SONU ILYAS MUGHAL (MIT DEPARTMENT 2ND SEMESTER)

Software engineering (CS504)

What is software engineering?

Software engineering is the process of analyzing user needs and designing, constructing, and testing end user applications that will satisfy these needs through the use of **software** programming languages. It is the application of **engineering** principles to **software development.**

2ND DEFINITION

Software engineering is not just concerned with the technical processes of software development but also with activities such as software project management and with the development of tools, methods and theories to support software production".

Why is software engineering important?

Software engineering is **important** because specific **software** is needed in almost every industry, in every business, and for every function. It becomes more **important** as time goes on – if something breaks within your application portfolio, a quick, efficient, and effective fix needs to happen as soon as possible.

Why **Study Software Engineering**? ... **Software Engineering** applies the knowledge and theoretical understanding gained through computer science to building high-quality **software** products. As a maturing discipline, **software** is becoming more and more important in our everyday lives.

What do software engineers do?
A **software engineer** applies mathematical analysis and the principles of computer science in order to design and develop computer **software**. ... When working with a client, a **software engineer** will typically analyze the client's needs, then design, test, and develop the computer **software** in order to meet those needs.

Why is software so important?
The operating system controls the basic functions of a computer or network. It's a **software** program that enables hardware to communicate and operate with the computer **software**. Microsoft (MSFT) is a leading player in this segment with 90% of personal computers using Windows as their operating system.

WHAT IS SOFTWEAR CRISIS?

Software crisis is a term used in the early days of computing science for the difficulty of writing useful and efficient computer programs in the required time. The software crisis was due to the rapid increases in computer power and the complexity of the problems that could not be tackled. With the increase in the complexity of the software, many software problems arose because existing methods were insufficient.

The term "software crisis" was coined by some attendees at the first NATO Software Engineering Conference in 1968 at Garmisch, Germany.[1][2] Edsger Dijkstra's 1972 ACM Turing Award Lecture makes reference to this same problem:[3]

The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.

According to handout?

In early 60s software had suffered from the similar kind of problem to which we call
Software Crisis. Techniques that were used to develop small software were not applicable

for large software systems. This thing resulted in the following consequences.

- In most of the cases that software which was tried to be build using those old tools
- and techniques were not complete.
    - Most of the times it was delivered too late.
    - Most of the projects were over-budgeted.
    - And in most of the case systems build using these techniques were not reliable –

meaning that they were not be able to do what they were expected to do. As a result of these problems a conference were held in 1960 in which the term software
crisis was introduced.

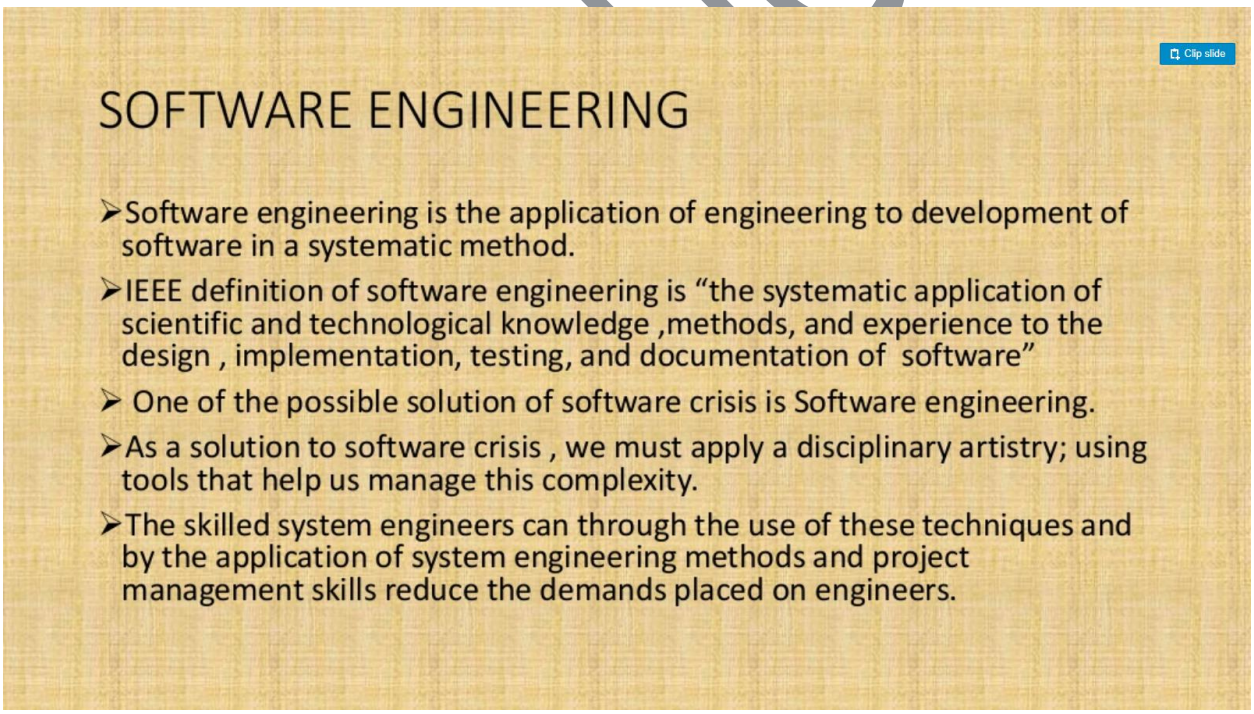
And the term of Software Engineering was also coined in the same conference.

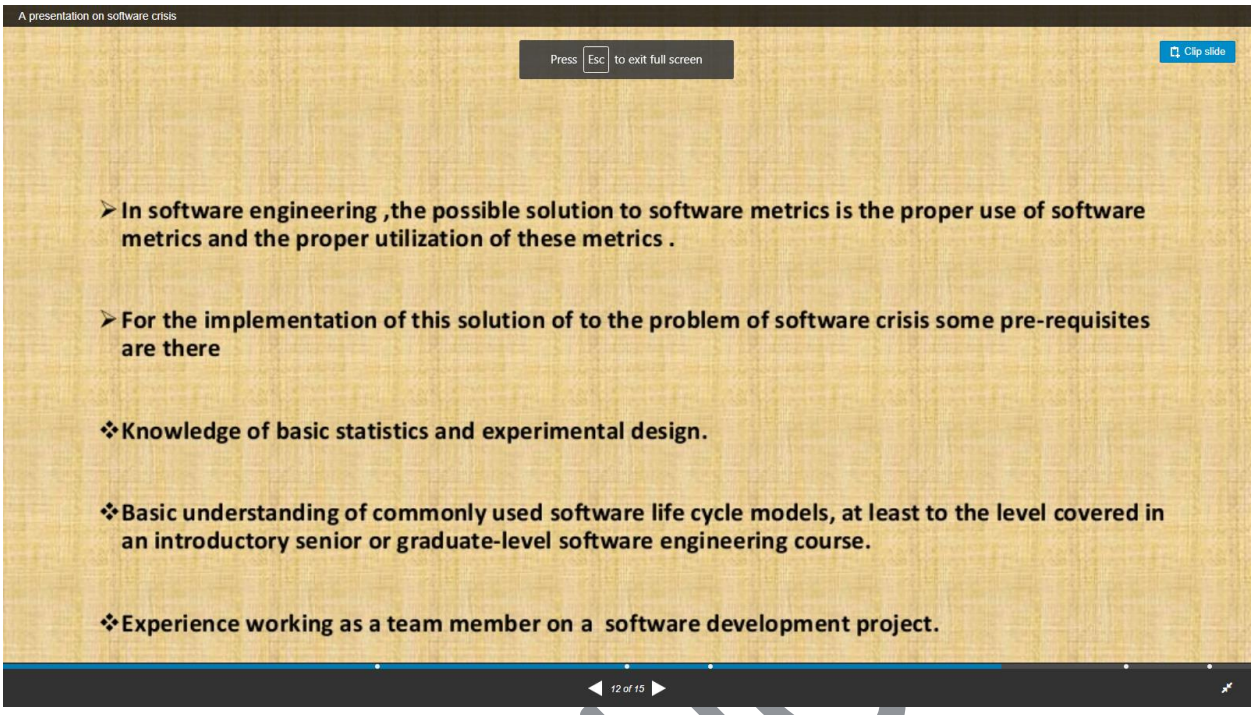
In the late 1960s, it became clear that the development of software is different from manufacturing other products. This is because employing more manpower (programmers) later in the software development does not always help speed up the development process. Instead, sometimes it may have negative impacts like delay in achieving the scheduled targets, degradation of software quality, etc. Though software has been an

