Week07 (c1, c2): Software Design

The design process comprise a set of principles, concepts and practices which allows a software engineer to model the system. It is phase in software engineering, in which a blueprint is developed to serve base for constructing the systems.

Good software design should have:

1. Firmness: a program should not have any bugs
2. Commodity: a program should be suitable for purpose which it is built for.
3. Delight: the experience of using program should be pleasurable one.

Objective of Software Design:

- Correctness, completeness, efficiency (resource utilization), flexibility, consistency, maintainability.

Software Design Guidelines: The Deign must:

1. Implement all explicit requirements.
2. Be understandable guide and readable
3. Provide complete picture of software.

Software Quality Guidelines: The Design should:

1. Exhibit architecture.
2. Be modular.
3. Lead to interfaces that reduce the complexity
4. Be derived using a repeatable method.
5. Be represented using appropriated notation that effectively communicate its meaning.

Software equality Attributes:

1. Functionality: is assessed with evaluating feature set and capability of the programs, also includes security of overall system
2. Usability: assessed by considering human factor, consistency and documentation
3. Reliability: is evaluated by frequency and severity of failure and the accuracy of output result.
4. Performance: is measure by considering processing speed, response time, resources consumption and throughput.
5. Supportability: combine the ability to extend the program for adaptability.

Software Design Principles: The Design should be

1. Traceable for the analysis model
2. Structured to accommodate changes
3. Exhibit uniformity and integration.
4. Reviewed to minimize conceptual (semantic) errors.

Fundamental concepts:

1. **Abstractions**: powerful design tool to consider components at an abstract level.
   Can be used in two ways:
   a. Process: mechanism of hiding irrelevant details and representing only the essential features.
   b. Entry: model or view of an item.
   Three commonly used abstraction mechanism:
   - Lower level: abstraction provide a more detail description of the solution
   - Procedural Abstraction A sequence on instruction that containing a specific and limited unction.
   - Data Abstraction: Collection of data that describe a data object.

2. **Architecture**: the whole structure of software is called a software architecture. It gives conceptual integrity for a system. Its main aim of the software design is to obtain an architectural framework of a system.
   Set of properties that should be specified an architectural design:
   - Structural propertied

- Extra-functional properties
- Families of related systems.

3. **Patterns**: represent a design structure and that structure solves a particular design problem in a specified content.
   Design pattern template: (Type of Class Diagram)

   - Pattern name: Describes the essence of the pattern in a short but expressive name
   - Intent: Describes the pattern and what it does
   - Also known as: Lists any synonyms for the pattern
   - Motivation: Provides an example of the problem
   - Applicability: notes specific design situations in which the pattern is applicable
   - Structure: Describes the classes that are required to implement the pattern
   - Participants: Describes the responsibilities of the classes that are required to implement the pattern
   - Collaborations: Describes how the participants collaborate to carry out their responsibilities
   - Consequences: Describes the "design forces" that affect the pattern and the potential trade-offs
   - Related patterns: Cross-references related design patterns

4. **Separation of Concerns:** any complex problem can be more easily handled if it is subdivide into piece that can each be solved independently. A concern I a feature or behaviors that is specified as part of requirements model by separating concern into smaller we easily manages small chunks, problem takes less effort and time to solve

5. **Modularity**: It is the single attribute of software that permits a program to be managed easily. A software s separately divide in to component sometimes they are called modules which integrate to satisfy the purpose of implementation and maintenance. Dividing the system or project into smaller parts to reduce complexity.

6. **Information Hiding**: modules must be specified and designed so that the information like algorithms and data presented is not accessible for other modules. Details of one modules are not assessable to other modules.
   Why information hiding required:
   - Reduced the likelihood of side effect
   - Emphasize communication through controlled interfaces.
   - Discourage the use of global data
   - Limit the global impact of local design decision.
   - Results in high quality software.

7. **Functional Independence**: is the concept of separation and related to the concept of modularity, abstraction and information hiding, it is accessed using two criteria:
   1. Cohesion: is shows the relationship within the modules. It is an extension of the information hiding concepts. It preform single task and requires small interaction with other components.
   2. Coupling: is shows the relationships between modules in a structure of system.

