Unit-2
Design

Chamundeswari Arumugam
Professor
SSN College of Engineering, Chennai

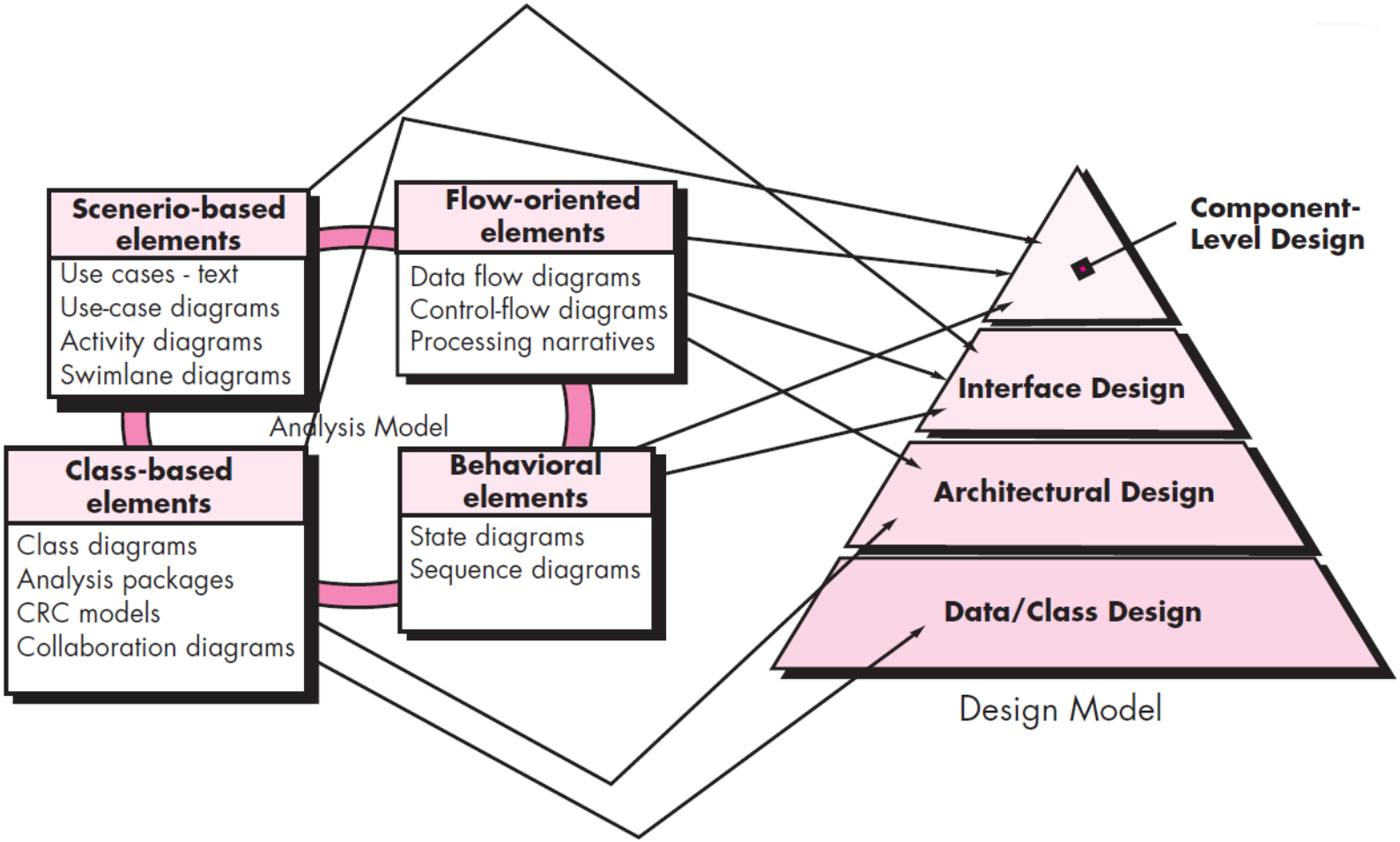
February 2017

Outline

- Translating the requirements model into the design model
- Design Process
- Design Concepts
- Design Model

Translating the requirements model into the design model

Fig : Requirement model to Design model



Design Process

Design

- Design sets a stage for construction (code generation and testing).
- Goal of the design is to produce a model that exhibits firmness, commodity, and delight.
- It produces a data design, an architectural design, an interface design, and a component design.
- It povides representation of software that can be assessed for quality.
- It accurately translates requirements into a finished software products or system. It support to build stable system.
- It serves as foundation for all software engineering and software support activities that follow.
- Iterative process - initial iteration represents hig level of abtraction. As it iterates it provides a low level abstraction.

