UNIT III

DESIGN ENGINEERING

DESIGN PROCESS AND DESIGN QUALITY

Encompasses the set of principles, concepts and practices that lead to the development of high quality system or product. Design creates a representation or model of the software. Design model provides details about S/W architecture, interfaces and components that are necessary to implement the system. Quality is established during Design. Design should exhibit firmness, commodity and design. Design sits at the kernel of S/W Engineering. Design sets the stage for construction.

QUALITY GUIDELINES

- Uses recognizable architectural styles or patterns
- Modular; that is logically partitioned into elements or subsystems
- Distinct representation of data, architecture, interfaces and components
- Appropriate data structures for the classes to be implemented
- Independent functional characteristics for components
- Interfaces that reduces complexity of connection
- Repeatable method

QUALITY ATTRIBUTES

FURPS quality attributes

- *Functionality*
* Feature set and capabilities of programs
* Security of the overall system
- *Usability*
* user-friendliness
* Aesthetics
* Consistency
* Documentation
- *Reliability*
* Evaluated by measuring the frequency and severity of failure
* MTTF
- *Supportability*
* Extensibility
* Adaptability
* Serviceability

DESIGN CONCEPTS

1. Abstractions
2. Architecture
3. Patterns
4. Modularity
5. Information Hiding

6. Functional Independence
7. Refinement
8. Re-factoring
9. Design Classes

DESIGN CONCEPTS

ABSTRACTION

Many levels of abstraction.

Highest level of abstraction: Solution is slated in broad terms using the language of the problem environment

Lower levels of abstraction: More detailed description of the solution is provided

- Procedural abstraction-- Refers to a sequence of instructions that a specific and limited function
- Data abstraction-- Named collection of data that describe a data object

DESIGN CONCEPTS

ARCHITECTURE--Structure organization of program components (modules) and their interconnection

Architecture Models

(a) Structural Models-- An organized collection of program components

(b) Framework Models-- Represents the design in more abstract way

(c) Dynamic Models-- Represents the behavioral aspects indicating changes as a function of external events

(d). Process Models-- Focus on the design of the business or technical process

PATTERNS

Provides a description to enables a designer to determine the followings: (a). whether the pattern isapplicable to the current work

(b). Whether the pattern can be reused

(c). Whether the pattern can serve as a guide for developing a similar but functionally or structurally different pattern

MODULARITY

Divides software into separately named and addressable components, sometimes called modules. Modules are integrated to satisfy problem requirements. Consider two problems p1 and p2. If the complexity of p1 iscp1 and of p2 is cp2 then effort to solve p1=cp1 and effort to solve p2=cp2

If cp1>cp2 then ep1>ep2

The complexity of two problems when they are combined is often greater than the sum of the perceived complexity when each is taken separately. • Based on Divide and Conquer strategy

: it is easier to solve a complex problem when broken into sub-modules

INFORMATION HIDING

Information contained within a module is inaccessible to other modules who do not need such information. Achieved by defining a set of Independent modules that communicate with one another only that information necessary to achieve S/W function. Provides the greatest benefits when modifications are required during testing and later. Errors introduced during modification are less likely to propagate to other location within the S/W.

FUNCTIONAL INDEPENDENCE

A direct outgrowth of Modularity. abstraction and information hiding. Achieved by developing a module with single minded function and an aversion to excessive interaction with other modules. Easier to develop and have simple interface. Easier to maintain because secondary effects caused b design or code modification are limited, error propagation is reduced and reusable modules are possible. Independence is assessed by two quantitative criteria:

(1) Cohesion

(2) Coupling

Cohesion -- Performs a single task requiring little interaction with other components Coupling--Measure of interconnection among modules. Coupling should be low and cohesion should be high for good design.

REFINEMENT & REFACTORING

REFINEMENT -- Process of elaboration from high level abstraction to the lowest level abstraction. High level abstraction begins with a statement of functions. Refinement causes the designer to elaborate providing more and more details at successive level of abstractions Abstraction and refinement are complementary concepts.