8. **Refinement**: it is top-down design approach. It is process of elaboration. I generally approach of adding details to a software design. It decomposed a statement of function in a stepwise manner till programing language statement is reached.

9. **Refactoring**: it is reorganization technique which simplified the design of component without changing its function behavior. It is done in a way that tt does not change the eternal behaviors of code but still improved its internal structure. When software is refactored, the exiting design is examined for:
   1. Redundancy
   2. Unused design elements
   3. Inefficient or unnecessary algorithm
   4. Poorly constructed or inappropriate data structures.

10. **Design Classes**: it play role in structuring the software architecture and defining how difference components interact with each other's. Provides design details that will enable analysis classes to be implemented.
    1. **Entity classes**: are refined from analysis classes during design phase. These classes represent the core data or business object manipulated by software. They encapsulate the essential attributes and behaviors related to specific components.
    2. **Boundary Classes**: are developed during design to create interface for user interaction. It manages the interaction between users and the software system such a thought GUI, APIs, etc.
    3. **Controller Classes**: are design to manage the flow of operations within the system. Moreover
        1. Manage the creation or update of any objects.
        2. Manage complex communications between set of objects.
        3. Manages validation of data communication between object or between use and the application

Design Model

1. Data Elements
    - Data Model → Data Structures and Database Architecture
2. Architectural element:
    - Application domain. Analysis classes, their relations, collaborations and behaviors.
3. Interface elements:
    - The user interface External interface to other systems, device and network. Internal interface between various design components.
4. Component elements
5. Deployment Elements:

Week07 (c3):

What is Estimation?

Project manager must set expectations about the time required to complete the software among the stakeholders, the team and the organizational management.

Elements of a Sound estimated: to generate a sound estimate, a project manager must have:

1. Work Breakdown Structure (WBS).
2. An effort estimate for each task
3. List of assumptions which were necessary for making the estimation.
4. Discussion and consensus among project team that the estimate is accurate

Assumptions make estimated more accurate:

Team members rely on assumptions to navigate incomplete information about the work ahead. When estimates need to be made based on pending decisions, assumptions provide a temporary answer for estimation purposes. It's essential to document these assumptions, ensuring transparency and accountability within the team. Moreover, by making and documenting assumptions, teams can kickstart discussions and progress on vital decisions early in the project, fostering collaboration and alignment towards successful development outcomes.

Entry Criteria for Wideband Delphi:

1. Vision and Scope documents have been agreed on by stakeholders
2. Kick-off meeting has been scheduled
3. Estimation meeting has been scheduled (1-2 hrs)

4. Moderator chosen (not the PM)
5. Agreement on the goal of the estimation session

Wideband Delphi: It is proves that a team can use to generate an estimate.it is repeatable estimation process. The project mange choose an estimation team and gain consensus among the team on the result.
Role included in Wideband Delphi:

1. Estimation Team: Project manages choose an estimation team that include reps from all project areas (developers architect, writers, designers)
2. Moderator: someone who understand Delphi process but has no stake in the results
3. Observers: selected stakeholder or user, encourage trust in the estimation process.

Wideband Delphi Process:

Step-01: Choose the Team:

1. The project mange select the estimation team and a moderator where team consist of 3-7 members

Note: The moderator should familiar with Delphi process but not have stake in outcomes. If possible the project manager should not be the moderator because he should ideally be part of the estimation team.

Step-02: Kickoff Meeting:

2. Project manager must make sure that each team member understand the Delphi process and also familiar with the project background and needs such as vision, scope and documentations.
3. Then a team brainstorm and write down assumption, generate a WBS with 10-20 tasks and agree on a unit of estimation.

Step-03: Individual Preparation:

4. Each member generate his own initial estimated for all task in the WBS and identify subtasks may help clarify an estimate.
5. For each task, the team member writes:
   - Estimate of effort required to complete task
   - Any assumption needed to make the estimate.