Design Quality Guidelines

- Use recognizable architecture styles
- Logically decompose the complex task into modules
- Reduce the complexity of connection between components and external environment.
- Exhibit independent functional characteristics
- Use notation that effectively communicates its meaning.

Quality attributes

- FURPS quality attributes
- Functionality, Usability, Reliability, Performance, Supportability

Evolution of software design

- Structured programming, Data flow, Object oriented approach
- Architecture and design pattern, Aspect oriented method
- Model driven development, Test driven development

Good design

- Accomodate implicit and explicit requirements
- Readable, understandable guide for developers, testers adn supporters.
- Address data, functional, and behavioral from implementation perspective.

Design Concepts

Important software design concept

- Abstraction, Architecture, Patterns, Separation of concerns, Modularality, Information Hiding, Functional Independence, Refinement, Aspects, Refactoring, OOD concepts, Design classes.

Abstraction

- Many level of abstraction - high, low. Low provides a more detailed desscription of the solution.
- Create procedural and data abstraction. Procedural abstraction describes specific and limited function and data abstraction describes a data object.

Architecture

- Structure or organization of program components, their interaction, data structures used.
- It should specify the properties - structural, extra functional and families of related systems.
- Different models used to represent architecture - structural models, framework models, dynamic models, process models, functional models.
- Architectural description language describes system components and their connectivity.

Patterns

- A software design pattern is a general reusable solution to a commonly occurring problem within a given context in software design.
- Designer can decide (1) whether the pattern is applicable to the current work, (2) whether the pattern can be reused (hence, saving design time), and (3) whether the pattern can serve as a guide for developing a similar, but functionally or structurally different pattern.

Separation of Concern

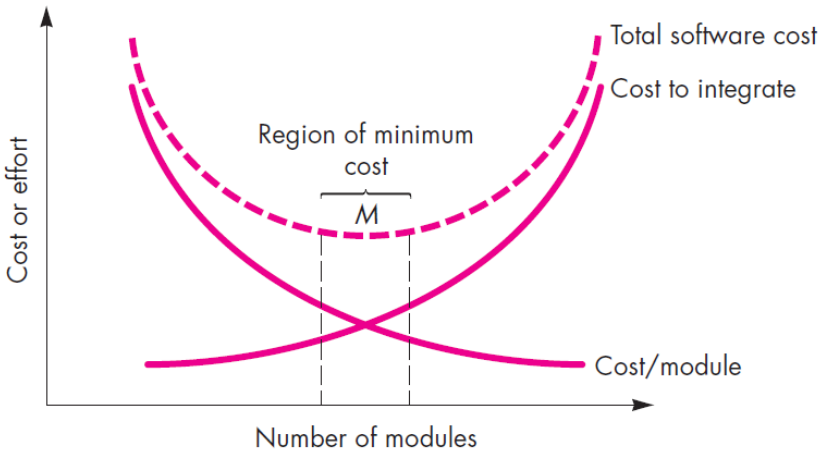
- complex problem can be more easily handled if it is subdivided into pieces
- A concern is a feature or behavior that is specified as part of the requirements model for the software
- separating concerns into smaller, and therefore more manageable pieces, a problem takes less effort and time to solve.
- Separation of concerns is manifested in other related design concepts: modularity, aspects, functional independence, and refinement.

Design Concepts(contd..)

Modularity

- Software is divided into components, called modules. Sub-division reduce the effort.
- Understanding the software becomes easier. it enables reduction in cost to build the software.
- Fig. - Effort to develop an individual software module decreases as the total number of modules increases. The effort(cost) to integrate the modules also grows.
- Advantages :- (1) easy the development plan, (2)software increments can be defined and delivered, (3) can accommodate changes, this in turn ease testing, and maintenance process with no serious side effects.

Fig. Modularity and software cost



Information Hiding

- Modules should be designed so that information(alg and data) contained within a module is inaccessible to other modules.
- Hiding defines and enforces access constraints to both procedural details within a module and any data structure used by the module.
- Modification becomes easy and this in turn aids testing, and maintenance process.

