PERT CHART

- Identify activities and milestones.
- Identify sequence of activities
- Prepare PERT chart: In this step, the PERT chart is prepared.
- Estimate the time consumed in activities
- Determine critical path
- Update PERT chart

CPM (Critical Path Method)

Critical path method is a technique that determines those activities, which have the least scheduling flexibility (that is, critical activities). Note that if the critical activities are delayed, the entire project is delayed. After determining these activities, CPM specifies the project schedule according to the activities that lie on the critical path.

The advantages of using the critical path method are:

- It represents the project in graphical form.
- It predicts the time required to complete form.
- It specifies how to speed up the project so that it is completed on

schedule. It specifies the optimal plan for speeding up the project

Boundary time calculations

BTC can be very useful in software project scheduling. In the design of one function, for example, can retard further development of other functions. Here is a description boundary times used in the program.

- **EST (Earliest Start Time):** The earliest time that can begin when all preceding tasks are completed in the shortest possible time. The EST is the maximum of all paths from start to this task.
- **EFT (Earliest Finish Time):** It is the earliest time(EST) + duration of a task.
- **LST (Latest Start Time):** The latest time that a task can begin, is the difference between MT and maximum of all paths from this task to the finish.
- **LFT (Latest Finish Time):** It is the min (LST of all predecessors).
- **Slack (slack Time):** The amount of surplus time allowed in scheduling tasks but staying on schedule. It is

  LST-EST and equivalently can be written as **LFT-EFT.**

The MT (minimum time) to complete the project is the maximum of all path from start to finish.

- A critical path is one with a zero slack time.
- A path from the start node to the finish node containing only critical tasks is called path.

"Critical path is when all stack times=0."

<u>Example</u>:

STEPS FOR CREATING PERT CHART for ATTENDANCE MAINTENANCE SYSTEM

- Identify activities and milestones.

a) Specification of Requirements

b) Design Database

c) Design GUI

d) Code for Database

e) Code for GUI
f) Write user manual
g) Integrate and test

- Identify sequence of activities

a) Specification-Design database-code database-Integrate and test
b) Specification-Design GUI-Code GUI-Integrate and test
c) Specification-Write user manual
d) The above mentioned sequences can be performed simultaneously

- Prepare PERT chart
- Estimate the time consumed in

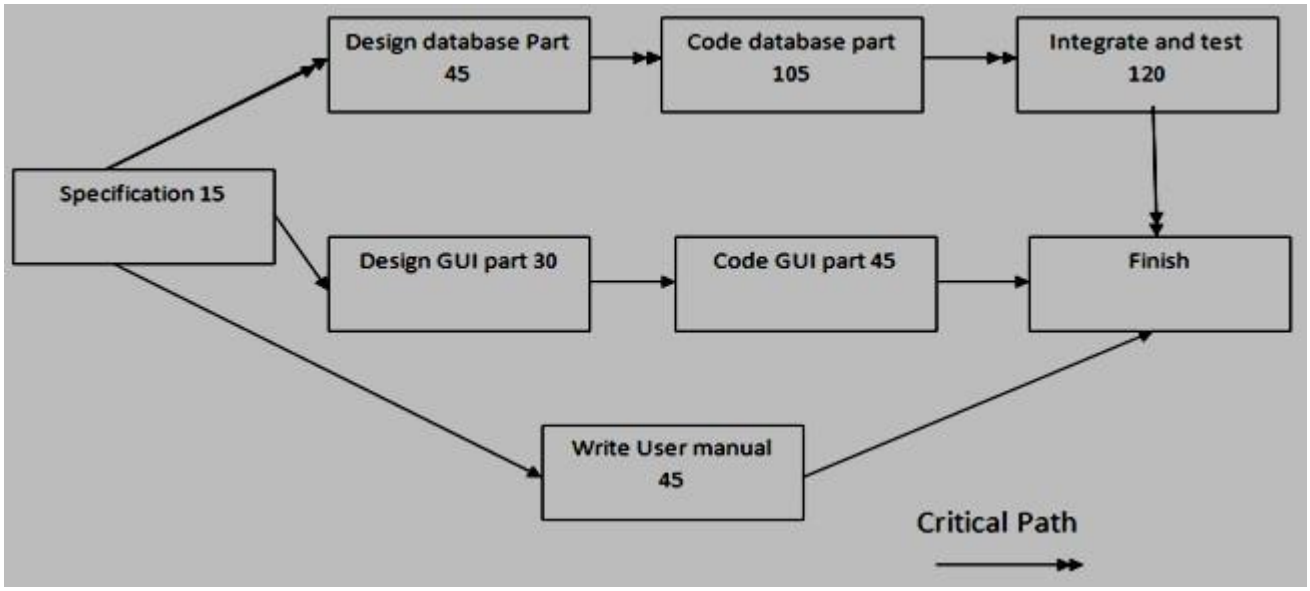
activities Assumptions are made for various
activities in days:

a) Specifications- 15 days
b) Design Database- 45 days
c) Design GUI-30 days
d) Code for Database- 105 days
e) Code for GUI-45 days
f) Write User Manual-45 days
g) Integrate and Test- 120 days

- Determine critical path

The longest path from start to finish in a PERT chart

Time-Line Chart



Critical Path : Specification→Design Database part→Code Database part→Integrate and Test→Finish

Project Table:

Tasks	EST	EFT	LST	LFT	ST
Specification Part	0	15	0	15	0
Design Database part	15	60	15	60	0
Design GUI part	15	45	90	120	75
Code Database part	60	165	60	165	0
Code GUI part	45	90	120	165	75
Integrate and Test	165	285	165	285	0
Write User-Manual	15	75	225	285	210

Experiment No-6

Aim: To Draw ER Diagrams and DFD's

ER Diagrams and DFD's for Attendance Maintenance System

ER Diagram

The ER model defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing

Entity

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

Attributes

Entities are represented by means of their properties, called **attributes**. All attributes have values. For example, a student entity may have name, class, and age as attributes.

There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative.

Data Flow Diagram

Data flow diagram is graphical representation of flow of data in an information system. It is capable of depicting incoming data flow, outgoing data flow and stored data. The DFD does not mention anything about how data flows through the system.

There is a prominent difference between DFD and Flowchart. The flowchart depicts flow of control in program modules. DFDs depict flow of data in the system at various levels. DFD does not contain any control or branch elements.

Types of DFD

Data Flow Diagrams are either Logical or Physical.

- **Logical DFD** - This type of DFD concentrates on the system process, and flow of data in the system.For example in a Banking software system, how data is moved between different entities.
- **Physical DFD** - This type of DFD shows how the data flow is actually implemented in the system. It is more specific and close to the implementation.