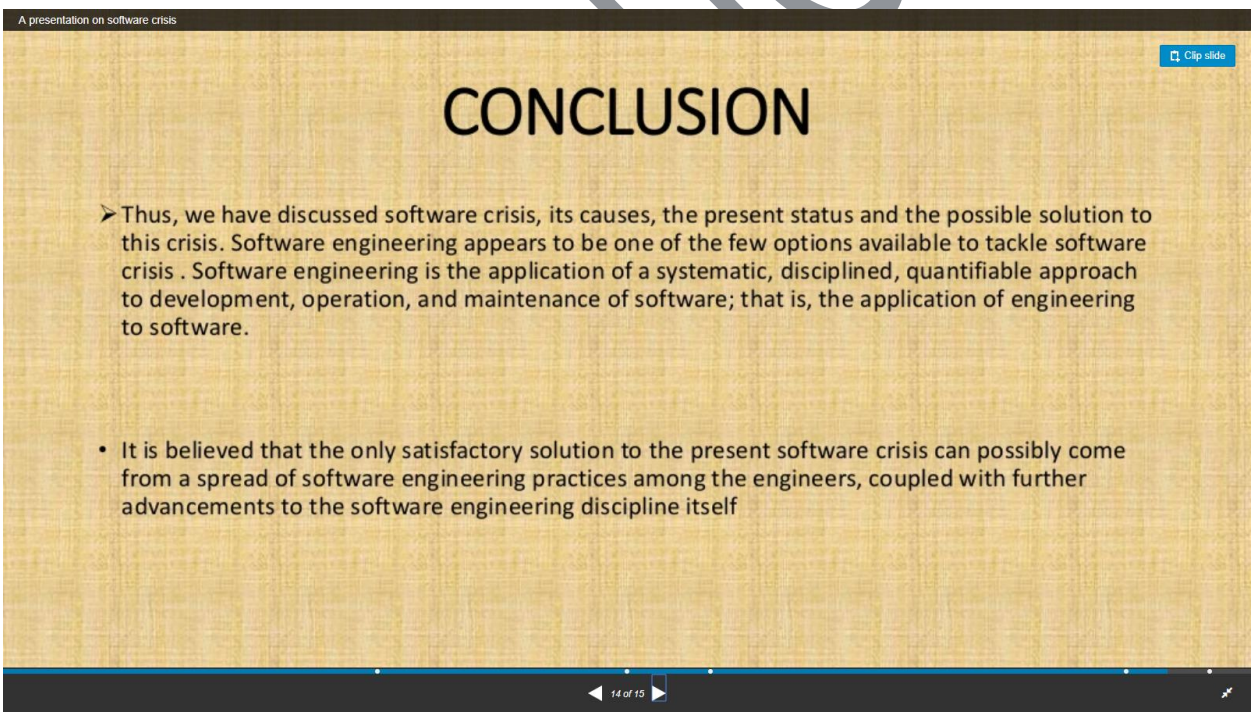
important element of many systems since a long time, developing software within a certain schedule and maintaining its quality is still difficult.

History has seen that delivering software after the scheduled date or with errors has caused large scale financial losses as well as inconvenience to many. Disasters such as the Y2Kproblem affected economic, political, and administrative systems of various countries around the world. This situation, where catastrophic failures have occurred, is known as **software crisis.** The major causes of software crisis are the problems associated with poor quality software such as malfunctioning of software systems, inefficient development of software, and the most important, dissatisfaction amongst the users of the software.

http://ecomputernotes.com/software-engineering/software-crisis\







What are software engineering principles?
Software Engineering Principles. **Software engineering principles**, when executed consistently and properly, ensure that

your **software** development process continually runs smoothly, efficiently and delivers high-quality applications.
Basically engineering comprises to planning, for doing the task efficiently and can say, Software engineering is the study and application of engineering to the design, development, and maintenance of software.

What is law of diminishing returns?

The **law of diminishing returns**, also referred to as the **law of diminishing** marginal **returns**, states that in a production process, as one input variable is increased, there will be a point at which the marginal per unit output will start to decrease, holding all other factors constant.

What is an example of law of diminishing returns?
The **law of diminishing** marginal **returns** states that, at some point, adding an additional factor of production results in smaller increases in output. For **example**, a factory employs workers to manufacture its products, and, at some point, the company operates at an optimal level.

FROM HANDOUT:
Whenever you perform any task like improving the efficiency of the system, try to improve its quality or user friendliness then all these things involve an
element of cost. If the quality of your system is not acceptable then with the investment
of little money it could be improved to a higher degree. But after reaching at a certain
level of quality the return on investment on the system's quality will become reduced.
Meaning that the return on investment on quality of software will be less than the effort
or money we invest. Therefore, in most of the cases, after reaching at a reasonable level
of quality we do not try to improve the quality of software any further.

benefit
cost

what are the major activities involved in the development of software.?

Known as the '**software development** life cycle,' these six steps include planning, analysis, design, **development** & implementation, testing & deployment and maintenance. Let's study each of these steps to know how the perfect **software** is developed.

1. **Planning**: Without the perfect plan, calculating the strengths and weaknesses of the project, development of software is meaningless. Planning kicks off a project flawlessly and affects its progress positively.

2. **Analysis**: This step is about analyzing the performance of the software at various stages and making notes on additional requirements. Analysis is very important to proceed further to the next step.

3. **Design**: Once the analysis is complete, the step of designing takes over, which is basically building the architecture of the project. This step helps remove possible flaws by setting a standard and attempting to stick to it.

4. **Development & Implementation**: The actual task of developing the software starts here with data recording going on in the background. Once the software is developed, the stage of implementation comes in where the product goes through a pilot study to see if it's functioning properly.

5. **Testing**: The testing stage assesses the software for errors and

documents bugs if there are any.

6. **Maintenance**: Once the software passes through all the stages without any issues, it is to undergo a maintenance process wherein it will be maintained and upgraded from time to time to adapt to changes. Almost every software development Indian company follows all the six steps, leading to the reputation that the country enjoys in the software market today.

Lecture 01 complete notes by SONU ILYAS MUGHAL

Lecture :02

Introduction to Software Development

What is Software development is a multi-activity process.?

Incremental Model is a process of software development where requirements are broken down into multiple standalone modules of software development cycle .**Incremental development** is done in steps from analysis design, **implementation**, **testing**/verification, **maintenance.**

What is software development process model?

In **software engineering**, a **software development process** is the **process** of dividing **software development** work into distinct phases to improve design, product management, and project management. ... For example, there are many specific **software development processes** that fit the spiral life-cycle **model**.

Software development process:

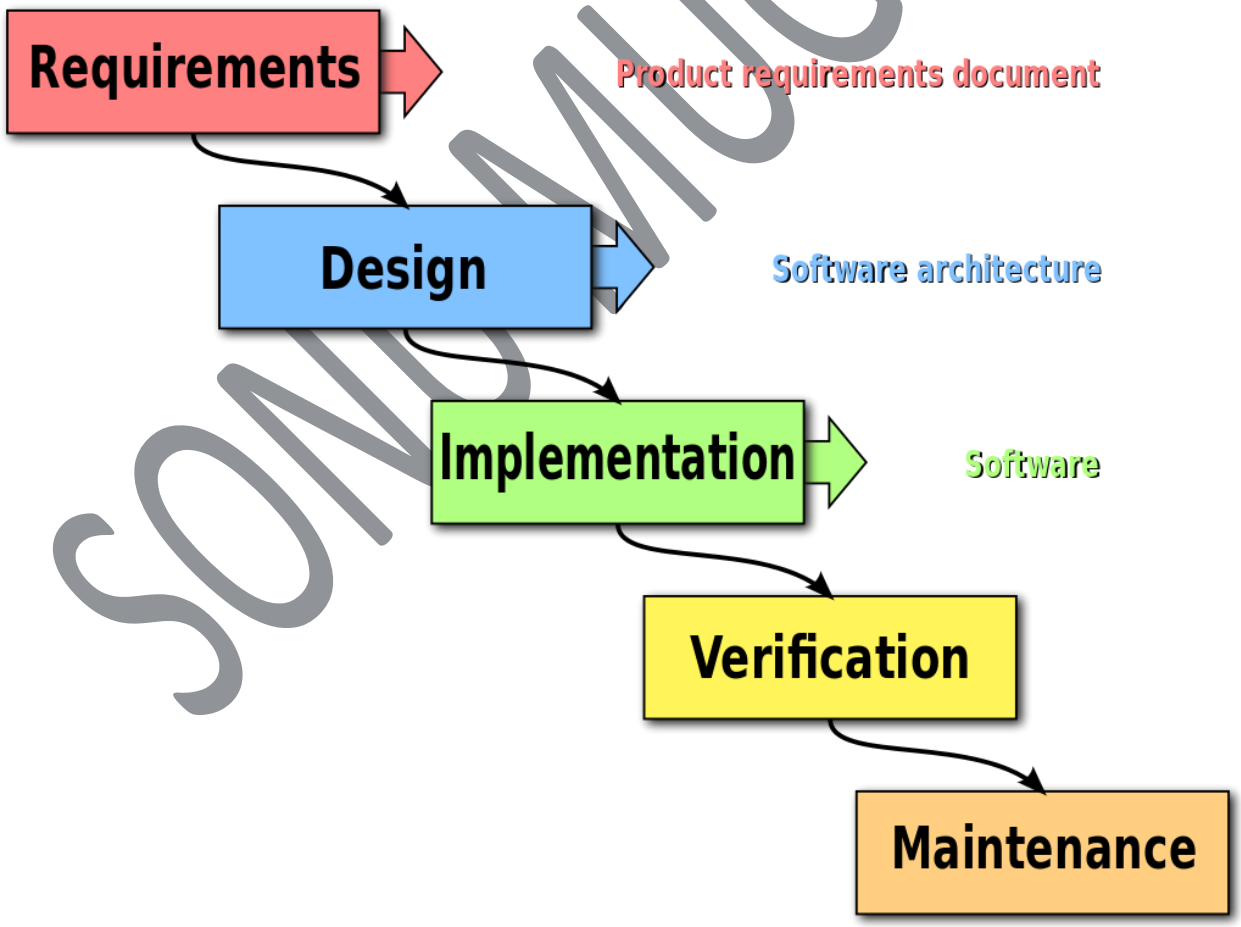
In **software engineering**, a **software development process** is the **process** of dividing **software development** work into distinct phases to improve design, product management, and project management. ... For example, there are many specific **software development processes** that fit the spiral life-cycle **model**.