Refinement

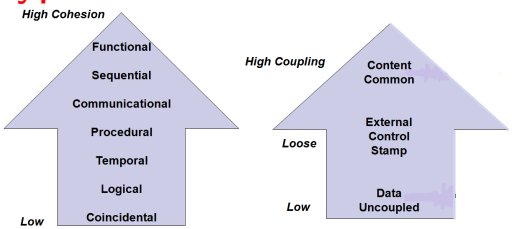
- Decopomposing a macroscopic statement, which is defined at high level abstraction, of a function, in a stepwise fashion until programming statements are reached.
- Abstraction and refinement are complementary concepts.
- Refinement helps to reveal low level design as design progresses.

Design Concepts (contd..)

Functional Independence

- This concept is an outgrowth of separation of concerns, modularity, and the concepts of abstraction, and information hiding.
- It is achieved by developing modules with "single minded" function and an "aversion" to excessive interaction with other modules.
- Advantages - Easy to maintain, modification limits side effects, error propagation is reduced, encourages reusable modules.
- Two qualification criteria : cohesion and coupling.
- Cohesion is a measure that defines the degree of intra-dependability within elements of a module. The greater the cohesion, the better is the program design.
- Coupling is a measure that defines the level of inter-dependability among modules of a program. The less the coupling, the better is the program design.

Fig. Types of Cohesion and Coupling



Aspects

- A concern is a feature or behavior that is specified as part of the requirements model for the software.
- An aspect is a representation of a crosscutting concern.
- Cross-cutting concerns are parts of a program that rely on or must affect many other parts of the system.
- The concerns include : requirements, use cases, features, data structures, quality-of-service issues, variants, intellectual property boundaries, collaborations, patterns and contracts.
- Concerns span the entire system and cannot be easily compartmentalized.
- Consider two requirements, A and B. Requirement A crosscuts requirement B. ie. B cannot be satisfied without taking A into account.

Refactoring

- Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code [design] yet improves its internal structure.
- During refactoring, the existing design is examined for redundancy, unused design elements, inefficient or unnecessary algorithms, poorly constructed or inappropriate data structures, or any other design failure that can be corrected to yield a better design.

Design concepts (contd..)

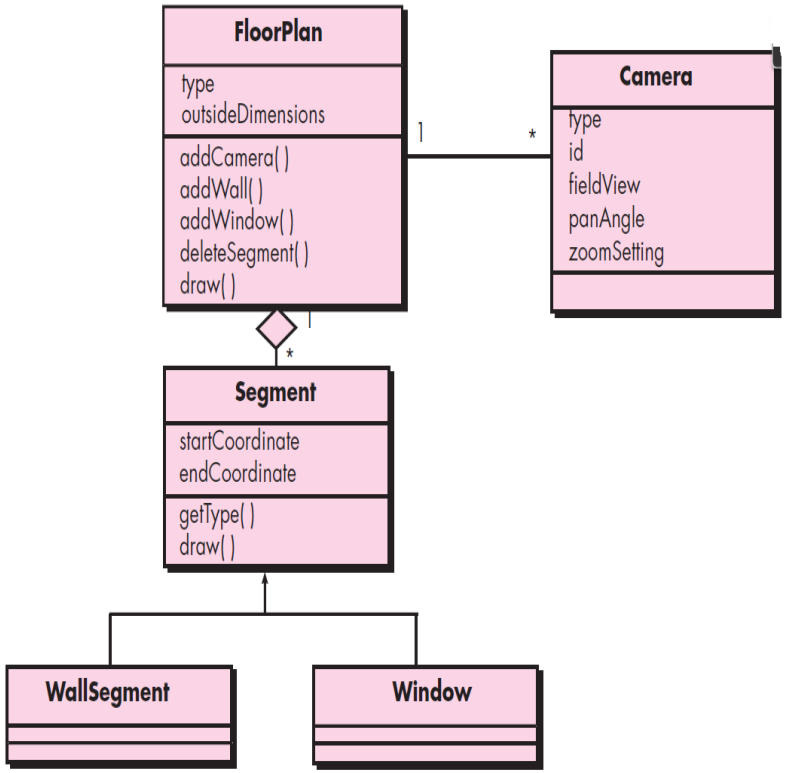
OO Design concepts

- OO paradigm is widely used in modern SE.
- Concepts inlude : classes, objects, inheritance, messages, polymorphism, etc.