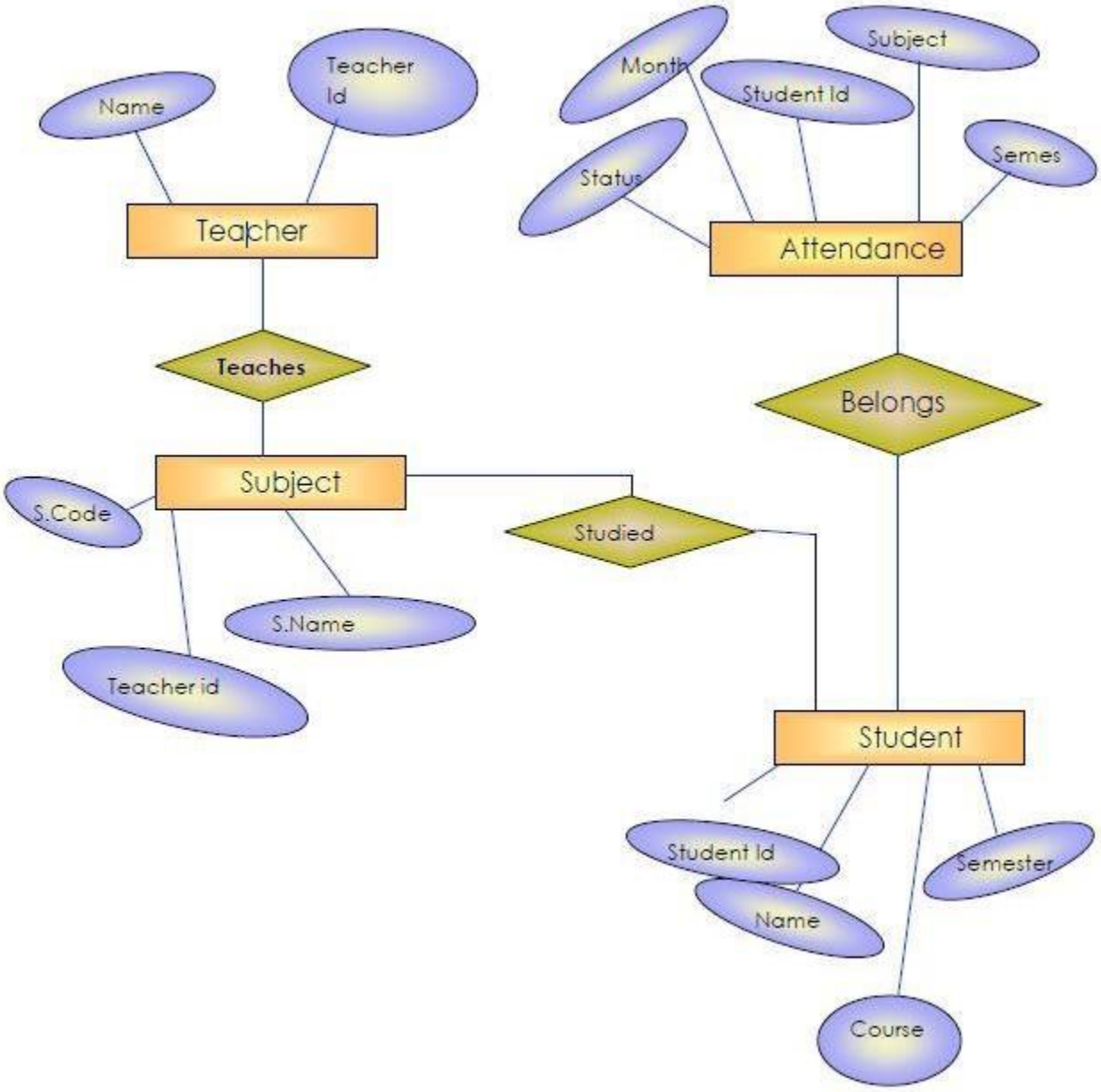
DFD Components

DFD can represent Source, destination, storage and flow of data using the following set of components -

- **Entities** - Entities are source and destination of information data. Entities are represented by a rectangles with their respective names.
- **Process** - Activities and action taken on the data are represented by Circle or Round-edged rectangles.
- **Data Storage** - There are two variants of data storage - it can either be represented as a rectangle with absence of both smaller sides or as an open-sided rectangle with only one side missing.
- **Data Flow** - Movement of data is shown by pointed arrows. Data movement is shown from the base of arrow as its source towards head of the arrow as destination.

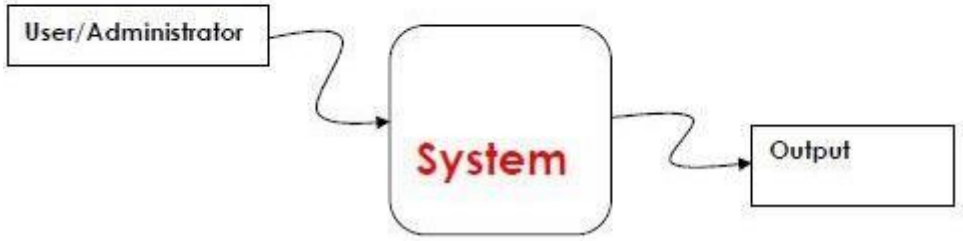
Example:

ER Diagram

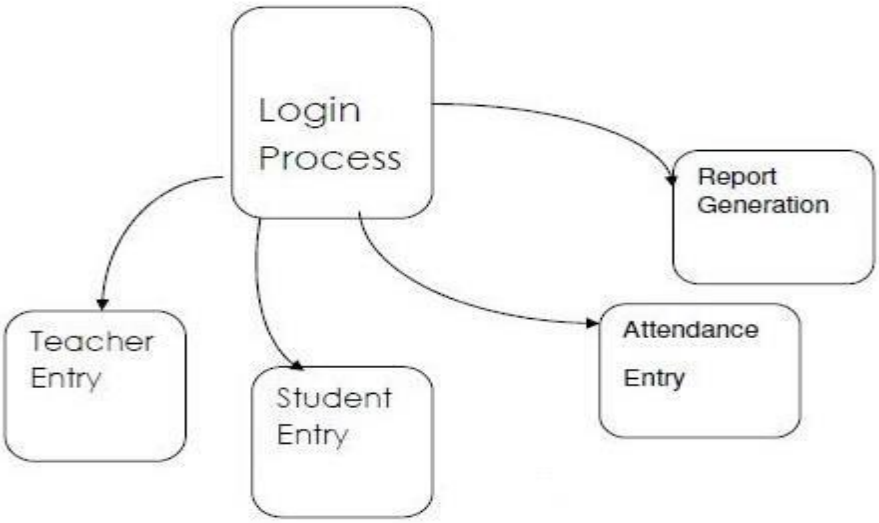


Data flow diagrams

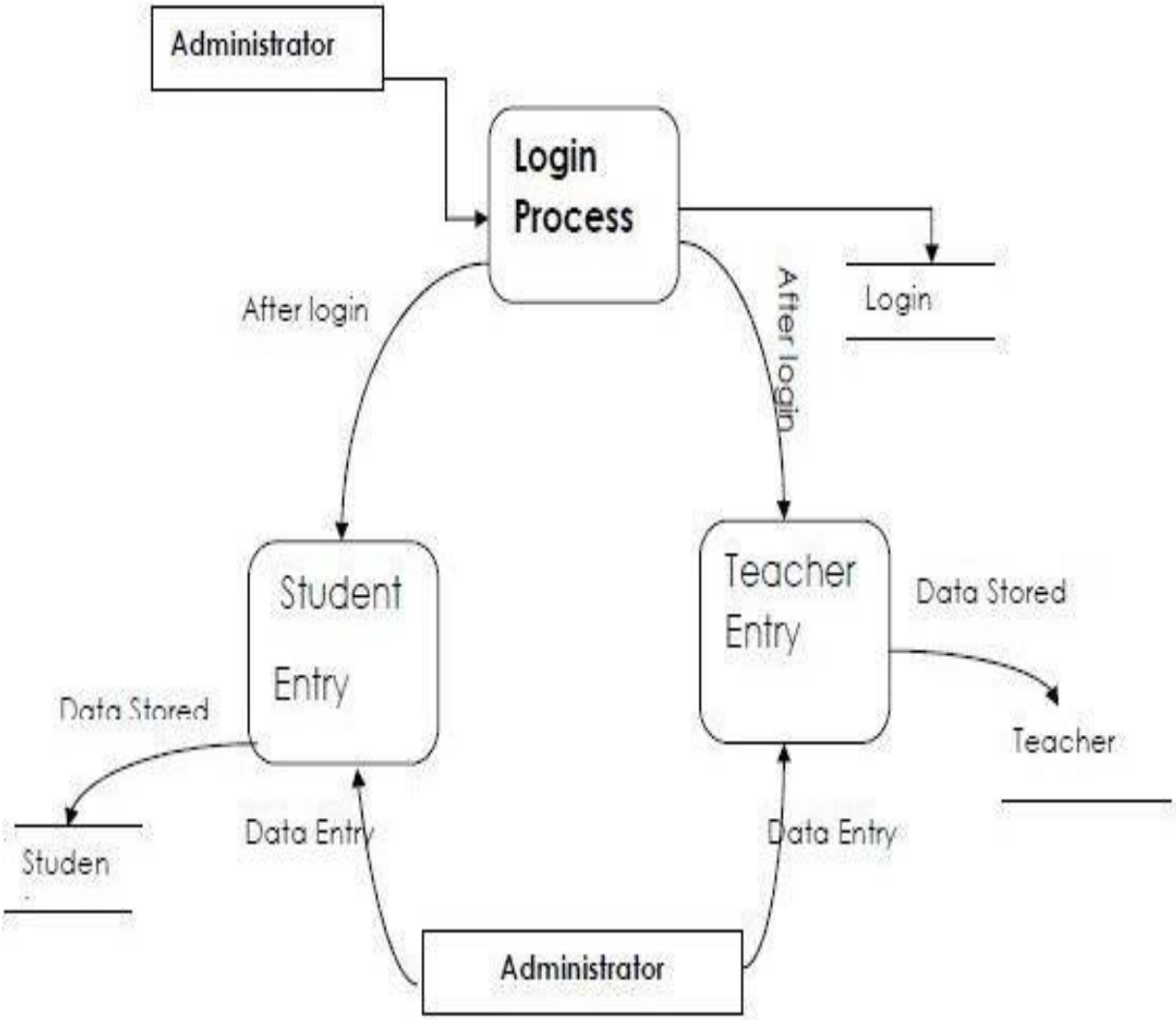
0-level DFD:



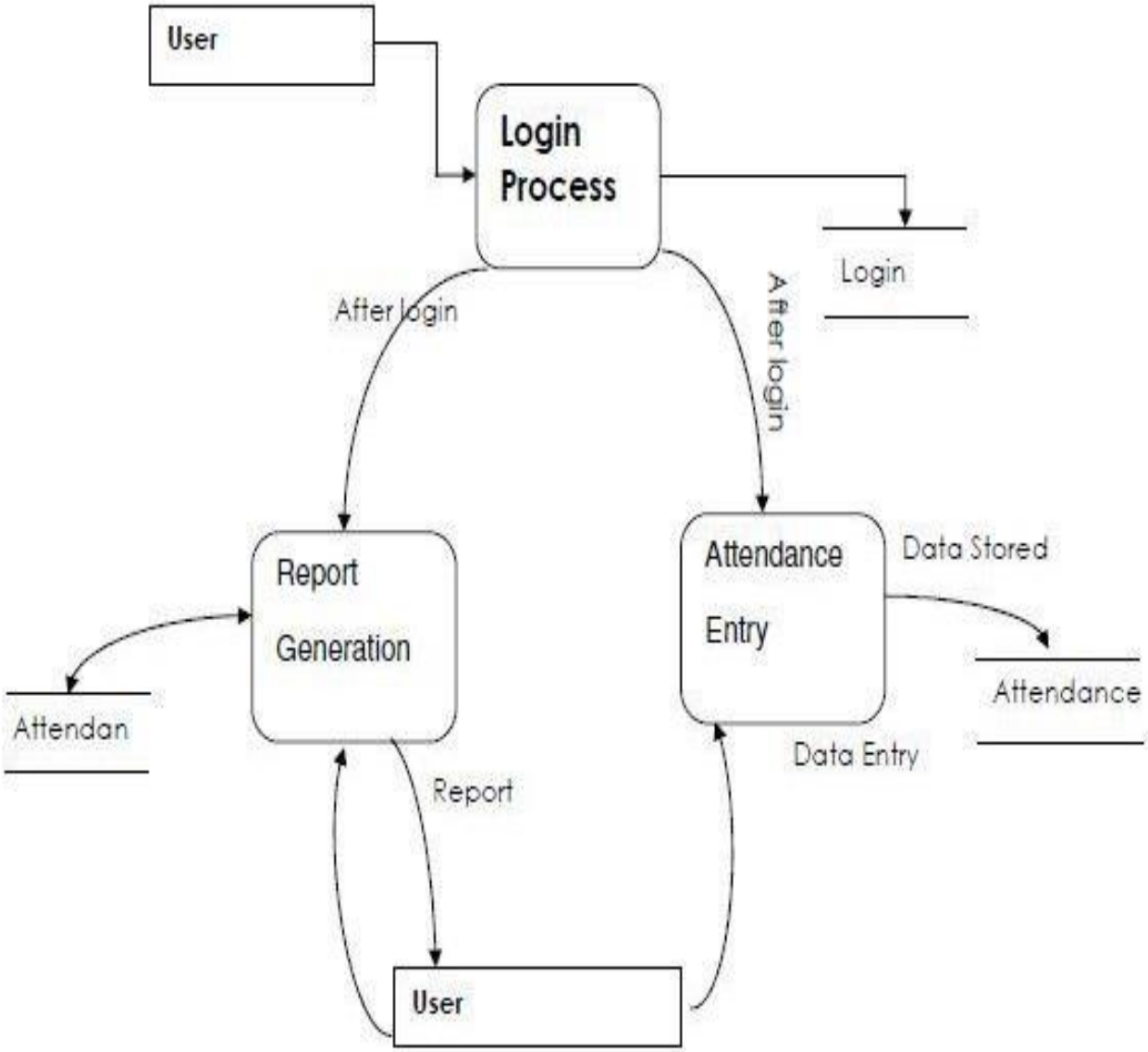
1-level DFD:



2.1:


Administrator
Login Process
Login
After login
After login
Student Entry
Teacher Entry
Data Stored
Data Stored
Teacher
Studen
Data Entry
Data Entry
Administrator

2.2


User
Login Process
Login
After login
After login
Report Generation
Attendance Entry
Data Stored
Attendance
Attendan
Data Entry
Report
User

Experiment No-7

Aim: To Design Test-Cases based on Requirements and Design

VARIOUS FILEDS IN A TEST CASE

TEST CASE FIELD	DESCRIPTION
Test case ID:	• Each test case should be represented by a unique ID. To indicate test types follow some convention like "TC_UI_1" indicating "User Interface Test Case#1."
Test Priority:	• It is useful while executing the test. o Low o Medium o High
Name of the Module:	• Determine the name of the main module or sub-module being tested
Test Designed by:	• Tester's Name
Date of test designed:	• Date when test was designed
Test Executed by:	• Who executed the test- tester
Date of the Test Execution:	• Date when test needs to be executed
Name or Test Title:	• Title of the test case
Description/Summary of Test:	• Determine the summary or test purpose in brief
Pre-condition:	• Any requirement that needs to be done before execution of this test case. To execute this test case list all pre-conditions
Dependencies:	• Determine any dependencies on test requirements or other test cases
Test Steps:	• Mention all the test steps in detail and write in the order in which it requires to be executed. While writing test steps ensure that you provide as much

detail as you can

Test Data:
- Use of test data as an input for the test case. Deliver different data sets with precise values to be used as an input

Expected Results:
- Mention the expected result including error or message that should appear on screen

Post-Condition:
- What would be the state of the system after running the test case?

Actual Result:
- After test execution, actual test result should be filled

Status (Fail/Pass):
- Mark this field as failed, if actual result is not as per the estimated result

Notes:
- If there are some special condition which is left in above field

Sample Test-Case:

Req ID	Risks	Requirement Type	Requirement Description	Trace from User Requirement/ Trace to System Requirement	Trace to Design Specification	UT	IT	ST	UAT	Trace to Test Script

Example: Test-Case for Attendance Management System based on Requirements and Design.