Refactoring -- Organization technique that simplifies the design of a component without changing its function or behavior. Examines for redundancy, unused design elements and inefficient or unnecessary algorithms.

DESIGN CLASSES

Class represents a different layer of design architecture. Five types of Design Classes

1. User interface class -- Defines all abstractions that are necessary for human computer interaction
2. Business domain class -- Refinement of the analysis classes that identity attributes and services to implement some of business domain
3. Process class -- implements lower level business abstractions required to fully manage the business domain classes
4. Persistent class -- Represent data stores that will persist beyond the execution of the software
5. System class -- Implements management and control functions to operate and communicate within the computer environment and with the outside world.

THE DESIGN MODEL

Analysis viewed in two different dimensions as process dimension and abstract dimension. Process dimension indicates the evolution of the design model as design tasks are executed as part of software process. Abstraction dimension represents the level of details as each element of the analysis model is transformed into design equivalent

Data Design elements

-- Data design creates a model of data that is represented at a high level of abstraction

-- Refined progressively to more implementation-specific representation for processing by the computer base system

-- Translation of data model into a data base is pivotal to achieving business objective of a system

THE DESIGN MODEL

Architectural design elements. Derived from three sources

(1) Information about the application domain of the software
(2) Analysis model such as dataflow diagrams or analysis classes.
(3) Architectural pattern and styles Interface Design elements Set of detailed drawings constituting:
(1) User interface
(2) External interfaces to other systems, devices etc
(3) Internal interfaces between various components

THE DESIGN MODEL

Deployment level design elements. Indicates how software functionality and subsystem will be allocated within the physical computing environment. UML deployment diagram is developed and refined Component level design elements Fully describe the internal details of each software component. UML diagram can be used

CREATING AN ARCHITECTURAL DESIGN

What is SOFTWARE ARCHITECTURE… The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components and the relationship among them.

Software Architecture is not the operational software. It is a representation that enables a software engineer to

- Analyze the effectiveness of the design in meeting its stated requirements.
- • consider architectural alternative at a stage when making design changes is still relatively easy .
- Reduces the risk associated with the construction of the software.

Why Is Architecture Important? Three key reasons

--Representations of software architecture enable communication and understanding between stakeholders

--Highlights early design decisions to create an operational entity.

--constitutes a model of software components and their interconnection

Data Design

The data design action translates data objects defined as part of the analysis model into data structures at the component level and database architecture at application level when necessary.

DATA DESIGN AT ARCHITECTURE LEVEL

- Data structure at programming level
- Data base at application level
- Data warehouse at business level.