Design classes

- Redefine the analysis class by providing design detail to implement.
- Types of design classes : UI classes, business domain classes, process classes, persistent classes, system classes.
- Level of abstraction is reduced as each analysis class is transformed into a design representation.
- Design class represent more technical details as a guide for implementation.
- four characteristics of a well formed design class: - (1) complete & sufficient (2) primitiveness (3) high cohesion (4) low coupling

Fig :Design class

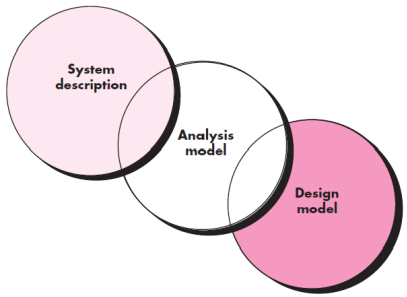


Design Model

Design model

- A design model that encompasses architectural, interface, component level, and deployment representations is the primary work product that is produced during software design
- All elements of the requirements model will be directly traceable to parts of the design model
- Four design models required for a complete specification of design.

Fig. Bridge between the system description & design model



The design model can be viewed in two different dimensions.

- Process dimension indicates the evolution of the design model as design tasks are executed as part of the software process.
- Abstraction dimension represents the level of detail as each element of the analysis model is transformed into a design equivalent and then refined iteratively

Design model elements

- Elements : - Architecture, interface, component, deployment
- These elements are elaborated as part of design. More implementation specific details are provided.
- The elements of the design model should be traceable to the requirements model.

Data design element

- Data design creates a model for data that has more influence on architecture of the software.
- It is important part of the software design at 3 level :- program component level, application level, business level.
- Program component level concentrates on data structures and associated algorithms, while application level on database, and business level on datastore.

Architecture design element

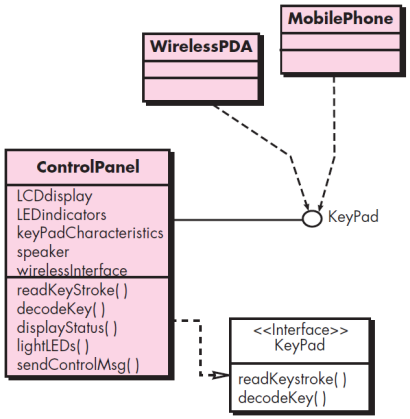
- Architecture design element gives an overall view of the software. It is derived from 3 sources - application domain, DFD/analysis classes, architecture style & pattern.

Design Model

Interface design elements

- Interface design elements concentrates on information flows and communication among the components.
- Elements of interface design - user interfaces, external interfaces, internal interfaces.
- Usability / UI - Usability design incorporates aesthetic elements (e.g., layout, color, graphics, interaction mechanisms), ergonomic elements (e.g., information layout and placement, metaphors, UI navigation), and technical elements (e.g., UI patterns, reusable components).
- External interface - Definitive information about the entity to which information is sent or received.
- Internal interface - all operations and the messaging schemes required to enable communication and collaboration between operations in various component level design classes.

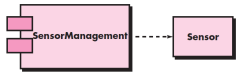
Fig. User interface representation for Control- Panel



Component level design element

- Describes the internal details of the software.
- Component level design element defines data structures and algorithm details of a component.
- Defines an interface that allows access to all component operations(behaviour)

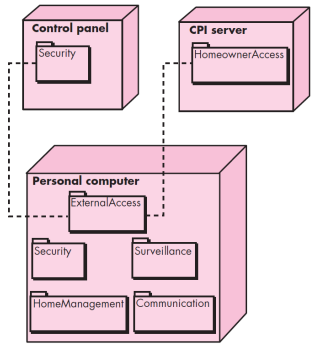
Fig. A UML Component diagram



Deployment level design elements

- Deployment-level design elements indicate how software functionality and subsystems will be allocated within the physical computing environment that will support the software.
- It shows the computing environment but does not explicitly indicate configuration details

Fig. A UML Deployment diagram



Summary

- Analysis models
- Design models
- Relationship between analysis and design model
- What is design?
- Quality attributes of Design
- What is good design?
- 12 design concepts
- Four design model elements

Assessment

- Example of design concept - abstraction
- Draw a sample architecture diagram
- Give an example for information hiding
- Identify the notation and draw a sample class diagram
- Identify the notation and draw a sample component diagram

References I

[1] RogerS. Pressman.
"Software Engineering a Practitiner's Approach"".
Seventh Edition, McGraw Hill Higher Education, 2010.