Req - Requirement (say)

Req id	**Risks**	**Req Type**	**Req Description**	**Trace from User Req/ Trace to system req**	**Trace to Design specifications**	**UT**	**IT**	**ST**	**UAT**	**Trace to test script**
3.2	Technical Aspect	Functional	Should perform	No	yes	Y	Y	Y	N	Y
4.1	GUI	Interface	Easy use	Yes	yes	Y	Y	N	N	Y
5.1	users login	Performance	So many users are not supported	No	No	N	N	N	N	Y
6	Design	Design	Overall design	Yes	Yes	Y	Y	Y	Y	Y
7.4	Updating Attendance	Maintainability	Add or Delete	No	yes	N	N	N	N	Y

Experiment No-8

Aim: To Prepare FTR (Formal Technical Review)

Document FTR for Attendance Maintenance System.

FTR is a formal review performed by a review team. It is performed in any phase of software development for any work product, which may include requirement specification, design document, code and test plan. Each FTR is conducted as a meeting and is considered successful only if it properly to planned, controlled, and attended. The objectives of FTR are:

1. To detect errors in functioning of software and errors occurring due to incorrect logic in software code.
2. To check whether the product being reviewed accomplished user requirements or not.
3. To ensure that a product is developed using established standards.

REVIEW GUIDELINES IN FTR:

- Review the product
- Set the agenda
- Keep track of Discussion.
- Indicate problem in the product
- Categorize the error
- Prepare notes
- Specify the number of people
- Develop a checklist

REVIEW MEETING:

A review meeting is conducted to review the product in order to validate its quality. Review team members examine the product to identify errors in it. As shown in Fig, a successful review consists of a number of stages.

Stages of Review Meeting

1. **Planning**. The Review team will check whether the work plan specified is properly working or not
2. **Overview:** In this stage the Attendance maintenance system is analyzed in order to detect errors. For this purpose, knowledge of the product is essential.
3. **Preparation:** In this stage, each review team members examines the Attendance maintenance system individually to detect and record problem

- **Completeness:** User requirements for Attendance maintenance are complete are checked here
- **Clarity:** Attendance maintenance requirements are understood properly
- **Consistency:** Names of data structures and functions are specified properly.
- **Feasibility:** Constraints such as time, resources and techniques are specified correctly.

4. **Meeting:** In this stage, the moderator reviews the agenda and issues related to the Attendance Maintenance system. The problems described in the overview stage are discussed among review team members. Then, the recorder records the problems in a defect list, which is used to detect and correct the errors later by the author. The defect list divided into the following categories:

- **Accept the product:** In this category, author accepts the product without the need for any further verification. This is because there are no such problems that halt the execution of the product.
- **Conditionally Accept the Product:** In this category, the author accepts the product, which requires verification. If there are any problems in the product, the next stage is followed
- **RE-examines the Product:** In this category, the author re-examines the product to understand the problem in it. After rework, the product is sent to the moderator again it verifies that problem are eliminated.

5. **Rework:** In this stage, the author revises the problem that is identified during review meeting. He determines the problem in the Attendance maintenance system and their causes with the help of a defect list. Then he resolves the problem in the product and brings it back to the moderator for the follow up stage.
6. **Follow-up:** In this stage, the Attendance maintenance system is verified after the author has performed rework on it. In case there are still errors in the product, a review meeting is conducted again.

Experiment No-9

Aim: To Prepare Version control and Change control for Software

configuration items Version control and Change control for Attendance

Maintenance System.

Software Configuration Management is a set of activities that have been developed to manage change throughout the life cycle of computer software. It is software quality assurance activity.

