Software Engineering

MODULE-3: Architectural Design:

Complied by : Ms. Mithila Satam
mithila.satam@vsit.edu.in

VSIT Vidyalankar School of Information Technology
NAAC Accredited College

Vidyalankar School of
Information Technology
Wadala (E), Mumbai
www.vsit.edu.in

VSIT | Vidyalankar School of Information Technology

NAAC Accredited College

Certificate

This is to certify that the e-book titled "SOFTWARE ENGINEERING" comprises all elementary learning tools for a better understating of the relevant concepts. This e-book is comprehensively compiled as per the predefined eight parameters and guidelines.

Date:20-11-2019

[illegible]

Mrs. Mithila Satam
Assistant Professor
Department of IT

Unit III Architectural Design

Contents:

1. Architectural Design
2. Decisions System Organisation
3. Modular Decomposition Styles
4. Control Styles, Reference Architectures.
5. Need of UI design
6. Design issues
7. The UI design Process
8. User analysis
9. User Interface Prototyping
10. Interface Evaluation.
11. Software Project Management
12. Management activities
13. Project Planning
14. Project Scheduling
15. Risk Management.
16. Process and Product Quality
17. Quality assurance and Standards
18. Quality Planning
19. Quality Control
20. Software Measurement and Metrics.

- **Recommended Books:**

1. Software Engineering, edition, Ian Somerville Pearson Education. Ninth
2. Software Engineering, Pankaj Jalote Narosa Publication
3. Software engineering, a practitioner's approach , Roger Pressman , Tata Mcgraw-hill , Seventh
4. Software Engineering principles and practice, WS Jawadekar , Tata Mcgraw-hill .
5. Software Engineering-A Concise Study, S.A Kelkar , PHI India.

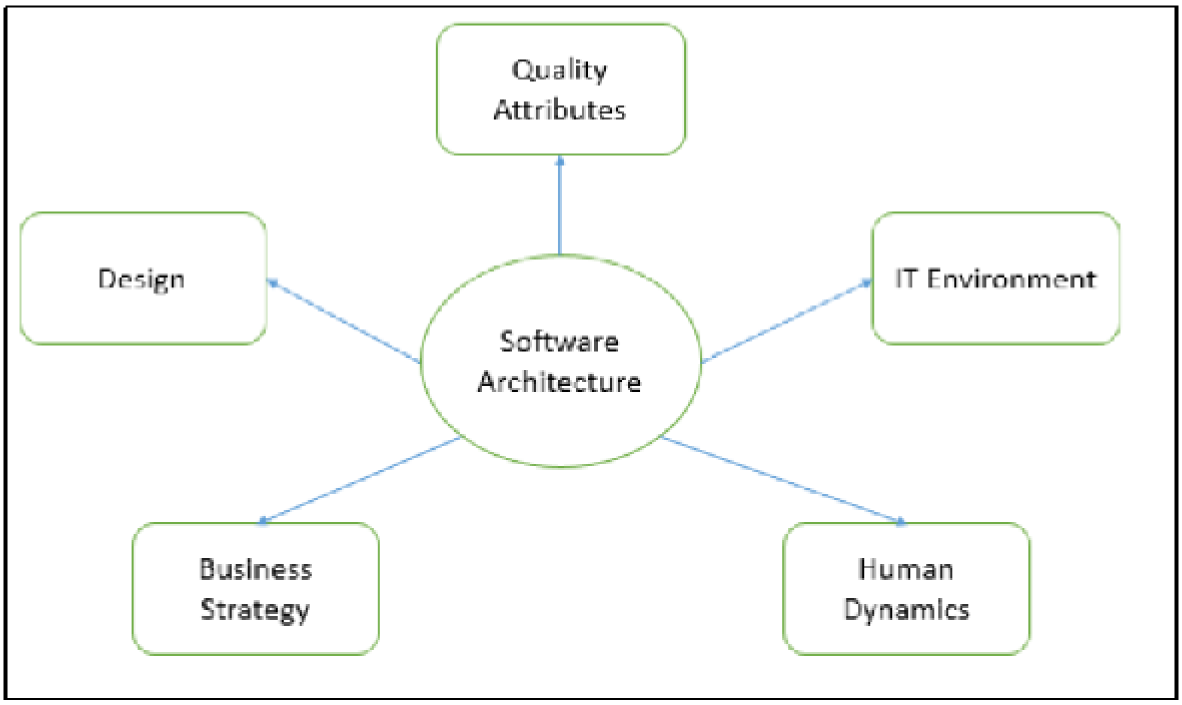
- **Prerequisites and Linking:**

Unit III	Pre-requisites	Sem. II	Sem. III	Sem. IV	Sem. V	Sem. VI
Architectural Design		OOP	DBMS			PM

Notes

Architectural Design

The architecture of a system describes its major components, their relationships (structures), and how they interact with each other. Software architecture and design includes several contributory factors such as Business strategy, quality attributes, human dynamics, design, and IT environment.



We can segregate Software Architecture and Design into two distinct phases: Software Architecture and Software Design. In **Architecture**, nonfunctional decisions are cast and separated by the functional requirements. In Design, functional requirements are accomplished.

System organisation

Reflects the basic strategy that is used to structure a system.
● Three organisational styles are widely used:

- shared data repository style;
- A shared services and servers style;
- An abstract machine or layered style.

Modular decomposition Styles

Another structural level where sub-systems are decomposed into modules .Two modular decomposition models covered
• An object model where the system is decomposed into interacting objects
• A data -flow model where the system is decomposed into functional modules which transform inputs to outputs. Also known as the pipeline model .
If possible, decisions about concurrency should be delayed until modules are implemented