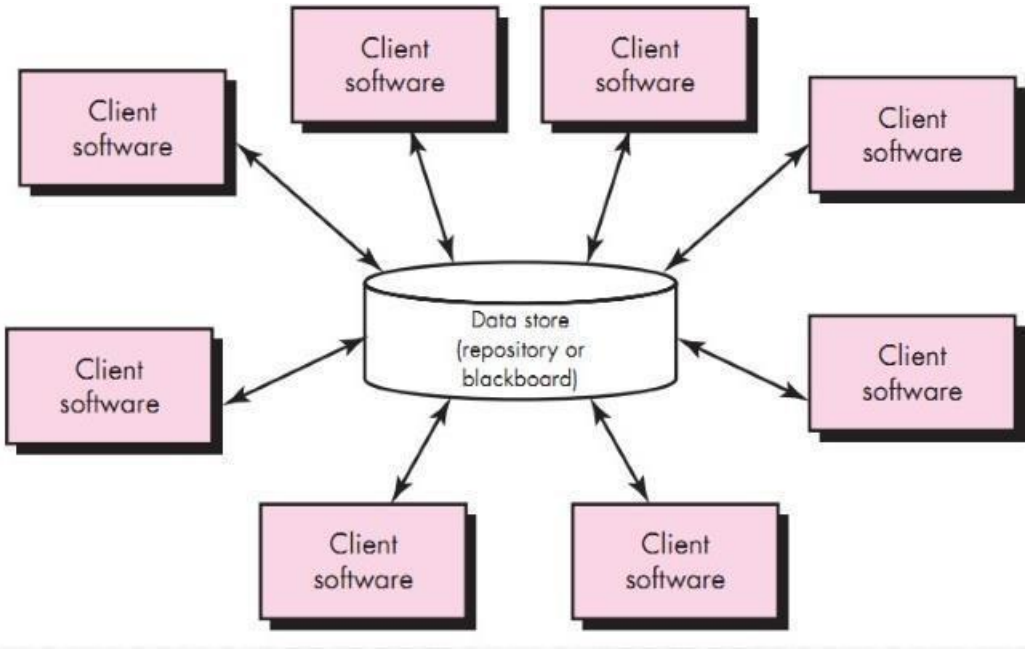
DATA DESIGN AT COMPONENT LEVEL

Principles for data specification:

1. Proper selection of data objects and data and data models

2. Identification of attribute and functions and their encapsulation of these within a class 3.Mechanism for representation of the content of **each** data object. Class diagrams may be used
4. Refinement of data design elements from requirement analysis to component level design.
5.Information hiding
6. A library of useful data structures and operations be developed.
7. Software design and PL should support the specification and realization of abstract data types..

ARCHITECTURAL STYLES
Describes a system category that encompasses:
(1) a set of *components*
(2) a set of *connectors* that enables "communication and coordination
(3) C*onstraints* that define how components can be integrated to form the system
(4) S*emantic models* to understand the overall properties of a system



Data-flow architectures
Shows the flow of input data, its computational components and output data. Structure is also called pipe and Filter. Pipe provides path for flow of data. Filters manipulate data and work independent of its neighboring filter. If data flow degenerates into a single line of transform, it is termed as batch sequential.
Call and return architectures
Achieves a structure that is easy to modify and scale .Two sub styles
(1) Main program/sub program architecture
-- Classic program structure

-- Main program invokes a number of components, which in turn invoke still other components

(2) Remote procedure call architecture
-- Components of main program/subprogram are distributed across computers over network

Object-oriented architectures
The components of a system encapsulate data and the operations. Communication and coordination between components is done via message

Layered architectures
A number of different layers are defined Inner Layer(interface with OS)
- Intermediate Layer Utility services and application function) Outer Layer (User interface)