What is software contruction and management ?

In general, **software construction** is mostly coding and debugging, but it also involves **construction** planning, detailed **design**, unit **testing**, integration **testing**, and other activities.

Maintenance

Correction, adaptation, enhancement



What is software framework ?

In **computer** programming, a **software framework** is an abstraction in which **software** providing generic functionality can be selectively changed by additional user-written code, thus providing application-specific**s oftware**. A **software framework** provides a standard way to build and deploy applications.

A **framework**, or **software framework**, is a platform for developing **software** applications. It provides a foundation on which **software** developers can build programs for a specific platform. ... A **framework** may also include code libraries, a compiler, and other programs used in the **software** development process.

What is software development loop?

The major stages of software development loop are described below:

•**Problem Definition**: In this stage we determine what is the problem agai nst which we are going to develop software. Here we try to completely c omprehend the issues and requirements of the software system to build.

•**Technical Development**: In this stage we try to find the solution of the pr oblem on technical grounds and base our actual implementation on it. Th is is the stage where a new system is actually developed that solves the pr oblem defined in the first stage.

Solution Integration: If there are already developed system(s) available with which our new system has to interact then those systems should also be the part of our new system. All those existing system(s) integrate with our new system at this stage.

Status Quo: After going through the previous three stages successfully, when we actually

deployed the new system at the user site then that situation is called status quo. But once

we get new requirements then we need to change the status qu After getting new requirements we perform all the steps in the software development loop again. The software developed through this process has the property that this could be evolved and integrated easily with the existing systems.

What is software development phases?

4 **Stages** of **Software** Development Process. ... Known as **software** development life cycle, these steps include vision, definition, development , maintenance.

Vision: Here we determine why are we doing this thing and what are our business
objectives that we want to achieve.
Definition: Here we actually realize or automate the vision developed in first phase. Here
we determine what are the activities and things involved.
Development: Here we determine, what should be the design of the system, how will it
be implemented and how to test it.
Maintenance: This is very important phase of software development. Here we control
the change in system, whether that change is in the form of enhancements or defect
removel.

What is importance of software maintenance ?

Maintaining a software is as important as software development. The **regular** maintenance of a system keeps it healthy by the time to deal with challenges and changes in the business environment.

According to stats, “when it comes to software, 60% of costing involves maintenance.

LECTURE 03

Requirement Engineering

What is Requirement Engineering in software development?

The **Requirement Engineering** (RE) is the most important phase of the **Software Development** Life Cycle (SDLC). This phase is used to translate the imprecise, incomplete needs and wishes of the potential users of **software** into complete, precise and formal specifications.

Phases of SDLC in software engineering?

We can divide the whole process in 4 distinct phases namely vision, definition, development, and maintenance.

During the vision phases, the focus is on why do we want to have this system;

during definition phase the focus shifts from why to what needs to be built to fulfill the previously outlined vision;

during development the definition is realized into design and implementation of the system;

and finally during maintenance all the changes and enhancements to keep the system up and running and

adapt to the new environment and needs are carried out.

Requirement engineering mainly deals with the definition phase of the system. Requirement engineering is the name of the process when the system services and constraints are established. It is the starting point of the development process with thefocus of activity on what and not how.

Software Requirements Definitions

Jones defines software requirements as a statement of needs by a user that triggers the development of a program or system

Alan Davis defines software requirements as a user need or necessary feature, function, or attribute of a system that can be sensed from a position external to the system.

According to Ian Summerville, requirements are a specification of what should be implemented. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system.

IEEE defines software requirements as:

1. A condition or capability needed by user to solve a problem or achieve an objective.

2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.

3. A documented representation of a condition or capability as in 1 or 2.

As can be seen, these definitions slightly differ from one another but essentially say the same thing: a software requirement is a document that describes all the services provided

by the system along with the constraints under which it must operate.

Fred Brooks in his classical book on software engineering and project management "The Mythical Man Month" emphasizes the importance of requirement engineering and writes:

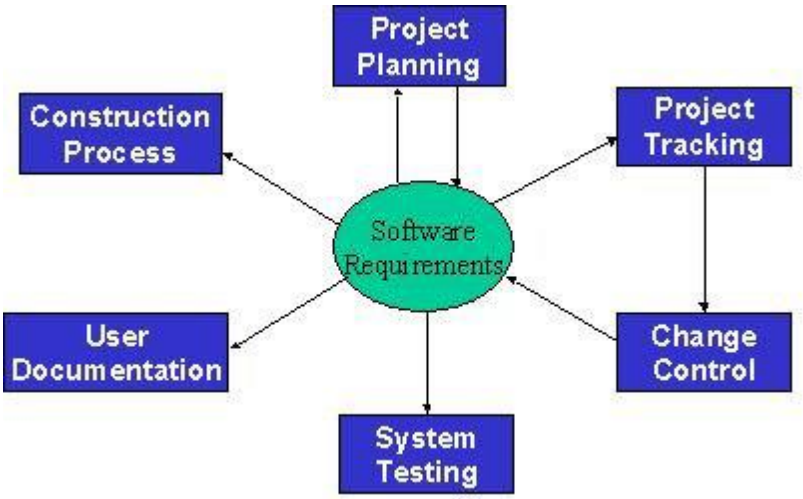
The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the work so cripples the system if done wrong. No other part is more difficult to rectify later.

Role of Requirements:

Software requirements document plays the central role in the entire software development process. To start with, it is needed in the project planning and feasibility phase. In this phase, a good understanding of the requirements is needed to determine the time and resources required to build the software. As a result of this analysis, the scope of the system may be reduced before embarking upon the software development.
Once these requirements have been finalized, the construction process starts. During this phase the software engineer starts designing and coding the software. Once again, the requirement document serves as the base reference document for these activities. It can be clearly seen that other activities such as user documentation and testing of the system would also need this document for their own deliverables.
On the other hand, the project manager would need this document to monitor and track

the progress of the project and if needed, change the project scope by modifying this
document through the change control process.

The following diagram depicts this central role of the software requirement document in
the entire development process.



Lecture 04

Requirement Engineering-2

Some Risks from Inadequate Requirement Process:

List are below :

Lecture 05

What is context diagram with example?

A **context diagram** is a **data flow diagram** that only shows the top level, otherwise known as Level 0. At this level, there is only one visible process node that represents the functions of a complete system in regards to how it interacts with external entities. ... Shows the overview of the boundaries of a system

What is a data flow diagram (DFD)?

A picture is worth a thousand words. A Data Flow Diagram (DFD) is traditional visual representation of the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or combination of both.