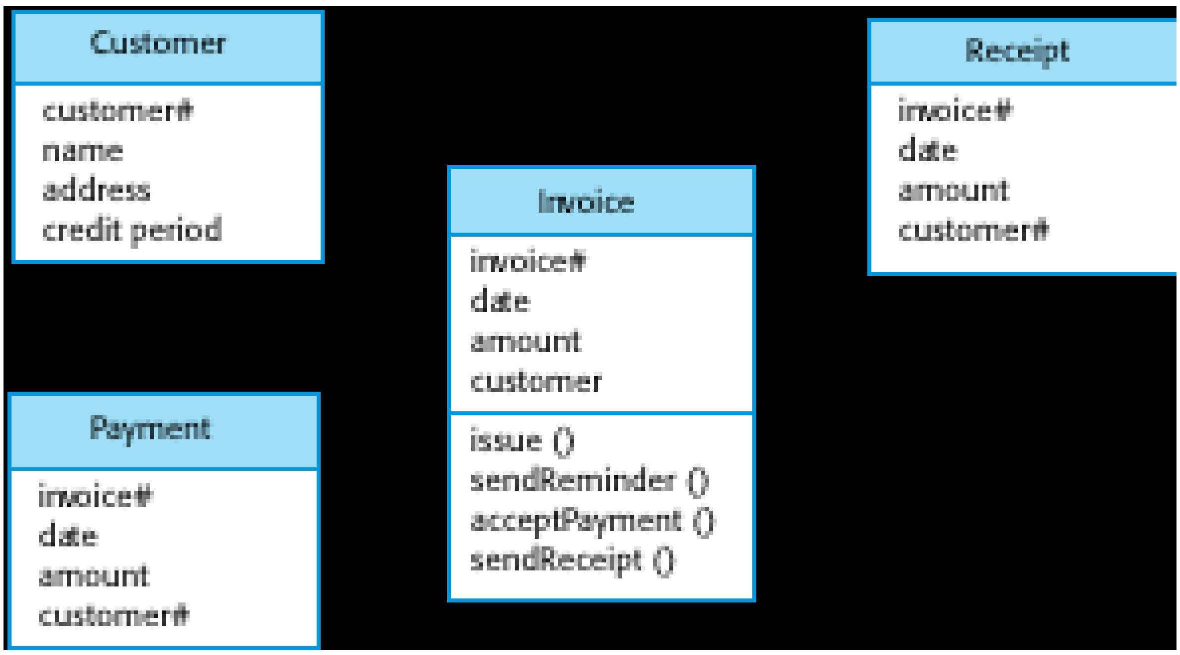
Object -oriented decomposition:

• Structure the system into a set of loosely coupled objects with well -defined interfaces

• Object-oriented decomposition is concerned with identifying object classes, their attributes and operations

• When implemented, objects are created from these classes and some control model used to coordinate object operations.

Invoice processing system



Advantages:

• Objects are loosely coupled , the implementation of the objects are modified without affecting other objects.

• Objects can be reused.

• Direct implementation of architectural components.

Disadvantages:

• To use services, objects must explicitly reference the name and the interface of other objects.

• Interface change is required to satisfy proposed system changes, the effect of that change on all users of the changed object must be evaluated.

Function Oriented Pipelining or Data flow model:

• Functional transformations process their inputs to produce outputs

• May be referred to as a pipe and filter model (as in UNIX shell)

• Variants of this approach are very common. When transformations are sequential, this is a batch sequential model which is extensively used in data processing systems

• Not really suitable for interactive systems

Control Styles

Control models : Are concerned with the control flow between sub-systems. Distinct from the system decomposition model.

• Centralized control : One sub - system has overall responsibility for control and starts and stops other sub - systems.

• Event - based control : Each sub - system can respond to externally generated events from other sub - systems or the system's environment.

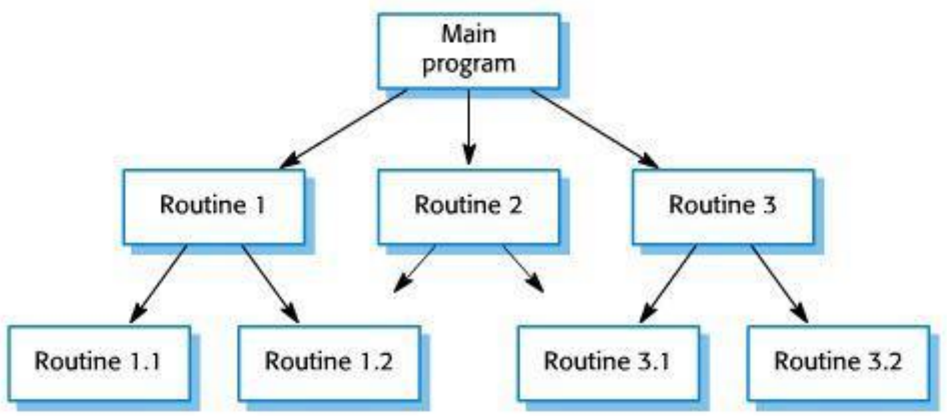
Centralized control

• A control sub - system takes responsibility for managing the execution of other sub - systems

• Call - return model: Top-down subroutine model where control starts at the top of a subroutine hierarchy and moves downwards. Applicable to sequential systems

• Manager model: Applicable to concurrent systems. One system component controls the stopping,starting and coordination of other system processes. Can be implemented in sequential systems as a case statement

Call Return Model



The call return model illustrated in the above figure .The main program can call routines 12,3.

• Routine 1 can call Routine 1.1 and Routine 1.2 and Routine 3 can call Routine 3.1 and Routine 3.2

• This model is embedded in Programming languages like Ada, C and Pascal.

• The below figure illustrates the centralized management model of control of concurrent system.

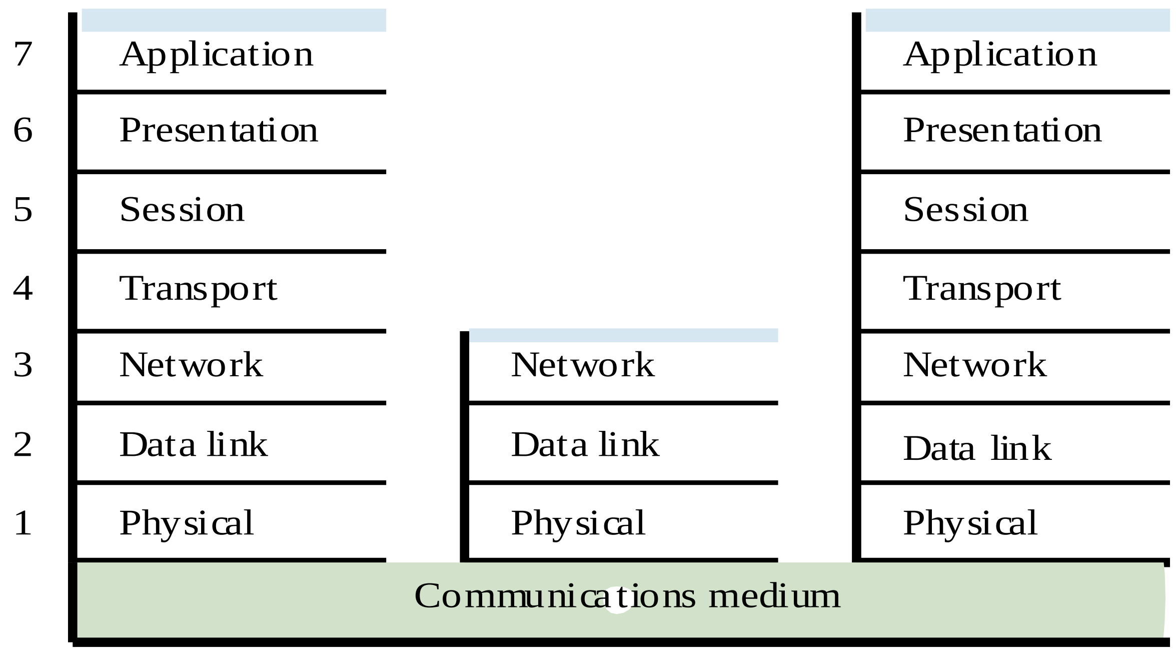
• This model is used often in 'soft' real -time systems which do not have very tight time constraints.