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

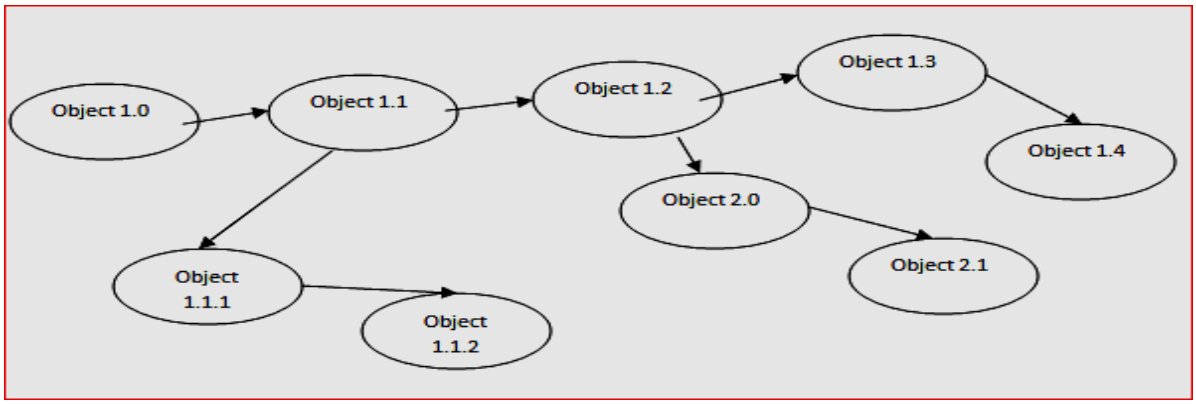
IEEE defines configuration item as-'an element of configuration item, consisting of selecting the configuration items for a system and recording their functional and physical characteristics in technical documentation.'

- Software Configuration items are not only program code segments but all type of documents according to development, e.g
    - all types of code files
    - drivers for tests
    - analysis or design documents
    - user or developer manuals
    - system configurations (e.g. version of compiler used)
- In some systems, not only software but also hardware configuration items (CPUs, bus speed frequencies) exist!
- Even a commercial product used in the system can be a configuration item.

SCM: VERSION CONTROL

- An initial release or re-release of a configuration item associated with a complete compilation or recompilation of the item. Different versions have different functionality.
- 1. Combines procedures and tool to manage difference version of configuration object that are created during software process.
- 2. One representation of different versions of a system is evaluation graph.

- 3. Each version of software is a collection of SCI, and each version may be composed of different variants.
- 4**.** Object Pool**:** It is a representation of components of SCI, and each version may be composed of different variants.



Each node on graph is an aggregate object i.e. a complete version of software.

SCM: CHANGE CONTROL

Change request, change report and engineering change order are generated as the part of the configuration control activity within the SCM Process. These documents are often represented as printed or electronic forms .the outline that follows is intended to provide an indication of the content of each. The actual structure of the SCM information may vary depending on its implementation. The software change report (SCR) is completed as consequence of a formal request for change of a base –lined SCI. The SCR is completed after the change request has been evaluated by software engineering staff

Parts of Software Change Report

- **Name , Identification and Description of SCI:** The name version/control Numbers of the SCI is specified, including page numbers if a document is involved.
- **Requester**: The name of the person requesting the change
- **Evaluator/Analyzer** : The name of the person who has evaluated the change request
- **Contact /information**: How to contact the requester and the Evaluator
- **Date/Location & Time**: When and where the change report was generated

Example:

Change Request Form			
Project: Attendance Maintenance system		**Number** :	19-10
Change Requester :	Rohini Yadav	**Date** :	11/11/15
Requested Change: When Monthly attendance component is selected from the structure, display the name of the file where it is stored			
Change Analyzer : Zigeesha Tendulkar			
Components Affected: Display-Icon. Select, Display-Icon. Display			
Associated Components: File Table where attendance is stored			
Change Assessments: Relatively simple to implement as a file name table is available. Requires the design and implementation of a display field. No changes to associated components are required.			
Change Priority: Medium			
Change Implementation: Based on Analyzer Comments and specifications			
Estimated Effort: 0.5 days			
Date to CCB :	1/12/15	**CCB decision date** :	16/01/16
CCB Decision: Accept change to be implemented in Release 2.1			
Change Implementer :	Srikanth Varma	**Date of Change** :	20/01/16
Date Submitted to QA :	22/01/16	**QA decision** :	Yes
Date Submitted to CM :	24/01/16		
Comments: The change was successfully made in the software			