It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

It is usually beginning with a context diagram as the level 0 of DFD diagram, a simple representation of the whole system. To elaborate further from that, we drill down to a level 1 diagram with lower level functions decomposed from the major functions of the system. This could continue to evolve to become a level 2 diagram when further analysis is required. Progression to level 3, 4 and so on is possible but anything beyond level 3 is not very common. Please bear in mind that

the level of details for decomposing particular function really depending on the complexity that function.

DFD Diagram Notations

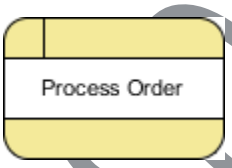
Now we'd like to briefly introduce to you a few diagram notations which you'll see in the tutorial below.

External Entity

An external entity can represent a human, system or subsystem. It is where certain data comes from or goes to. It is external to the system we study, in terms of the business process. For this reason, people used to draw external entities on the edge of a diagram.

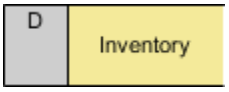
Process

A process is a business activity or function where the manipulation and transformation of data takes place. A process can be decomposed to finer level of details, for representing how data is being processed within the process.



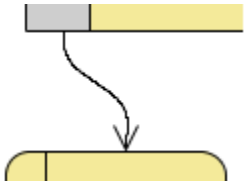
Data Store

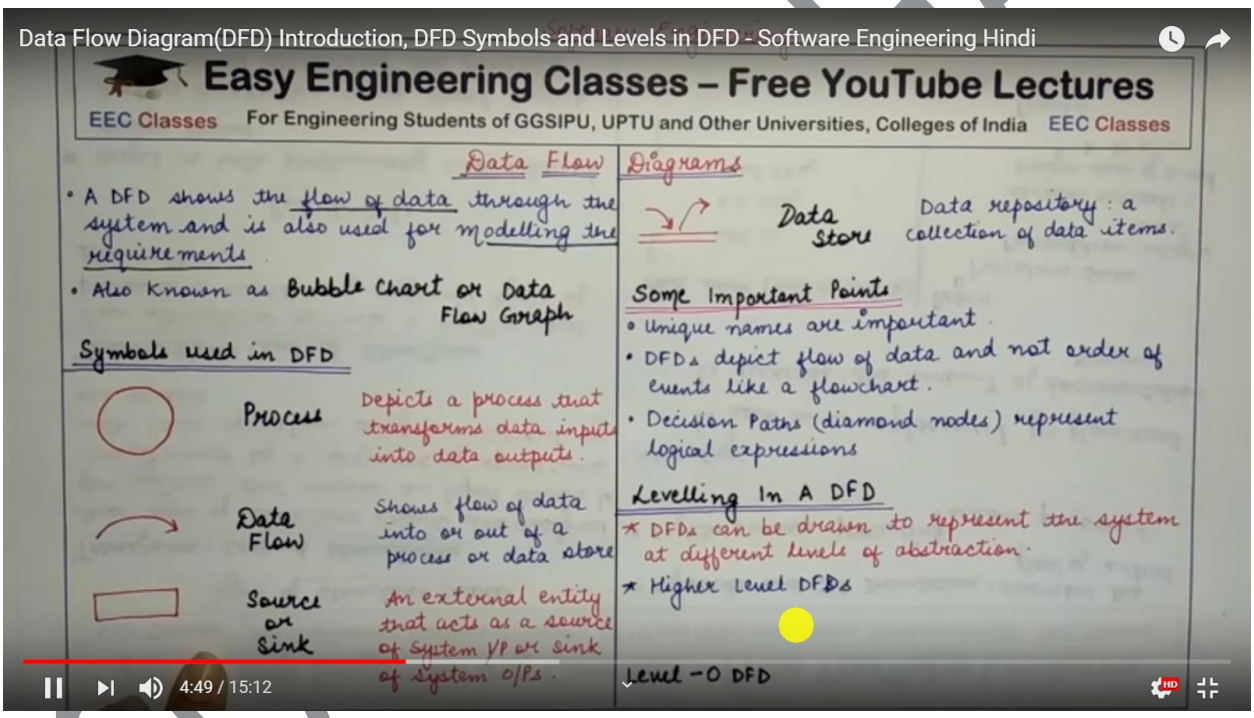
A data store represents the storage of persistent data required and/or produced by the process. Here are some examples of data stores: membership forms, database table, etc.



Data Flow

A data flow represents the flow of information, with its direction represented by an arrow head that shows at the end(s) of flow connector.





Non-functional requirements are those requirements which impose constraint on the system. They are sometimes called as “Quality attributes”. For example, attributes such as performance, security, usability, compatibility are not a feature or requirement of the system but are a required characteristic.

Dear Student,

ILF (Internal logic File) is a user identifiable group of data which is maintained with in the boundary of the application.

For more understanding about the following term visit the following link

http://www.informit.com/articles/article.aspx?p=19807&seqNum=5
http://www.softwaremetrics.com/FPLive/InternalLogicalFiles.pdf
https://people.eecs.ku.edu/~hossein/811/Papers/fpa-ref.pdf

Software Requirements – Definition – Jones

The statement of needs by a user that triggers the development of a program or system - *Jones 1994*

DEFINITION BY IEEE

1 A condition or capability needed by user to solve a problem or achieve an objective.

2 A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.

A documented representation of a condition or capability as in 1 or 2

DEFINITION BY JONE

The statement of needs by a user that triggers the development of a program or system - *Jones 199*

BY DAVIS

- A user need or necessary feature, function, or attribute of a system that can be sensed from a position external to that system - *Alan Davis 1993*

Requirements are ... A specification of what should be implemented. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system. - *Sommerville 199*

LEVEL OF REQUIRMENT

- **Business Requirements**

User will be able to correct spelling errors in a document efficiently and it will be integrated with the existing system

- **User Requirements**

Finding spelling errors in the document and decide whether to replace each misspelled word with the one chosen from a list of suggested words.

- **Functional Requirements**

2. Find and highlight misspelled words.

3. **Display a dialog box with suggested replacements.**
4. **Making global replacements.**

- **Non-Functional Requirements**

It must be integrated into the existing word-processor which runs on windows platform.

Ambiguous Requirements:

The operator identity consists of the operator name and password; the password consists of six digits. It should be displayed on the security VDU and deposited in the login file when an operator logs into the system."

Requirement Statement Characteristics

- **Complete -** Each requirement must fully describe the functionality to be delivered.

Correct - Each requirement must accurately describe the functionality to be built. A user requirement that conflict with a corresponding system requirement isn't correct

- **Feasible -** It must be possible to implement each requirement within the known capabilities and limitations of the system and its environment.
- **Necessary -**Each requirement should document something that the customer really need or something that is required for conformance to an external system requirement or standard.
- **Prioritized -** Assign an implementation priority to each requirement, feature or use case to indicate how essential it is to a particular product release.
- **Unambiguous -** All readers of a requirement statement should arrive at a single, consistent interpretation of it

- **Verifiable -** Examine each requirement to see whether you can devise a small number of tests or use other verification approaches, such as inspection or demonstration, to determine whether the requirement was properly implemented.

Mixed Level of Abstraction

The purpose of the system is to track the stock in the warehouse and might be intermixed with when the loading clerk types in the *withdraw* command he or she will communicate the order number, the identity of the item to be removed, and the quantity removed. The system will respond with a confirmation that the removal is allowable."

USE CASE MODEL

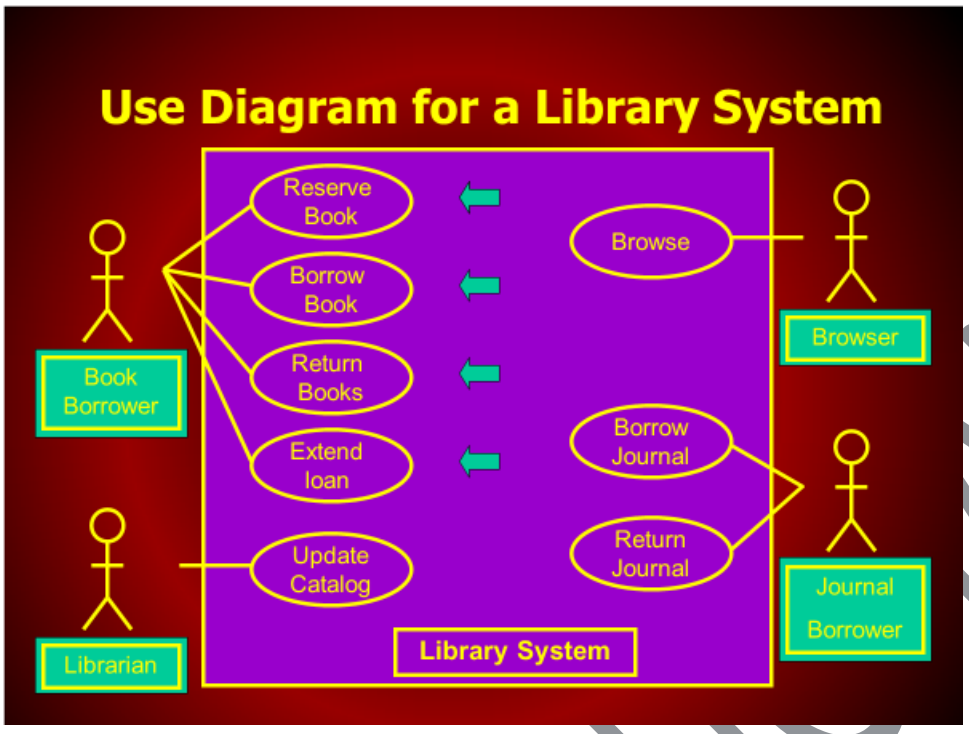
- **Use Case**

Boundaries of the system are defined by functionality that is handled by the system.