• The central controller manages the execution of a set of processes associated with sensors and actuators.

Reference architectures

- Reference models are derived from a study of the application domain rather than from existing systems.
- May be used as a basis for system implementation or to compare different systems. It acts as a standard against which systems can be evaluated.
- OSI model is a layered model for communication systems

OSI reference model



- **The software architect is responsible for deriving a structural system model, a control model and a sub-system decomposition model.**
- **Large systems rarely conform to a single architectural model**

User Interface Design

- Computer system design encompasses a spectrum of activities from hardware design to user interface
- Large organisations normally employ specialist interface designers for their application software
- Therefore software engineers must take responsibility for user interface design as well as for the design of the software to implement that interface
- Important factors that engineer must consider are
  - People have a limited short term memory hence if you present users with too much information they may not be able to take all of it

- When systems go wrong and issue warning messages and alarms this puts more stress on the users thus increasing the chances that they will make operational errors
- Some people see and hear better than others, some people are colour blind and some are better at physical manipulation. Hence you should not design for your own capabilities and assume that all other users will be able to cope
- Some people like to work with pictures, others with text. Direct manipulation is natural for some people but others prefer a style of interaction that is based on issuing commands to the system

https://www.youtube.com/watch?v=dQcHUyHPlx0 (User Interface Design)

User Interface Design Principles

Principle	Description
User familiarity	**The interface should use terms and concepts drawn from the experience of the people who will make most use of the system.**
Consistency	**The interface should be consistent in that, wherever possible, comparable operations should be activated in the same way.**
Minimal surprise	**Users should never be surprised by the behaviour of a system.**
Recoverability	**The interface should include mechanisms to allow users to ecover from errors.**
User guidance	**The interface should provide meaningful feedback when errors occur and provide context-sensitive user help facilities.**
User diversity	**The interface should provide appropriate interaction facilities for different types of system users.**

Design Issues

- Some design issues to be considered by user interface designers
    - How should a user interact with the computer system
    - How should information from the computer system be presented to the user
- User interaction means issuing commands and associated data to the computer system.
- On early computers this was done through a command line interface and a special purpose language was used to communicate with the machine
- Now the forms of interaction has been changed and classified into five different styles

Styles Of Interaction

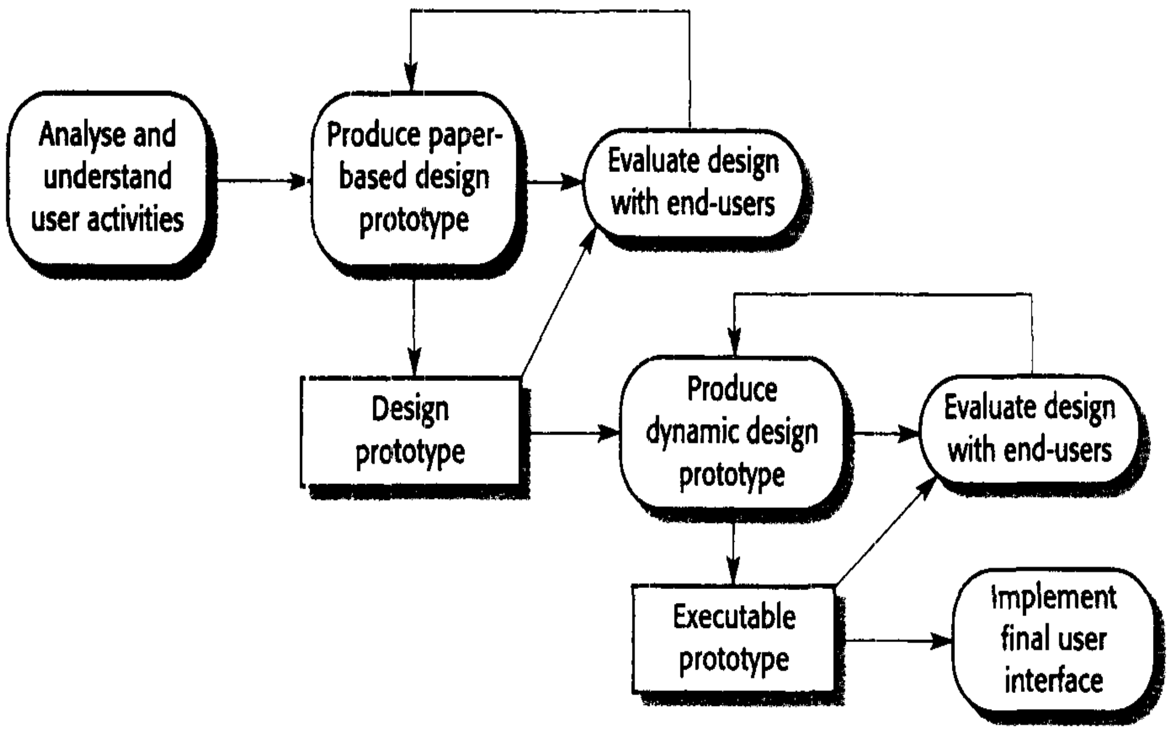
1. *Direct manipulation* The user interacts directly with objects on the screen. Direct manipulation usually involves a pointing device (a mouse, a stylus, a trackball or, on touch screens, a finger) that indicates the object to be manipulated and the action, which specifies what should be done with that object. For example, to delete a file, you may click on an icon representing that file and drag it to a trash can icon.
2. *Menu selection* The user selects a command from a list of possibilities (a menu). The user may also select another screen object by direct manipulation, and the command operates on that object. In this approach, to delete a file, you would select the file icon then select the delete command.
3. *Form fill-in* The user fills in the fields of a form. Some fields may have associated menus, and the form may have action 'buttons' that, when pressed, cause some action to be initiated. You would not normally use this approach to implement the interface to operations such as file deletion. Doing so would involve filling in the name of the file on the form then 'pressing' a delete button.
4. *Command language* The user issues a special command and associated parameters to instruct the system what to do. To delete a file, you would type a delete command with the filename as a parameter.
5. *Natural language* The user issues a command in natural language. This is usually a front end to a command language; the natural language is parsed and translated to system commands. To delete a file, you might type 'delete the file named xxx .