Step-04: Estimation Session:

6. During the estimation session, the team comes to a consensus on the effort required for reach task in WBS.
7. Each team member fills out an estimation form which contain individual estimates
8. The rest estimation session nisi divided into round during which each estimation team member revises their estimated base on group discussion.
9. Then moderator collects the estimation forms and plot the sum of effort from each form on a line.
10. Team resolves any issues or disagreements that are bought up.
11. The estimator all revise their individual estimate. The moderator updates the plot with the new total.
12. Moderator lead the team though several rounds of estimated to gain consensus and estimates.

Note: Exit Criteria: estimation session continues until the team unwilling to revise estimate.

Step-05: Assemble Tasks

13. The project manager works with the team to collect the estimates from the team members at the end of the meeting and compiles the final task list, estimates and assumptions.

Step-06: Review Results

14. The project manager reviews the final task list with the estimation team.

Week08: Architectural Design

Content:

1. Architectural Design and Decisions
2. Architectural Views
3. Architectural Patterns
4. Application Architectures

***Architectural Design:**

It is concerned with understanding how a software system should be organized and designing the overall structure of that system. Moreover it is the critical link and show relationships between design and requirement engineering

- At small level we are concerned with how individual program is decomposed in to components
- At large level we concerned with architecture of complex system includes different system program components.

Advantage of Explicit architecture:

1. Stakeholder communication.
2. System analysis: concerned about whether system can meet its non-functional requirements.
3. Large-scale reuse:

Architectural representations:

1. Informal Block Diagrams: shows entities and relationships but lack the type of relationships they have between,
2. Box and Line Diagrams: Very abstract, useful for communication with stakeholder and project planning, but they do not show the nature of component relationships no the properties of sub-systems.

Use of architectural models:

1. A way of facilitating discussion about the system: useful for communication with stakeholder and project planning as it is an abstract way not cluttered with details.
2. A way of documenting an architecture that has been designed: produce complete system model that show difference components in the system and their interfaces.

***Architectural Design Decisions:**

It is a creative process that address architectural significant requirements.
The queries which are cater in this phase are:

1. Architectural patterns or style might be used
2. Fundamental approach used to structure the systems
3. What structural component should be decomposed into sub-components
4. Is there any generic application architecture that can acts a template for the stem that is being deigned.

Architecture and system characteristics:

1. Performance: minimize communications, use large rather than fine-grain components
2. Security: use a layered architecture.
3. Maintainability: use and design replaceable components
4. Availability/: include redundant components and mechanism for fault tolerance,

***Architectural views:**

View or perspectives are useful when deigning and documenting a systems architecture. As it fives representations of the overall architecture of system meaningful and understandable to stakeholder. Moreover each architectural model only shows one view or perspective of the system.

4+1 View model of software architecture.

1. **User Case View**: user view to handling product as per requirement, User case diagram used,
2. **Design View**: organizer design information, significant features entities and attributes
3. **Process View**: Describe communication, behavior and synchronization aspects of the design
4. **Implementation View**: it address source code integrators and developers of project.
5. **Deployment View**: it describes and explain environment within system runner and executed.

Representing architectural views:
UML is not an appropriate notations for describing documenting system architecture for high-level system description.
Architectural description languages (ADLs) have been developed but not widely used.

MVC:
Description: Separate presentation and interaction from system data.
System is structured into 3 component. Model (manages system data), View (manages how data is presented to user, and Controller (manages user interaction).
When Used: when multiple ways to view and interact data, used when presenting unknown data when future requirement
Advantages: Allows data to change independently, support same data in different ways
Disadvantages: can involve additional code and code complexity.