Each use case specifies a complete functionality (from its initiation by an actor until it has performed the requested functionality).

- **Actor**

An entity that has an interest in interacting with the system – a human or some other device or system.



COMPONENTS OF USE CASE

- **Priority**
- **Actor**
- **Summary**
- **Precondition**
- **Post- Condition**
- **Extend**
- **Normal Course of Events**
- **Alternative Path**
- **Exception**
- **Assumption**

ACTIVITY DIAGRAM

- **Animation sequence is on the slide.**
- **Use arrow for animation**
- **If it does not fit on one slide then u can use the scrolling feature. Take care show at least three levels at a time (not less in any case and then use the highlight feature.)**

7TH LECTURE

TYPES OF MODEL

- **Business Process Model**
- **State Transition Model**
- **Data Flow Model**

Captures the flow of data in a system.

.

It helps in developing an understanding of system's functionality.

.

What are the different sources of data, what different transformations take place

on data and what are final outputs generated by these transformations.

.

It describes data origination, transformations and consumption in a system.

.

Information is organized and disseminated at different levels of abstraction. Thus

this technique becomes a conduit for top down system analysis and requirements

modeling.

Data Functions: EIs, EOs and EQs

- **External Inputs**
- **External Outputs**
- **External InquirAy**

DFD versus Flow Charts

DFD versus Flow Charts

Flow charts are usually used to describe flow of control in a system. It describes control flow in an algorithm. Flow charts are quite detailed. Whereas DFD does not captures control flow information, it just shows the flow of the data in a system. Flow charts show the sequential activities of an algorithm. So, decisions are made, loops or iterations are described. On the other hand, DFD does not show the sequential activities. It just displays the business flow (without sequence among activities). As if you visit an organization, business activities are being performed in parallel. Therefore, DFD does not contain control or sequential activities just data transition is captured.

DFD	Flow Chart
• Processes on a data flow can operate in parallel. • Looping and branching are typically not shown. • Each process path may have a very different timing.	• Processes on flowcharts are sequential. • Show the sequence of steps as an algorithm and hence looping and branching are part of flowcharts.

Data Functions: EIs, EOs and EQs

- **External Inputs**
- **External Outputs**
- **External InquirAy**

- **External Inputs**

An external input (EI) is an elementary process that processes data or control information that comes from outside the application boundary. The primary intent of an EI is to maintain one or more ILFs and/or to alter the behavior of the system.

- **External Outputs**

An external output (EO) is an elementary process that sends data or control information outside the application boundary. The primary intent of an external output is to present information to a user through processing logic

- **External Inquiry**

An external inquiry (EQ) is an elementary process that sends data or control information outside the application boundary. The primary intent of an external inquiry is to present information to a user through the retrieval of data

Function	Transactional Function Type		
	EI	EO	EQ
Alter the behavior of the system	PI	M	N/A
Maintain one or more ILFs	PI	M	N/A
Present information to a user	M	PI	PI

Form of Processing Logic	Transactional Functional Type		
	EI	**EO**	**EQ**
1. Validations are performed	c	c	c
2. Mathematical Formula and calculations are performed	c	m*	n
3. Equivalent Values are converted	c	c	c
4. Data is filtered and selected by using specified criteria to compare multiple sets of data.	c	c	c
5. Conditions are analyzed to determine which are applicable	c	c	c
6. At least one ILF is updated	m*	m*	n
7. At least one ILF or EIF is referenced	c	c	m
8. Data or control information is retrieved	c	c	m
9. Derived data is created	c	m*	n
10. Behavior of system is altered	m*	m*	n
11. Prepare and present information outside the boundary	c	m	m
12. Capability to accept data or control information that enters the application boundary	m*	c	c
13. Resorting or rearranging a set of data	c	c	c

Tabular Method An Example

If the taxable income is less than Rs. 60,000, there will be no income tax. If the income exceeds Rs. 60,000 but is less than Rs. 150,000 then income tax will be charged at the rate of 7.5% for income exceeding Rs. 60,000. If the income exceeds Rs. 150,000

Income	**Tax**
Less than Rs. 60,000	**0%**

Between Rs. 60,000 and Rs. 150,000	**7.5% of (Income - 60,000)**
Between Rs. 150,000 and Rs. 300,000	**12.5% of (Income - 150,000) + 6,7**
Between Rs. 300,000 and Rs. 400,000	**20% of (Income - 300,000) + 25,5**
Between Rs. 400,000 and Rs. 700,000	**25% of (Income - 400,000) + 45,5**
Greater than Rs. 700,000	**35% of (Income - 700,000) + 120,**

FOR MID TERMS NOTES

Software Engineering

Lec#01 SUBJECTIVE

Q : What is software? (marks 2)

Ans. When we write a program for computer we named it as software. But software is not just
a program; many things other than the program are also included in software.

Q : How many items in software? Write name and explain . (marks 3)

Ans. ***Program*****:** The program or code itself is definitely included in the software.

Data**:** The data on which the program operates is also considered as part of the
software.

Documentation**:** Another very important thing that most of us forget is
documentation. All the documents related to the software are also considered as part
of the software.

Q : What is Engineering? (marks 2)

Ans. The process of productive use of scientific knowledge is called engineering.

Q : Difference between computer science and software engineering. (marks 5)

Ans. When we use physics in making machines like engines or cars then it is called mechanical
engineering. And when we apply the knowledge of physics in
-
developing electronic devices then the process is called electrical engineering. The relation of computer science with software engineering is similar as the relation of physics with the electrical, mechanical or civil engineering or for that matter the relation of any basic science with any engineering field. So,This is the process of utilizing our knowledge of computer science in effective production of software systems.

Q : Difference between Software and Other Systems. (marks 3)

Ans. If the software has a bug and that bug was present in the older CD then that will remain in the new one. This is a fundamental

difference between software and other systems.

Q : What is Software Crisis? (marks 3)

Ans. Software Crisis used to process the data of census. More powerful
hardware resulted into the development of more powerful and complex software. Those
very complex software was very difficult to write. So the tools and techniques that were
used for less complex software became inapplicable for the more complex software.

Q :What is software engineering as by IEEE? (marks 2)

Ans. The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. And the IEEE stands for International institute of Electric and Electronic Engineering.

-

Q : What is software engineering used in software production? (marks 5)

Ans. Software
Engineering is the combination of all the tools, techniques, and processes that used in software production.

- Programming Language
- Programming Language Design
- Software Design Techniques
- Tools
- Testing
- Maintenance
- Development

Q : Characteristics of well-Engineered software? (marks 3)

Ans.

- It is reliable
- It has good user-interface
- It has acceptable performance
- It is of good quality
- It is cost-effective

Q : What is Law of balancing act in software? (marks 5)
Ans.
Software Engineering is actually the balancing act. You have to balance many things like cost, user friendliness, Efficiency, Reliability etc. You have to analyze which one is the more important feature for your software is it reliability, efficiency, user friendliness or something
-
else. There is always a trade-off among all these requirements of software. It may be the case that if you try to make it more userfriendly then the efficiency may suffer. And if you try to make it more cost-effective then reliability may suffer. Therefore there is always a trade-off between these characteristics of software. These requirements may be conflicting

Lec#02 SUBJECTIVE

Q :.What is the Construction and major types of Construction? (marks 5)

Ans. The construction activities are those that directly related to the development of software,
e.g. gathering the requirements of the software, develop design, implement and test the software etc. Some of the major construction activities are listed below.

- Requirement Gathering
- Design Development
- Coding
- Testing

Q : Write the name of the major activities of management? (marks 3)

Ans.