Advantages And Disadvantages Of Interaction Styles

Interaction style	Main advantages	Main disadvantages	Application examples
Direct manipulation	Fast and intuitive interaction Easy to learn	May be hard to implement Only suitable where there is a visual metaphor for tasks and objects	Video games CAD systems
Menu selection	Avoids user error Little typing required	Slow for experienced users Can become complex if many menu options	Most general-purpose systems
Form fill-in Easy to learn Checkable	Simple data entry	Takes up a lot of screen space Causes problems where user options do not match the form fields	Stock control Personal loan processing
Command language	Powerful and flexible	Hard to learn Poor error management	Operating systems Command and control systems
Natural language	Accessible to casual users Easily extended	Requires more typing Natural language understanding systems are unreliable	Information retrieval systems

User Interface Design Process

- It is an interactive process where users interact with designers and interface prototype to decide the features, organisation and the look and the feel of the system user interface
- Sometimes the interface is separately prototyped in parallel with other software engineering activities
- Most commonly the user interface design proceeds incrementally as the software is developed
- Before you start programming there should be some paper based designs developed
- The overall user interface design process is given in the figure


Analyse and understand user activities
Produce paper-based design prototype
Evaluate design with end-users
Design prototype
Produce dynamic design prototype
Evaluate design with end-users
Executable prototype
Implement final user interface

Core Activities In The User Design Process

1. *User analysis* In the user analysis process, you develop an understanding of the tasks that users do, their working environment, the other systems that they use, how they interact with other people in their work and so on. For products with a diverse range of users, you have to try to develop this understanding through focus groups, trials with potential users and similar exercises.

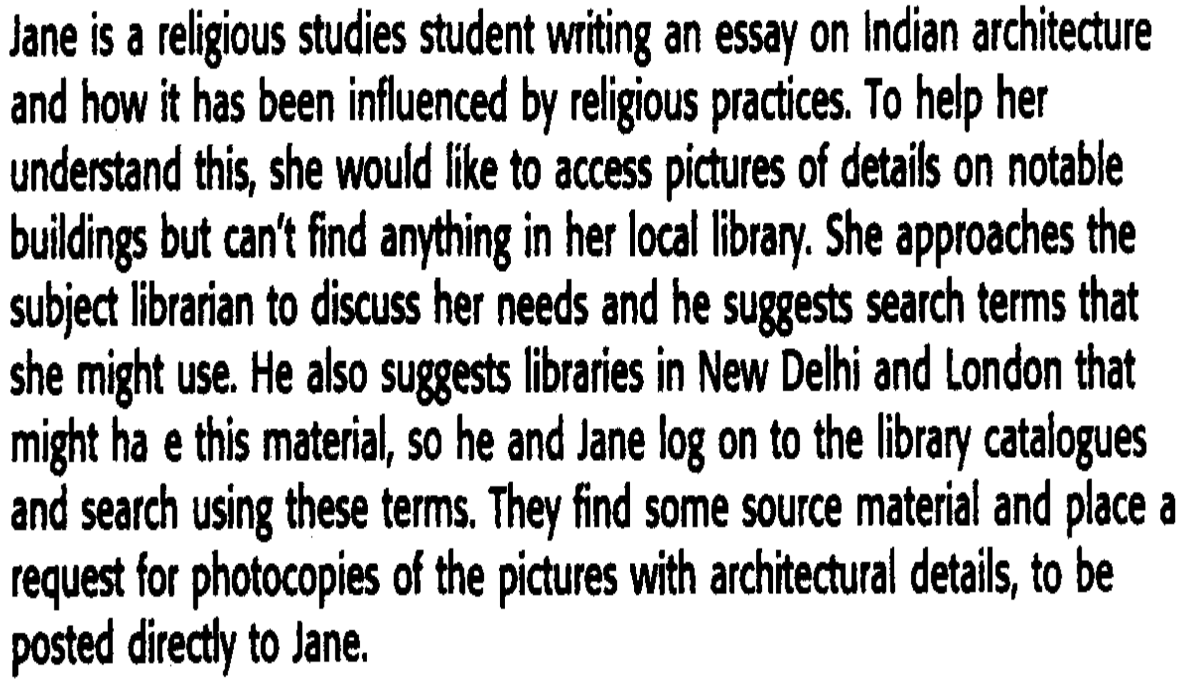
2. *System prototyping* User interface design and development is an iterative process. Although users may talk about the facilities they need from an interface, it is very difficult for them to be specific until they see something tangible. Therefore, you have to develop prototype systems and expose them to users, who can then guide the evolution of the interface.

3. *Interface evaluation* Although you will obviously have discussions with users during the prototyping process, you should also have a more formalised evaluation activity where you collect information about the users actual experience with the interface.