Name	Layered Architecture	Repository Architecture	Client- Server Architecture	Pipe and Filter Pattern
Description	Organize system into layers (sub system). Where each sub system exhibit a specific service.	All data in a system is managed in a central repository and component only interact thought that repository only.	System organized into service, where each service belong to separate server. Clients are user of these services.	Processing of data in system is organized so that component carries out one type of data transmission. Data flow from one component to another as in a pipe.
When used	Building new facilities on top of the existing systems. Development spread across several team. Requirement of Multi level security	When large volume of information are generated and store for long time. Use in data-driven system.	Used when data in a shared database has to accessed from range of locations. Server can be replicated, may also use in load balancing on the system	Used in data processing application (Batch and transaction based) where input are processed in separate stages to generated output.
Advantages	Allows replacement of entire layers so long as the interface maintained. Redundant facility can be provide to each layer	Components can be independent. Changed made by one component can be propagated to all components. All data managed consistently (at one place)	Distributed across a network General functionality can be available to all client and does not need to be implemented by all services.	Easy to understand support transformation reuse. Workflow style matches the structure of many business processes. Implemented either sequential or concurrent sytem.
Disadvantages	Clean separation between layers is often difficult. Performance can be problem due to multiple levels.	Single point failure, problem in repository affect whole system. Distributing repository across component may be difficult	Single point failure due to service attacks or server failure. Performance may be unpredictable because depending on network. Management problem if servers are owned by difference organization.	Format of data should be same between components. Parsing the input and un-parsing the output cause overhead in system, thus make impossible to reuse functional transformation and incomputable data structure,

User Interface Design

Typical Design Errors:

1. Lack of consistency, Too much memorization, no context sensitivity, poor response, unfriendly, on guidance

UI modeling: it is the iterative process:

1. Compiling a complete list → Categorization component → Layout → Design → refinement.

Golden Rules:

1. **Place the user in control**
    a. Provide flexible interaction, which do not cause user into unnecessary or undesired actions
    b. Hide technical internals from the user
    c. Display descriptive messages and text for user to understand
2. **Reduce the user's memory load**
    a. Reduce demand of short term memory
    b. Define shortcuts the are easy to understand
    c. The visual layout should be based on real world metaphor
3. **Make the interface consistent.**
    a. Maintain consistency across a family of applications
    b. Keep interaction result the same as per user expectations.
    c. Meaningful user task performed.

User interface Analysis and Design Model:

User interface is the front-end application view to which user interact with the software, moreover how command if given to computer and how the data is displayed on the screen.

1. **User Model**: a profile of all end users of the system
2. **Deign Model**: a design realization of the user model
3. **Mental Model (system perception**): user mental image of what the interface is.
4. **Implementation Model**: describe interface syntax and semantics.

User Interface Analysis:

Elements of the user interface; to perform user analysis, a practitioner understand the four element

1. The user will interact with system though interface
2. The task that end users must perform to do their work
3. The content that is presented as part of interface
4. The work environment in which these task will conducted.

User Interface Analysis – User Analysis:

Analyst get the user's mental model and the design model to converge by understanding

- The users themselves and how these people use the system

Set of Question should be answered:

- What level of formal education the average user have.
- What is the age range of the user community
- Are the user trained professional, technician, or workers

User Interface Analysis– Task Analysis and Modeling

- Task analysis strived to know and understand:
- The work the user performs in specific circumstances
- The sequence of work task and hierarchy of tasks

Use cases:

- Show how user performs some specific work- related task

User interface Analysis- Content Analysis:

The display content may range from character-based reports, to graphical displays, to multimedia information

Set of Question should be answered:

- Are various types of data assigned to consistent locations of the screen
- Are user able to customize the screen location for content.(responsiveness)
- How color is used to enhance understanding.

Categorize Component: can be controls and container which are grouped accordingly multiple task (forms), similar task (toolbars), and ensure clarity of use

User interface Analysis- Work Environment Analysis:

Software product need to be designed to fit into the work environment, otherwise they may be difficult or frustrating to use.

Factors to consider:

- Display size and height, type of lightening, mouse type, weather, temperature, time restrictions.

User Interface Design:

User Interface Design – User system interaction:

Two problems must be addressed in interactive systems design

- How should information from the user be provided to the computer system?
- How should information from the computer system be presented to the user?