- Project Planning and Management
- Configuration Management
- Software Quality Assurance
- Installation and Training

-

Q : Write the name of the major stage of software development loop? (marks 2)

Ans.
1. Problem Definition
2. Technical Development
3. Solution Integration
4. Status Quo

Q : How many software Engineering phase? and write the name. (marks 3)

Ans. There are four basic phase of S.E
1.Vision
2.Definition
3.Development
4.Maintenance

Lec#03 SUBJECTIVE

Q : Write the name of the role of software Requirement. (marks 3)

Ans.
1.Project Planning
2.project Tracking
3.Change Control
-
4.System Testing
5.User Dacumentation
6.Construction Process

Lec#04 SUBJECTIVE

Q : Write the name of the Requirement Statement Characteristics. (marks 3)

Ans.
1. Complete
2. Correct
3. Feasible
4. Necessary
5. Prioritized
6. Unambiguous
7. Verifiable

Q: What is Ambiguity? (marks 2)

ANS: Ambiguity means that two different readers of the same

document interpret the
requirement differently.

Q: What is Gold-Plating? (Marks 3)

Ans: These called cool features. Which are not added in requirements but developers add this. Gold-plating refers to features are not present in the original requirement document
and in fact are not important for the end-user but the developer adds
-
them anyway
thinking that they would add value to the product.

Q: Write the names of level of requirements: (Marks 3)

ANS:
1. Business Requirements
2. User Requirements
3. Functional Requirements
4. Non-Functional Requirements

Q: What is Business Scope? (Marks 2)

ANS: In which document we write business requirements these called business scope individual

Q: What is Requirement definition? (Marks 2)

Ans: In which document we write user requirements these called requirement definition

Q: What is function specification? (Marks 2)

Ans: In which document we write functional/nonfunctional requirements these called function specification

Q: Define Business requirement: (Marks 2)

User will be able to correct spelling errors in a document efficiently and is will be integrated with he existing system.

Q: Define User requirement: (Marks 2)

-
Finding spelling errors in the document and decide whether to replace each misspelled word with the one chosen from a list of suggested words.

Q: Define Function requirement: (Marks 3)

• Find and highlight misspelled words

• Display a dialogue box with suggested replacements.
• Making global replacements.

Q: Define Non functional requirement: (Marks 2)

It must be integrated into the existing word-processor which runs on windows platform.

Lec#05 SUBJECTIVE

Q : Write the Use Case Model Components? (marks 3)

Ans.

There are two use case model component
1.Cases = case specifies a complete functionality
2.Actors= An actor is an entity that has an
interest in interacting with the system. An actor can be a human or some other device or
system.

Q: What is scope? (marks 3)

-

ANs: Project scope defines the concept and range of the proposed solution, and limitations
identify certain capabilities that the product will not include. Clarifying the scope and
limitations helps to establish realistic stakeholder's expectations

Lec#06 SUBJECTIVE

Q :Write the name of Delete Information use case? (marks 2)

Ans. There are two existing use cases
1.Record Transaction
2.Cancel Transaction

Q : Write the name of Customer classes? (marks 2)

Ans. There are two classes.
1. Individual Customer
2. Corporate Customer

Q: What is Exception? (MARKS 2)

ANS: The system will not allow a user to delete information that is being used
in the system.

The system will not allow a user to delete another user that has subordinates.

Q: What is Assumption Act? (Marks 3)

ANS: Deleting information covers a permanent deletion of an entire set of data
such as a
commission plan, user, group etc. Deleting a portion of an entire set constitutes
modifying the set of data.

-

Deleted information is not retained in the system.

A user can only delete information that has not been used in the system.

Q: Write the names of limitations of use case? (Marks 2)

ANS:

- Usability
- Reliability
- Performance
- Portability
- Access

What is elaborated Use case? Explain it. (marks 3)

Answer:

After the derivation of the use case model, each use is elaborated by adding detail of interaction between the user and the software system. An elaborated use case has the following components: Use Case Name, actors, summary, precondition, post-condition, extend, uses, normal course of events, alternative path, exception, assumption.

Q: UML stands for…..?

ANS: Unified Modeling Language

Lec#07 SUBJECTIVE

Q: What is Source? (Marks 3)

Ans: Sources of requirements are the origins from where the corresponding
business process is initiated. By this concept, one has to trace from a requirement back to its origins to see who is involved in its initiation.

Q: What is Sink? (Marks 3)

-

ANS: Sink is the consumer of certain information. It is that entity which provides a logical end to a business process. Thus, ‘sinks of requirements’ is a
concept that helps in identifying persons, organizations or external systems
that gets certain functionality from the system.

Q.Write the name Techniques of Logical System Modals? (marks 3)

Ans.

1. User business processes
2. User activities for conducting the business processes
3. Processes that need to be automated
4. Processes which are not to be automated

Q : What is Business process model? (marks 2)

Ans. The first model that we will look at is called the process model. This model provides a high-level pictorial view of the business process. This model
can be used as a starting point in giving the basic orientation to the reader of
the document.

Q: There are three models used write their names: (marks 2)

ANs:

- Business Model
- State transition Model
- Data flow Model

Lec#08 SUBEJCTIVE

Q: What is STD? (Marks 2)

STD stands for State transition diagrams. This is another technique to document domain knowledge. This is an easy technique to design a work flow
application.

-

Q : What Types of International Function Point User’s Group (IFPUG)?
(marks 2)

Ans. There are three types
1. External Inputs
2. External Outputs
3. External Inquiry

Q : What is a data flow model? And explain it. (marks 5)

Ans.

Captures the flow of data in a system.
It helps in developing an understanding of system's functionality.
What are the different sources of data, what different transformations take place
on data and what are final outputs generated by these transformations.
It describes data origination, transformations and consumption in a system.
Information is organized and disseminated at different levels of abstraction.
Thus
this technique becomes a conduit for top down system analysis and requirements
modeling.

Q : Write the types of shapes of notation. And explain it. (marks 5)

Ans. There are four shapes of notations

1. **Process**

What are different processes or work to be done in the system.
Transforms of data.

2. **External Agent**

External systems which are outside the boundary of this system. These are represented
using the squares

3. **Data Store**

-

Where data is being stored for later retrieval.
Provides input to the process
Outputs of the processes may be going into these data stores.

4. **Data Flow**

Where the data is flowing.

Represents the movement of the data in a data flow diagram.

Q: What is External Inputs? (Marks 2)

Ans: An external input (EI) is an elementary process that processes data or
control information
that comes from outside the application boundary.

Q: What is External Query? (Marks 3)

ANS: An external output (EO) is an elementary process that sends data or control information
outside the application boundary. The primary intent of an external output is to
present
information to a user through processing logic other than, or in addition to, the
retrieval
of data or control information.

Q : What is the difference between DFD and Flow Chart. (marks 5)

Ans.

DFD

- Processes on a data flow can operate in parallel.
- Looping and branching are typically not shown.
- Each process path may have a very Different timing.

Flow Chart

- Processes on flowcharts are sequential.

-

- Show the sequence of steps as an algorithm and hence looping and branching are part of flowcharts.

Lec#09 SUBJECTIVE

Q : Write the name of CRUD Operations? And explain it. (marks 3)

Ans.

These are four operations

Create: creates data and stores it.

Read: retrieves the stored data for viewing.

Update: makes changes in an stored data.

Delete: deletes an already stored data permanently.

Q : What is Common Mistakes in Data Flow Diagrams? (marks 2)
Ans.
 There is no input for the process Freeze Member Account
 In a similar manner, the process Create a New Member Account does not
produce any output.

Lec#10 SUBJECTIVE

Q : What is GUI or Graphical user interface? (marks 2)
Ans. Graphical user interface is a computer interface that allows user to interact with a device through graphical elements such as pictures and animations, as opposed to text-based commands.

Q : What is Motivation for GUI? (marks 3)
Ans.
System users often judge a system by its interface rather than its functionality
-
A poorly designed interface can cause a user to make catastrophic errors
Poor user interface design is the reason why so many software systems are
never
Used

Q : Write the three types of Pitfalls of using GUIs in Functional Specifications. (marks 3)
Ans.
UIs distract from business process understanding (what) to interfacing details
(how)
Unstable requirements cause frequent modifications in UIs
An extra work to be done at the requirement level each time a GUI change has to
be incorporated

Q :. What is Prototype? (marks 2)
Ans. Prototyping is yet another technique that can be used to reduce customer

dissatisfaction at the requirement stage. A prototype is not the real product. It
is rather just a real looking mock-up of what would be eventually delivered
and might not do anything useful.