User Analysis

- A critical UI design activity is the analyses of the user activities that are to be supported by the computer system.
- To develop the understanding between the users requirements with the system techniques such as task analysis ethnographic studies, user interviews and observations or a commonly a mixture of all these
- A challenge for the engineers involved in the user analysis is to find a way to describe user analyses so that they communicate the essence of tasks to other designers and to the users
- UML sequence charts may be able to describe user interactions and are ideal for communicating with software engineers
- The other users may think that the charts are technical and hence will not be able to understand them but it is very important to engage the users in the design process and hence the engineer has to develop a natural language to describe user activities
- Following is the library interaction scenario

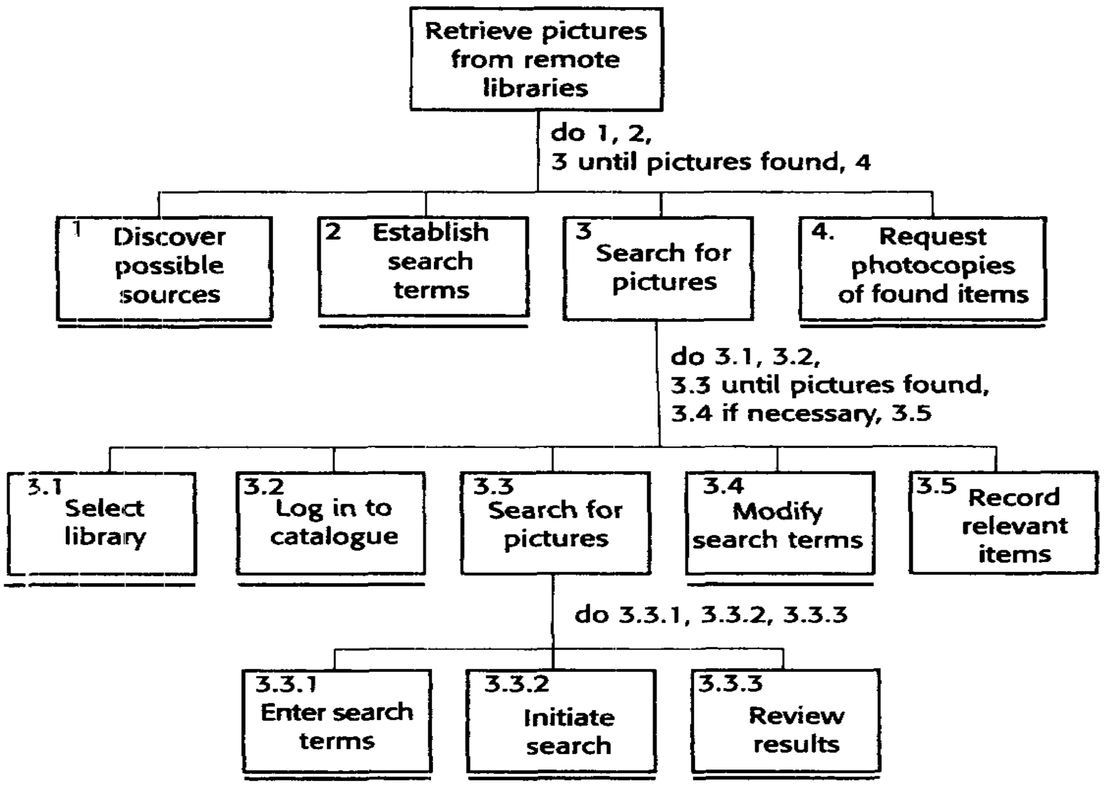
Library Interaction Scenario



Jane is a religious studies student writing an essay on Indian architecture and how it has been influenced by religious practices. To help her understand this, she would like to access pictures of details on notable buildings but can't find anything in her local library. She approaches the subject librarian to discuss her needs and he suggests search terms that she might use. He also suggests libraries in New Delhi and London that might ha e this material, so he and Jane log on to the library catalogues and search using these terms. They find some source material and place a request for photocopies of the pictures with architectural details, to be posted directly to Jane.

- The above figure is an example of natural language scenario that is developed during the specification and design process.
- It describes a situation where a student need to retrieve information from another library
- From the scenario a designer can see following requirements
    - Users might not be aware of appropriate search terms and may need to access ways of helping them choose search terms
    - Users have to be able to select collections to search
    - Users need to be able to carry out searches and request copies of relevant material
- The user analysis does not generate very specific user interface requirements but just helps to understand the needs and concerns of the system users
- There are various techniques of user analysis but the most commonly used is hierarchical task analysis
- The following figure illustrates how HTA helps in the problem for library interaction scenario

Hierarchical Task Analysis


Retrieve pictures from remote libraries
do 1, 2, 3 until pictures found, 4
1 Discover possible sources
2 Establish search terms
3 Search for pictures
4. Request photocopies of found items
do 3.1, 3.2, 3.3 until pictures found, 3.4 if necessary, 3.5
3.1 Select library
3.2 Log in to catalogue
3.3 Search for pictures
3.4 Modify search terms
3.5 Record relevant items
do 3.3.1, 3.3.2, 3.3.3
3.3.1 Enter search terms
3.3.2 Initiate search
3.3.3 Review results

User Interface Prototyping

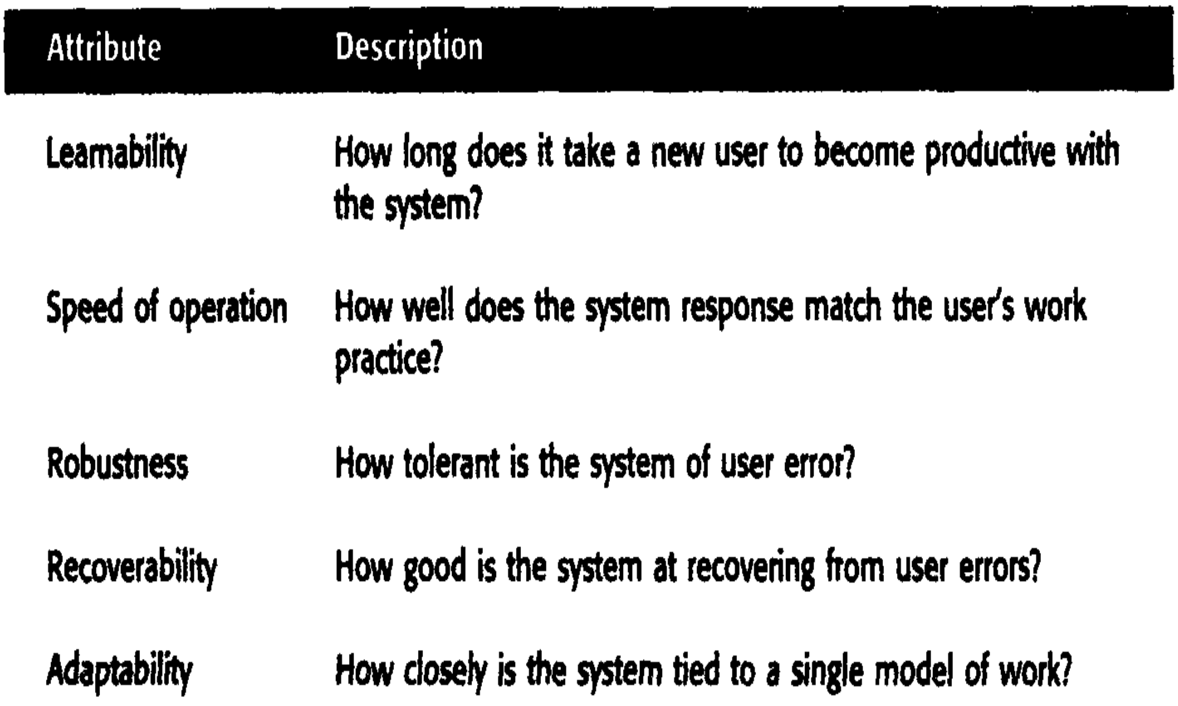
- The aim of prototyping is to allow the users to gain direct experience with the interface
- When prototyping a user interface a two staged prototyping process is used
  - Develop a paper prototype, mock ups of screen designs and walk through these with end users
  - Refine the design and develop sophisticated automated prototypes and make them available for to users for testing and activity simulation
- Paper prototype is a cheap and surprisingly effective approach to prototype development
- Storyboarding technique can be used to present the interface design which is a series of sketches that illustrate a sequence of interactions
- There are 3 approaches that can be used for user interface prototyping
  - Script Driven Approach – In this approach create screens with visual elements such as buttons and menus and associate a script with these elements. So when a user interacts with these screens the script is executed and the next screen is presented sowing them the results of their actions. There is no application logic involved