Design Issues:

1. Reponses time: time between request and response of the system take long which frustrates the user
2. Error Handling: poor error message
3. Help facilities: user request helps when he need some infomation and he cannot find it then the user is in trouble
4. Application accessibility: states whether application is simple to interactor with or not. Special guidelines are given to user while interacting with software.

User interface Design- Guideline for Error Message:

The message should:

- Describe the problem in plain language that atypical user can understand.
- Indicate any negative consequences of the error so that user can be ensured that they had not occurred.
- Never place blame on the user (be non-judgmental)

User Interface Design- Menu labeling and Typed Commands:

Question for Menu labeling and Typed Commands:

- Will every menu option have a corresponding commands?
- What can be done if command is forgotten?
- What form will a command take: either control sequence or function ley or a typed word?

User Interface Design- UI Design Principle:

1. **User Familiarity**: the interface should be based on user oriented terms and concepts.
2. **Consistency**: the system command and menu should have same format and parameter
3. **Recoverability**: user make often mistake while working with the system, these mistake can be reduce but cannot eliminated.
4. **Efficiency**: enable multiple user interface where difference task can perform simultaneously.
5. **Readability**: All information presented through interface should be readable by young and old all kind of user.
6. **Track State**: track the user state so that user can easily continue after a break from where they left from

User Interface Design- UI Design Guideline:

1. User prefer not to scroll.
2. Navigation would be consistence (headers,, menus) available on the pages

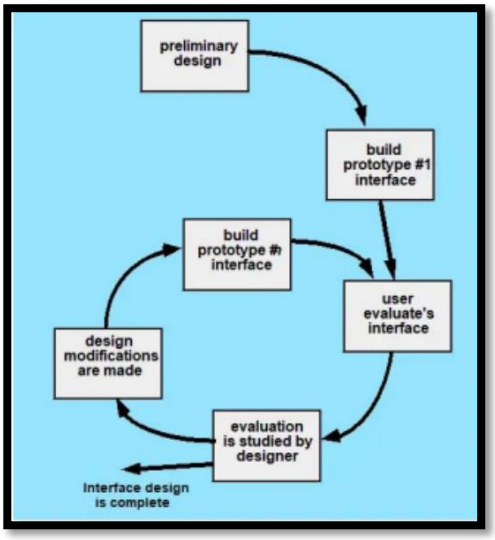
3. Functionality over aesthetics: A simple button might be better navigation than aesthetically pleasing.

Design and Prototype Evaluation:

1. **Design Evaluation**: This happens before a prototype is built. It involves reviewing the design model itself based on various criteria. These criteria include how much users will have to learn to use the interface, how long it will take them to complete tasks, and how much they'll have to remember to use it effectively.

2. **Prototype Evaluation**: This is where a prototype is created and then tested by users. This can range from an informal setting to a more formal study with questionnaires. The goal is to see if the prototype meets the needs of the users and follows usability best practices.

- **NOTE:** The text mentions that this is an iterative process. This means that after the prototype is evaluated, it can be redesigned to address any issues that are found. Then it's retested and this cycle continues until the problems are resolved.



Design and Prototype Evaluation- Evaluation Techniques:

- Questionnaires for user feedback
- Video recording of system use
- Review code to collect information about the user errors.

Efficient GUI:

- **Human Perception**: GUI should be similarity, balance, vision
- **Visual pleasing composition**: GUI should be balance, symmetric, unity, simplicity, grouping, sequentially

UI Design has its own principles:
Alignment, Balance, Clarity, Compatibility, Comprehensibility, Configurability, Consistency, Control, Directness, Efficiency, Familiarity, Flexibility, Forgiveness, Predictability, Recovery, Responsiveness, Simplicity, Uniformity

Material Design: Material is a Metaphor

- **Goal-1**: Create a visual language that synthesizes classic principles of good design with the innovation and possibility of technology and science

- **Goal-2**: Develop a single underlying system that allows for a unified experience across platforms and device sizes. Mobile precepts are fundamental, but touch, voice, mouse, and keyboard are all first-class input methods.