Cs 504 lec no 11

Q1: what parameters are used to measure and analyze design quality? 5 marks

Answer:- (Page 71) A software design can be looked at from different angles and different parameters
can be used to measure and analyze its quality. These parameters include efficiency, compactness,
reusability, and maintainability. A good design from one angle may not seem to be suitable when looked
from a different perspective. For example, a design that yields efficient and compact code may not be
very easy to maintain. In order to establish whether a particular design is good or not, we therefore have to
look at the project and application requirements

-

Question No: 2 What should be consideration for maintain design? (Marks: 5)

Answer:- (Page 71) In order to make a design that is maintainable, it should be understandable and the
changes should be local in effect. That is, it should be such that a change in some part of the system
should not affect other parts of the system. This is achieved by applying the principles of modularity,
abstraction, and separation of concern. If applied properly, these principles yield a design that is said to be
more cohesive and loosely coupled and thus is easy to maintain.

Question No: 3 It is fact that good design makes maintenance easier. Which design principle help this to be achieved? (Marks: 3)

Answer:- (Page 71) A good design from one angle may not seem to be suitable when looked from a
different perspective. For example, a design that yields efficient and compact code may not be very easy
to maintain. In order to establish whether a particular design is good or not, we therefore have to look at
the project and application requirements.

Q 4 :To manage the complexity of the system we need to apply the principles of separation of concern. Discuss
briefly 2 MARKS

Answer: (Page 69) Separation of concern allows us to deal with different individual aspects of a problem by
considering these aspects in isolation and independent of each other. A complex system may be divided
into smaller pieces of lesser complexity called modules.

Q5: DEFINE Software Design Qualities ?

Ans : A software design can be looked at from different angles and different parameters can be used to
measure and analyze its quality. These parameters include efficiency, compactness, reusability, and
maintainability.

Q6 : what is data modeling ? 2 marks

Ans : s. Data modeling is an essential activity performed during the design phase. This includes the
identification of data entities and their attributes, relationships among these entities, and the appropriate
data structures for managing this data.

Lecture no 12

Q1:Define abstraction? 2 marks

-

Answer:- (Page 79) An abstraction is a technique in which we construct a model of an entity based upon
its essential characteristics and ignore the inessential details.

Question No: 2 (Marks: 3) To manage the complexity of the system we need to apply the principle of abstraction. Discuss briefly?

Answer:- (Page 79) An abstraction is a technique in which we construct a model of an entity based upon
its essential characteristics and ignore the inessential details. The principle of abstraction also helps us in
handling the inherent complexity of a system by allowing us to look at its important external characteristic,
at the same time, hiding its inner complexity. Hiding the internal details is called encapsulation.

Q 3 :What are architectural designs Process, explain briefly? 5 Marks

Answer:- (Page 79) System structuring: - System structuring is concerned with decomposing the system into
interacting sub-systems. The system is decomposed into several principal sub-systems and communications
between these sub-systems are identified. Control modeling:- Control modeling establishes a model of the
control relationships between the different parts of the system. Modular decomposition:- During this activity,
the identified sub-systems are decomposed into modules. This design process is further elaborated in the
following section where architectural views are discussed.

Q4:Define cohesion 2 marks

. Answer: (Page 72) Cohesion is an internal property of a module. Cohesion describes the intra-component
linkages while couple shows the inter-component linkages. Cohesion measures the independence of a
module

Q 5:Define coupling ? 2 marks

Ans : Coupling is a measure of independence of a module or component. Loose coupling means that
different system components have loose or less reliance upon each other.

Q 6:What is encapsulation? 2 marks

Ans : Hiding the internal details is called encapsulation

Question No: 7 (Marks: 5) What is action-oriented approach for Software Design?

Answer: (Page 80) In the case of action-oriented approach, data is decomposed according to
functionality requirements. That is, decomposition revolves around function. In the OO approach,
decomposition of a problem revolves around data. Action-oriented paradigm focuses only on the
functionality of a system and typically ignores the data until it is required. Object- oriented paradigm
focuses both on the functionality and the data at the same time. The basic difference between these two
is decentralized control mechanism versus centralized control mechanism respectively. Decentralization
gives OO the ability to handle essential complexity better than action-oriented approach.

Lecture no 13

Question No: 1 (Marks: 3) HOW DO YOU DETERMINE THAT AN OBJECTIVE BELONGS TO CERTAIN CLASS?

Answer:- (Page 85) The basic unit of object oriented design is an object. An object can be defined as a
tangible entity that exhibits some well defined behavior. The structure and behavior of similar objects are
defined in their common class. A class specifies an interface and defines an implementation.

Q2: What is behavior driven perceptive of an objective? 3 marks

Answer:- (Page 85) Behavior is how an object acts and reacts in terms of its state changes and message
passing. The behavior of an object is completely defined by its actions. A message is some action that one
object performs upon another in order to elicit a reaction. The operations that clients may perform upon an
object are called methods

Q3: What is the difference between Aggregation and Association? 3 marks

Answer:- (Page 87) As compared to association, aggregation implies a tighter coupling between the two
objects which are involved in this relationship. Therefore, one way to differentiate between aggregation
and association is that if the two objects are tightly coupled, that is, if they cannot exist independently, it is
an aggregation, and if they are usually considered as independent, it is an association.

Q4 : define Relationship Among Objects ? 2 marks

Ans:The object model presents a static view of the system and illustrates how different objects collaborate with
one another through patterns of interaction. Inheritance, association and aggregation are the three interobject
relationships specified by the object model.

Q5 :Define the object model?

Ans:The elements of object oriented design collectively are called the Object Model. The object model
encompasses the principles of abstraction, encapsulation, and hierarchy or
inheritance.

Question No: 25 (Marks: 5) Code example of High Coupling

Answer: Click here for detail Tightly Coupled Example: public class CartEntry { public float Price; public int
Quantity; } public class CartContents { public CartEntry[] items; } public class Order { private CartContents
cart; private float salesTax; public Order(CartContents cart, float salesTax) { this.cart = cart; this.salesTax =
salesTax; } public float OrderTotal() { float cartTotal = 0; for (int i = 0; i < cart.items.Length; i++) { cartTotal +=
cart.items[i].Price * cart.items[i].Quantity; } cartTotal += cartTotal*salesTax; return cartTotal; } }

Lecture no 14

Q1: What is Textual Analysis? Explain it 3 marks

Answer:- (Page 90) Textual analysis was developed by Abbot and then extended by Graham and others. In
this technique different parts of speech are identified within the text of the specification and these parts
are modeled using different components.

Q2 : name the four layers of the OO design pyramid ? 2 marks

The four layers of the OO design pyramid are:
1) The subsystem layer.
2) The class and object layer
3) The message layer
4) The responsibility layer

Q 3: define problem statement? 3 marks

A simple cash register has a display, an electronic wire with a plug, and a numeric keypad, which has keys
for subtotal, tax, and total. This cash storage device has a total key, which triggers the release on the
drawer..

Q4: Explain the four layer of the OO design pyramid ? 5 marks

1) The subsystem layer. Contains a representation of each of the subsystems that enable the software to
achieve its customers defined requirements and to implement the technical infrastructure that supports
customer requirements.
2) The class and object layer. Contains the class hierarchies that enable the system to be created using
generalization and increasingly more targeted specializations. The layer also contains design
representations for each object.
3) The message layer. Contains the details that enable each object to communicate with its collaborators.

This layer establishes the external and internal interfaces for the system
. 4) The responsibility layer. Contains the data structures and algorithmic design for all attributes and
operations for each object.

Lecture no 16

No: 1 (Marks: 5) How the objects are identified in peter codd's technique?

Answer: (Page 93) Objects are identifying in the following way.

Select actors : Actors are people and organizations that take part in the system under
consideration. Examples of actors are: person, organization (agency, company, corporation,
foundation)

. **Select Participants** A participant is a role that each actor plays in the system under
consideration. Examples of participants are: agent, applicant, buyer, cashier, clerk,
customer, dealer, and distributor. Etc.

Select Places : Places are where things come to rest or places that contain other objects.
Examples of places are: airport, assembly-line, bank, city, clinic, country, depot, garage and
hospital etc

. **Select Transactions :** Transactions are the "events". These transactions usually come from a
window (GUI), some object which monitors for significant event and logs that information, or
a another system that interacts with the system under consideration and logs some
information. Examples of transactions are: agreement, assignment, authorization, contract,
delivery, deposit, incident, inquiry, order, payment, problem report, purchase and sales etc.