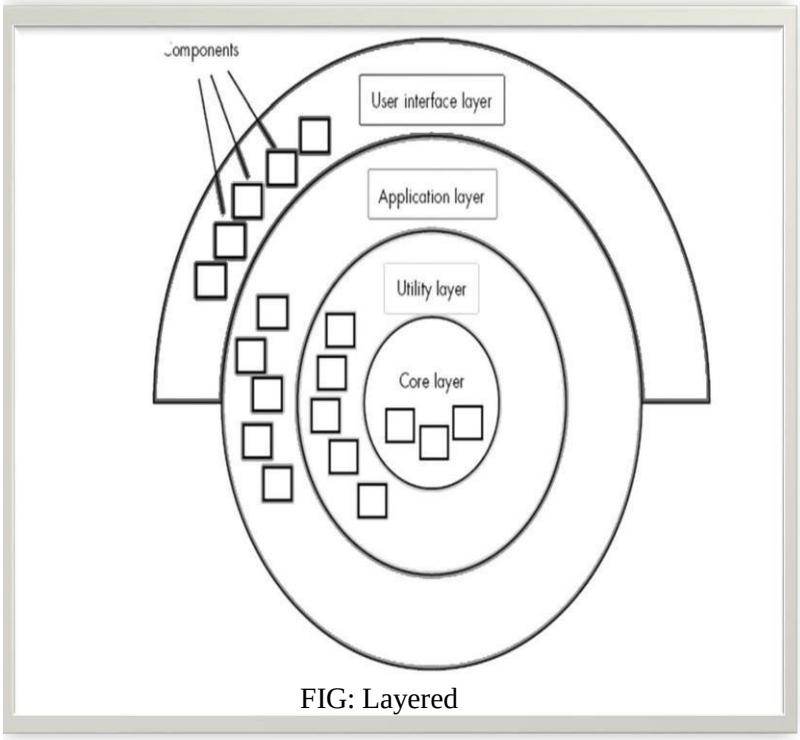


FIG: Layered

ARCHITECTURAL PATTERNS
A template that specifies approach for some behavioral characteristics of the system Patterns are imposed on the architectural styles
Pattern Domains 1.Concurrency
--Handles multiple tasks that simulate parallelism.
--Approaches (Patterns)
(a) Operating system process management pattern
(b) A task scheduler pattern 2.Persistence
--Data survives past the execution of the process
--Approaches (Patterns)
(a) Data base management system pattern
(b) Application Level persistence Pattern(word processing software)

3. Distribution
-- Addresses the communication of system in a distributed environment
--Approaches (Patterns)
(a) Broker Pattern
-- Acts as middleman between client and server.

Object-Oriented Design: Objects and object classes, An Object-Oriented design process, Design evolution.

- **Performing User interface design:** Golden rules, User interface analysis and design, interface analysis, interface design steps, Design evaluation.

Object and Object Classes

- Object: An object is an entity that has a state and a defined set of operations that operate on that state.
- An object class definition is both a type specification and a template for creating objects.
- It includes declaration of all the attributes and operations that are associated with object of that class.

Object Oriented Design Process
There are five stages of object oriented design process
1) Understand and define the context and the modes of use of the system. 2) Design the system architecture
3) Identify the principle objects in the system. 4) Develop a design models
5)Specify the object interfaces

Systems context and modes of use. It specifies the context of the system. it also specify the relationships between the software that is being designed and its external environment.

- If the system context is a static model it describes the other system in that environment.
- If the system context is a dynamic model then it describes how the system actually interact with theenvironment.

System Architecture
Once the interaction between the software system that being designed and the system environment have been defined. We can use the above information as basis for designing the System

Architecture.

Object Identification--This process is actually concerned with identifying the object classes. We can identify the object classes by the following
1) Use a grammatical analysis 2) Use a tangible entities 3) Use a behavioral approach
4) Use a scenario based approach

Design model

Design models are the bridge between the requirements and implementation. There are two type of design models

1) Static model describe the relationship between the objects. 2) Dynamic model describe the interaction between the objects

Object Interface Specification

It is concerned with specifying the details of the interfaces to objects.

Design evolution. The main advantage OOD approach is to simplify the problem of making changes to the design. Changing the internal details of an object is unlikely to effect any other system object.

Golden Rules

1. Place the user in control
2. Reduce the user's memory load
3. Make the interface consistent

Place the User in Control

- Define interaction modes in a way that does not force a user into unnecessary or undesired actions.
- Provide for flexible interaction.
- Allow user interaction to be interruptible and undoable.
- Streamline interaction as skill levels advance and allow the interaction to be customized.
- Hide technical internals from the casual user.
- Design for direct interaction with objects that appear on the screen.

Make the Interface Consistent. Allow the user to put the current task into a meaningful context. Maintain consistency across a family of applications. If past interactive models have created user expectations, do not make changes unless there is a compelling reason to do so.

USER INTERFACE ANALYSISAND DESIGN

The overall process for analyzing and designing a user interface begins with the creation of different models of system function. There are 4 different models that is to be considered when a user interface is to be analyzed and designed.

User Interface Design Models

User model —Establishes a profile of all end users of the system

Design model — A design model of the entire system incorporates data, architectural, interface and procedural representation of the software.