  - Visual Programming Language - These incorporate a powerful development environment, access to a range of reusable objects and a user interface development system that allows interfaces to be created quickly with components and scripts associated with interface objects

  - Internet Based Prototyping – These solutions based on web browsers offer a ready made user interface. By adding functionality of segments information can be displayed. These segments are executed automatically when the page is loaded on the browser. This is a fast way to develop user interface prototype

Interface Evaluation

- It is a process of accessing the usability of an interface and checking that it meets user requirements.
- Hence it should be a part of normal verification and validation process
- Systematic evaluation of a user interface design can be an expensive process
- There are simpler, less expensive techniques of user interface evaluation that can identify particular user interface design deficiencies
  - Questionnaires that collect the information from the users
  - Observation of users at work with the system
  - Video snapshots of typical system use
  - The inclusion in software of code which collection information about the most used facilities and the most common errors
- An evaluation should be conducted against a usability based on usability attributes shown in figure below



Attribute	Description
Leamability	How long does it take a new user to become productive with the system?
Speed of operation	How well does the system response match the user's work practice?
Robustness	How tolerant is the system of user error?
Recoverability	How good is the system at recovering from user errors?
Adaptability	How closely is the system tied to a single model of work?

Project Management

Software Project Management

The job pattern of an IT company engaged in software development can be seen split in two parts:

- Software Creation
- Software Project Management

A project is well-defined task, which is a collection of several operations done in order to achieve a goal (for example, software development and delivery). A Project can be characterized as:

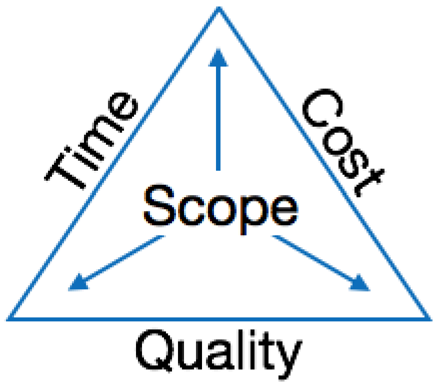
- Every project may has a unique and distinct goal.
- Project is not routine activity or day-to-day operations.
- Project comes with a start time and end time.
- Project ends when its goal is achieved hence it is a temporary phase in the lifetime of an organization.
- Project needs adequate resources in terms of time, manpower, finance, material and knowledge-bank.

Software Project

A Software Project is the complete procedure of software development from requirement gathering to testing and maintenance, carried out according to the execution methodologies, in a specified period of time to achieve intended software product.

Need of software project management

Software is said to be an intangible product. Software development is a kind of all new stream in world business and there's very little experience in building software products. Most software products are tailor made to fit client's requirements. The most important is that the underlying technology changes and advances so frequently and rapidly that experience of one product may not be applied to the other one. All such business and environmental constraints bring risk in software development hence it is essential to manage software projects efficiently.



The image above shows triple constraints for software projects. It is an essential part of software organization to deliver quality product, keeping the cost within client's budget constrain and deliver the project as per scheduled. There are several factors, both internal and external, which may impact this triple constrain triangle. Any of three factor can severely impact the other two.

Therefore, software project management is essential to incorporate user requirements along with budget and time constraints.

Software Management Activities

Software project management comprises of a number of activities, which contains planning of project, deciding scope of software product, estimation of cost in various terms, scheduling of tasks and events, and resource management. Project management activities may include:

- **Project Planning**
- **Scope Management**
- **Project Estimation**

Project Planning

Software project planning is task, which is performed before the production of software actually starts. It is there for the software production but involves no concrete activity that has any direction connection with software production; rather it is a set of multiple processes, which facilitates software production. Project planning may include the following:

Scope Management

It defines the scope of project; this includes all the activities; process need to be done in order to make a deliverable software product. Scope management is essential because it creates boundaries of the project by clearly defining what would be done in the project and what would not be done. This makes project to contain limited and quantifiable tasks, which can easily be documented and in turn avoids cost and time overrun.

During Project Scope management, it is necessary to -

- Define the scope
- Decide its verification and control
- Divide the project into various smaller parts for ease of management.
- Verify the scope
- Control the scope by incorporating changes to the scope

https://www.youtube.com/watch?v=l49ufdtS6N8 (Project Management)

Project Estimation

For an effective management accurate estimation of various measures is a must. With correct estimation managers can manage and control the project more efficiently and effectively.

Project estimation may involve the following:

- **Software size estimation**

Software size may be estimated either in terms of KLOC (Kilo Line of Code) or by calculating number of function points in the software. Lines of code depend upon coding practices and Function points vary according to the user or software requirement.

- **Effort estimation**

  The managers estimate efforts in terms of personnel requirement and man-hour required to produce the software. For effort estimation software size should be known. This can either be derived by managers' experience, organization's historical data or software size can be converted into efforts by using some standard formulae.