Select Container Objects Containers are objects that hold other objects. e.g. bin, box,
cabinet, folder, locker, safe, shelf, etc. Therefore a place is also a container but every
container need not be a place.
Select Tangible things Take a "walk" through the system and select "tangible" things around
you used in the problem domain. These may be characterized as all the remaining (not yet
selected) "nouns" that make up the problem domain. Examples are: account, book,
calendar, cash box, cash drawer, item, plan, procedure, product, schedule, skill, tool, etc

Lecture no 17
Q1: what is Identify Structures?
Ans : A structure is a manner of organization which expresses a semantically strong
organization within the problem domain.

Q2: how many types of Identify Structures?
Ans : There are two type of structures
Generalization-Specialization (Gen-Spec) and whole-part.

Q3 : Define Attributes - ?
Ans : The first two activities would identify most of the objects (classes) in the problem
domain. Now is the time to think about the role and responsibilities of these objects. The
first thing to consider is their attributes,

Q4 : Show Collaborations (associations and aggregations) Who I know? 5 marks
The second step in establishing each object's responsibility is to identify and show how

this object collaborates with other objects, i.e., who it knows. These collaborations can be
identified with the help of the following steps:
1. For an actor, include an object connect to its participants (association).
2. For a participant, include an object connection to its actor (already established) and
its
transactions (association).
3. For a location, include object connections to objects that it can hold (association), to
its part objects (aggregation), and to the transactions that are taking place at that
location (association).
4. For transactions, include object connections to its participants (already established),
its line items (aggregation), and its immediate subsequent transaction (aggregation).
5. For a transaction line item, include object connections to its transaction (already
established), its item (association), a companion

Q5 : GIVE EXAMPLE ARE ATTRIBUTES ?
Ans : Examples of attributes are: number, name,
address, date, time, operational state, phone, status, threshold, type, etc

lecture no 19
Q 1:Keeping in mind the Connie's case study, what rule of thumbs was identified,
list them down ? 5 marks
Ans :
Who I Know - Rules of
Thumb
1 :
an actor knows about its participants person knows
about cashier

2 :
a transaction knows about its participants a session knows about its register and cashier
3
A transaction contains its transaction line items sale contains its sales line items
4
A transaction knows its sub transactions session knows about its sales sale knows
about its payments
5
-
A place knows about its transactions store knows about its sessions
6
A place knows about its descriptive objects store knows about its tax categories
7
A container knows about its contents a store knows about its cashiers, items, and registers

Q2:Keeping connie's case study in mind, as discussed in lecture, list down whole parts structures
which were identified. Answer: (Page 100) 3 marks
Identify Whole-Part Structures
1 :
1 :A store as a whole is made up of cashiers,
registers, and items.
2 :PA register contains a
cash drawer.
3: A sale is constituted of sale
line items.

Q3 : Define data flow diagram ? 2 marks

Answer: (Page 100) A data flow diagram (DFD) is a graphical representation of the "flow" of
data through an information system, modeling its process aspects. Often they are a
preliminary step used to create an overview of the system which can later be elaborated.

Lecture no 20 :

Q1: Why we use series of diagram? 3 marks

A series of diagrams can be used to describe the dynamic behavior of an object-oriented
-
system. This is done in terms of a set of messages exchanged among a set of objects
within a context to accomplish a purpose.

Q2 : Define the purpose of Interaction diagrams is to ? 3 marks

The purpose of Interaction diagrams is to:
Model interactions between objects
Assist in understanding how a system (a use case) actually works
Verify that a use case description can be supported by the existing classes
Identify responsibilities/operations and assign them to classes

Q3 : Define life line of objects ? 2 marks

Ans The boxes denote objects (or classes), the solid lines depict messages being sent from one
object to the other in the direction of the arrow, and the dotted lines are called life-lines of
objects..

Q4 : write the syntax used for naming objects in a sequence diagram ? 3 marks

The syntax used for naming objects in a sequence diagram is as follows:
syntax: [instanceName][:className]
Name classes consistently with your class diagram (same classes).

Include instance names when objects are referred to in messages or when several

Lecture No 21:

(Q) How Many Types of messages are there?

Ans. There Are four type of messages are there.

(1) Synchronous
(2) Asynchronous
(3) Create
(4) Destroy

-

Q No: 2. what is Synchronous Messages?

Synchronous messaging describes communications that takes place between two applications or systems, where the system places a message in

a message queue (also called an Event Queue in enterprise messaging systems) and then waits for a message response before it continues processing. Contrast with asynchronous messaging. They are denoted by the full arrow.

Synchronous messaging is also known as synchronous communication.

(Q3) What is Asynchronous Messages?

Ans. Asynchronous messages are "signals," denoted by a half arrow. They do not block the caller.

Asynchronous messages typically perform the following actions:

Create a new threat.
Create a new object.
Communication with threat is already
running.

Q4. Object Creation and Destruction

Ans. An object may create another object via a <<create>> message. Similarly an object may

Destroy another object via a <<destroy>> message. An object may also destroy itself. One

Should avoid modeling object destruction unless memory management is critical.

Q: No 5. Collaboration Diagrams depict Dynamic behavior of the system, explain it.

Ans.

Collaboration diagrams can also be used to depict the dynamic behavior of a system. They show how objects interact with respect to organizational units (boundaries!). Since a boundary shapes communication between system and outside world e.g. user interface or other system, collaboration diagrams can be used to show this aspect of the system. The sequence of messages determined by numbering such as 1, 2, 3, 4, This shows which operation calls which other operation.

Collaboration diagrams have basically two types of components: objects and messages.

Objects exchange messages among each-other. Collaboration diagrams can also show synchronous, asynchronous, create, and destroy message using the same notation as used In sequence diagrams. Messages are numbered and can have loops

Q: No 6. Evaluating the Quality of an Object-Oriented Design.

Ans. Judging the quality of a design is difficult. We can however look at certain object-Oriented design attributes to estimate its quality. The idea is to analyze the basic principle Of encapsulation and delegation to judge whether the control is centralized or distributed, Hence judging the coupling and cohesion in a design. This will tell us how maintainable a Design is.

You may also recall our earlier discussion of coupling and cohesion. It can be easy to see That OO design yield more cohesive and loosely coupled systems.

Lecture No 22:

Q: No 1. What is Software Architecture?

Ans. Software architecture refers to the fundamental structures of a software system, the discipline of creating such
structures, and the documentation of these structures. These structures are needed to reason about the software
system. Each structure comprises software elements, relations among them, and properties of both elements and
relations,[1] along with rationale for the introduction and configuration of each element. The architecture of a software
system is a metaphor, analogous to the architecture of a building.

Q: No 2. Why is architecture important?

Ans. If a project has not achieved a system architecture, including its rationale, the Project should not proceed to full-scale system development. Specifying the Architecture as a deliverable enables its use throughout the development and Maintenance process.

Q: 3. No Architectural design process

Ans. Just like any other design activity, design of software architecture is a creative and
Iterative process. This involves performing a number of activities, not necessarily in any
Particular order or sequence. These include system structuring, control modeling, and
Modular decomposition.

Q: No 4. Architectural Attributes

Ans. Software architecture must address the non-functional as well as the functional Requirements of the software system. This includes performance, security, safety, Availability, and maintainability.

Q: No 5. What Is Performance in Architectural Design?
Ans. – Performance can be enhanced by localizing operations to minimize sub-System communication. That is, try to have self-contained modules as Much as possible so that inter-module communication is minimized.

Q: No 6. What is Security in Architectural Design?
Ans. – Security can be improved by using a layered architecture with critical Assets put in inner layers.
Q: No. 7. what is Safety in Architectural design
Ans. – Safety-critical components should be isolated
Q: No what is Availability in Architectural design
Ans. – Availability can be ensured by building redundancy in the system and Having redundant components in the architecture.

Q: No 8. What is Maintainability in Architectural design?
Ans. – Maintainability is directly related with simplicity. Therefore, Maintainability can be increased by using fine-grain, self-contained Components.

Q: No 9. What are architectural designs Process, explain briefly?
Ans. Just like any other design activity, design of software architecture is a creative and
Iterative process. This involves performing a number of activities, not necessarily in any

Particular order or sequence. These include system structuring, control modeling, and
Modular decomposition.
System structuring: - System structuring is concerned with decomposing the system into interacting
sub-systems. The system is decomposed into several principal sub-systems and communications
between these sub-systems are identified.

Control modeling: - Control modeling establishes a model of the control relationships between the
different parts of the system.
Modular decomposition: - During this activity, the identified sub-systems are decomposed into
modules. This design process is further elaborated in the following section where architectural views
are discussed.

Q: No 10. Differentiate between architectural design and system architecture in a single line?
Ans. Architecture faces towards strategy, structure and purpose, towards the abstract while Design
faces towards implementation and practice, towards the concrete.