A design realization of the user model

User's Mental model (system perception). the user's mental image of what the interface is

Implementation model — the interface "look and feel" coupled with supporting information that describe interface syntax and semantics

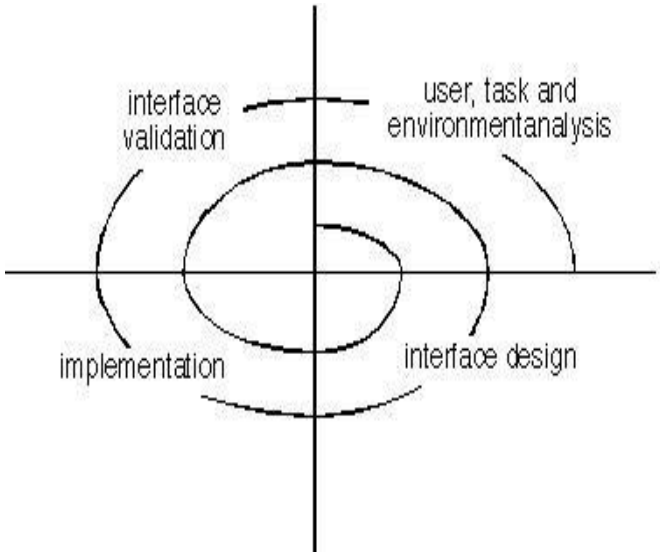
Users can be categorized as

1. Novice – No syntactic knowledge of the system and little semantic knowledge of the application or computer usage of the system
2. Knowledgeable, intermittent users- Reasonable semantic knowledge of the application but low recallof syntactic information to use the system
3. Knowledgeable, frequent users- Good semantic and syntactic knowledge

User interface analysis and design process

- The user interface analysis and design process is an iterative process and it can be represented as a spiral model

It consists of 5 framework activities 1.User, task and environment analysis 2.Interface design3.Interface construction 4.Interface validation



User Interface Design Process

Interface analysis

-Understanding the user who interacts with the system based on their skill levels.i.e, requirement gathering

-The task the user performs to accomplish the goals of the system are identified, described and elaborated. Analysis of work environment.

Interface design

In interface design, all interface objects and actions that enable a user to perform all desired task are defined

Implementation

A prototype is initially constructed and then later user interface development tools may be used to complete the construction of the interface.

- **Validation**

The correctness of the system is validated against the user requirement

Interface Analysis

Interface analysis means understanding

– (1) the people (end-users) who will interact with the system through the interface;
– (2) the tasks that end-users must perform to do their work,
– (3) the content that is presented as part of the interface
– (4) the environment in which these tasks will be conducted**.**

User Analysis

- Are users trained professionals, technician, clerical, o manufacturing workers?
- What level of formal education does the average user have?
- Are the users capable of learning from written materials or have they expressed a desire for classroom training?
- Are users expert typists or keyboard phobic?
- What is the age range of the user community?
- Will the users be represented predominately by one gender?
- How are users compensated for the work they perform?
- Do users work normal office hours or do they work until the job is done?

Task Analysis and Modeling

Analysis Techniques

- Use-cases define basic interaction
- Task elaboration refines interactive tasks
- Object elaboration identifies interface objects(classes)
- Workflow analysis defines how a work process is completed when several people (and roles) are involved

– What work will the user perform in specific circumstances?

Interface Design Steps

- Using information developed during interface analysis define interface objects and actions (operations).
- Define events (user actions) that will cause the state of the user interface to change. Model this behavior.
- Depict each interface state as it will actually look to the end-user.
- Indicate how the user interprets the state of the system from information provided through the interface.

Interface Design Patterns. Patterns are available for

– The complete UI
– Page layout
– Forms and input
– Tables
– Direct data manipulation
– Navigation
– Searching
– Page elements
– e-Commerce

Design Issues

- Response time
- Help facilities
- Error handling
- Menu and command labeling
- Application accessibility
- Internationalization

Design Evaluation Cycle: Steps:

Preliminary design Build prototype #1
Interface evaluation is studied by designer Design modifications are made
Build prototype # *n*
Interface
User evaluate's interface Interface design is complete