- **Time estimation**

  Once size and efforts are estimated, the time required to produce the software can be estimated. Efforts required is segregated into sub categories as per the requirement specifications and interdependency of various components of software. Software tasks are divided into smaller tasks, activities or events by Work Breakthrough Structure (WBS). The tasks are scheduled on day-to-day basis or in calendar months.

  The sum of time required to complete all tasks in hours or days is the total time invested to complete the project.

- **Cost estimation**

  This might be considered as the most difficult of all because it depends on more elements than any of the previous ones. For estimating project cost, it is required to consider -

  - Size of software
  - Software quality
  - Hardware
  - Additional software or tools, licenses etc.
  - Skilled personnel with task-specific skills
  - Travel involved
  - Communication
  - Training and support

Project Planning

Project planning is an organized and integrated management process, which focuses on activities required for successful completion of the project. It prevents obstacles that arise in the project such as changes in projects or organization's objectives, non-availability of resources, and so on. Project planning also helps in better utilization of resources and optimal usage of the allotted time for a project. The other objectives of project planning are listed below.

- It defines the roles and responsibilities of the project management team members.
- It ensures that the project management team works according to the business objectives.

- It checks feasibility of the schedule and user requirements.
- It determines project constraints.

Project Scheduling

Project Scheduling in a project refers to roadmap of all activities to be done with specified order and within time slot allotted to each activity. Project managers tend to define various tasks, and project milestones and them arrange them keeping various factors in mind. They look for tasks lie in critical path in the schedule, which are necessary to complete in specific manner (because of task interdependency) and strictly within the time allocated. Arrangement of tasks which lies out of critical path are less likely to impact over all schedule of the project.

For scheduling a project, it is necessary to -

- Break down the project tasks into smaller, manageable form
- Find out various tasks and correlate them
- Estimate time frame required for each task
- Divide time into work-units
- Assign adequate number of work-units for each task
- Calculate total time required for the project from start to finish

Risk Management.

Risk management involves all activities pertaining to identification, analyzing and making provision for predictable and non-predictable risks in the project. Risk may include the following:

- Experienced staff leaving the project and new staff coming in.
- Change in organizational management.
- Requirement change or misinterpreting requirement.
- Under-estimation of required time and resources.
- Technological changes, environmental changes, business competition.

There are following activities involved in risk management process:

- **Identification -** Make note of all possible risks, which may occur in the project.
- **Categorize -** Categorize known risks into high, medium and low risk intensity as per their possible impact on the project.
- **Manage -** Analyze the probability of occurrence of risks at various phases. Make plan to avoid or face risks. Attempt to minimize their side-effects.
- **Monitor -** Closely monitor the potential risks and their early symptoms. Also monitor the effects of steps taken to mitigate or avoid them.

Quality Management

The quality of software has improved significantly over the past two decades. One reason for this is that companies have used new technologies in their software development process such as object-oriented development, CASE tools, etc. In addition, a growing importance of software quality management and the adoption of quality management techniques from manufacturing can be observed. However, software quality significantly differs from the concept of quality generally used in manufacturing mainly for the next reasons [1]:

1. The software specification should reflect the characteristics of the product that the customer wants. However, the development organization may also have requirements such as maintainability that are not included in the specification.
2. Certain software quality attributes such as maintainability, usability, reliability cannot be exactly specified and measured.
3. At the early stages of software process it is very difficult to define a complete software specification. Therefore, although software may conform to its specification, users don't meet their quality expectations.

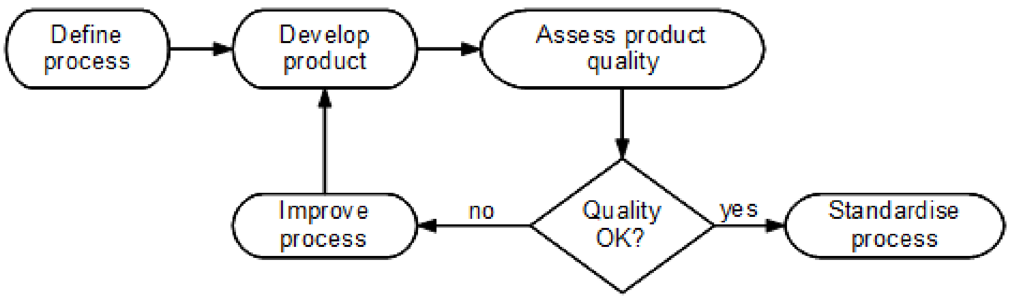
Software quality management is split into three main activities:

1. Quality assurance. The development of a framework of organizational procedures and standards that lead to high quality software.
2. Quality planning. The selection of appropriate procedures and standards from this framework and adapt for a specific software project.
3. Quality control. Definition of processes ensuring that software development follows the quality procedures and standards.

Quality management provides an independent check on the software and software development process. It ensures that project deliverables are consistent with organizational standards and goals.

Process and product quality

It is general, that the quality of the development process directly affects the quality of delivered products. The quality of the product can be measured and the process is improved until the proper quality level is achieved. Figure 12.1. illustrates the process of quality assessment based on this approach.



Process based quality assessment.

In manufacturing systems there is a clear relationship between production process and product quality. However, quality of software is highly influenced by the experience of software engineers. In addition, it is difficult to measure software quality attributes, such as maintainability, reliability, usability, etc., and to tell how process characteristics influence these attributes. However, experience has shown that process quality has a significant influence on the quality of the software.

Process quality management includes the following activities:

1. Defining process standards.
2. Monitoring the development process.
3. Reporting the software.

Quality assurance and standards

- ISO
- Documentation standards

Quality assurance is the process of defining how software quality can be achieved and how the development organization knows that the software has the required level of quality. The main activity of the quality assurance process is the selection and definition of standards that are applied to the software development process or software product. There are two main types of standards. The product standards are applied to the software product, i.e. output of the software process. The process standards define the processes that should be followed during software development. The software standards are based on best practices and they provide a framework for implementing the quality assurance process.

The development of software engineering project standards is a difficult and time consuming process. National and international bodies such as ANSI and the IEEE develop standards that can be applied to software development projects. Organizational standards, developed by quality assurance teams, should be based on these national and international standards. Table 12.1. shows examples of product and process standards.

Examples of product and process standards.

Product standards	Process standards
Requirements document structure	Project plan approval process
Method header format	Version release process
Java programming style	Change control process
Change request form	Test recording process

ISO

ISO 9000 is an international set of standards that can be used in the development of a quality management system in all industries. ISO 9000 standards can be applied to a range of organizations from manufacturing to service industries. ISO 9001 is the most general of these standards. It can be applied to organizations that design, develop and maintain products and develop their own quality processes. A supporting document (ISO 9000-3) interprets ISO 9001 for software development.

The ISO 9001 standard isn’t specific to software development but includes general principles that can be applied to software development projects. The ISO 9001 standard describes various aspects of the quality process and defines the organizational standards and procedures that a company should define and follow during product development. These standards and procedures are documented in an organizational quality manual.

The ISO 9001 standard does not define the quality processes that should be used in the development process. Organizations can develop own quality processes and they can still be ISO 9000 compliant companies. The ISO 9000 standard only requires the definition of processes to be used in a company and it is not concerned with ensuring that these processes provide best practices and high quality of products. Therefore, the ISO 9000 certification doesn’t means exactly that the quality of the software produced by an ISO 9000 certified companies will be better than that software from an uncertified company.

Documentation standards

Documentation standards in a software project are important because documents can represent the software and the software process. Standardized documents have a consistent appearance, structure and quality, and should therefore be easier to read and understand. There are three types of documentation standards:

1. Documentation process standards. These standards define the process that should be followed for document production.

2. Document standards. These standards describe the structure and presentation of documents.
3. Documents interchange standards. These standards ensure that all electronic copies of documents are compatible.

Quality planning

Quality planning is the process of developing a quality plan for a project. The quality plan defines the quality requirements of software and describes how these are to be assessed. The quality plan selects those organizational standards that are appropriate to a particular product and development process. Quality plan has the following parts:

1. Introduction of product.
2. Product plans.
3. Process descriptions.
4. Quality goals.
5. Risks and risk management.

The quality plan defines the most important quality attributes for the software and includes a definition of the quality assessment process. Table 12.2. shows generally used software quality attributes that can be considered during the quality planning process.

Software quality attributes.

Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability
Maintainability		

Quality control

Quality control provides monitoring the software development process to ensure that quality assurance procedures and standards are being followed. The deliverables from the software development process are checked against the defined project standards in the quality control process. The quality of software project deliverables can be checked by regular quality reviews and/or automated software assessment. Quality reviews are performed by a group of people. They review the software and software process in order to check that the project standards have been followed and that software and documents conform to these standards. Automated software assessment processes the software by a program that compares it to the standards applied to the development project.

Software measurement and metrics

The measurement process
Product metrics

Software measurement provides a numeric value for some quality attribute of a software product or a software process. Comparison of these numerical values to each other or to standards draws conclusions about the quality of software or software processes. Software product measurements can be used to make general predictions about a software system and identify anomalous software components.

Software metric is a measurement that relates to any quality attributes of the software system or process. It is often impossible to measure the external software quality attributes, such as maintainability, understandability, etc., directly. In such cases, the external attribute is related to some internal attribute assuming a relationship between them and the internal attribute is measured to predict the external software characteristic. Three conditions must be hold in this case:

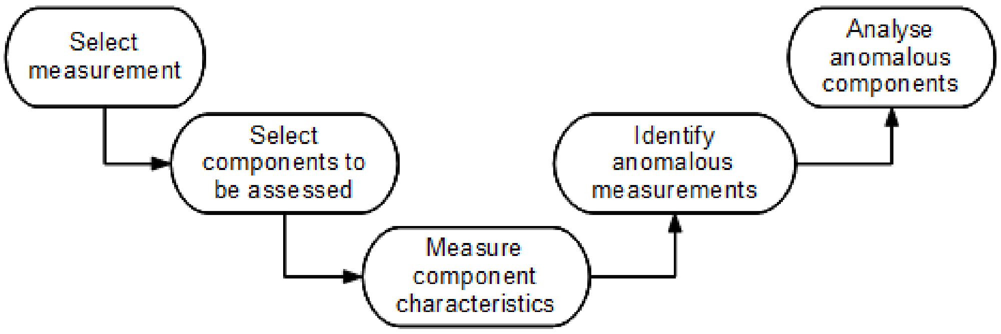
1. The internal attribute must be measured accurately.
2. A relationship must exist between what we can measure and the external behavioural attribute.
3. This relationship has to be well understood, has been validated and can be expressed in terms of a mathematical formula.

The measurement process

A software measurement process as a part of the quality control process is shown in Figure 12.2. The steps of measurement process are the followings:

1. Select measurements to be made. Selection of measurements that are relevant to answer the questions to quality assessment.
2. Select components to be assessed. Selection of software components to be measured.
3. Measure component characteristics. The selected components are measured and the associated software metric values computed.
4. Identify anomalous measurements. If any metric exhibit high or low values it means that component has problems.
5. Analyze anomalous components. If anomalous values for particular metrics have been identified these components have to be examined to decide whether the anomalous metric values mean that the quality of the component is compromised.

Generally each of the components of the system is analyzed separately. Anomalous measurements identify components that may have quality problems.


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

The software measurement process.

Product metrics

The software characteristics that can be easily measured such as size do not have a clear and consistent relationship with quality attributes such as understandability and maintainability. Product metrics has two classes:

1. Dynamic metrics. These metrics (for example execution time) are measured during the execution of a program.
2. Static metrics. Static metrics are based on measurements made of representations of the system such as the design, program or documentation.

Dynamic metrics can be related to the efficiency and the reliability of a program. Static metrics such as code size are related to software quality attributes such as complexity, understandability, maintainability, etc.

Questions:

1. What are the advantages of Architectural Design? Which factors are dependable during the design?
2. What is the decision perspective of architectural design? Explain.
3. Describe the various supportive control styles of software systems.
4. Write a short note on OSI reference architectural model.
5. What are the three system organization styles of architectural design? Explain in brief.
6. Human Factors are the basis of design principles interface design. Explain.
7. Discuss the design issues faced by the UI designers.
8. Describe the principles of user interface design process.
9. Describe the concepts of Interface evaluation.
10. List and explain the responsibilities of software manager as part of the management team.
11. Explain the structure of the project plan.
12. Elaborate the steps of Project scheduling.
13. What are the types of risk? Also explain some of the types of project risks.
14. Illustrate the stages of Risk management Process.
15. How does the process and product quality management link together?
16. List the types of Quality Assurance Standards .What is its importance?
17. What is the structure of Quality Planning? Specify in brief the attributes of software quality.
18. Explain Quality Review.
19. What is metrics and measurement? Explain.
20. Short note on Quality management.