GOVERNMENT OF ANDHRA PRADESH

COMMISSIONERATE OF COLLEGIATE EDUCATION

DESIGN CONCEPTS

Software Engineering

Computer Science

Smt. G.Sumalatha MTech,(PhD)

Govt. Degree College , Salur

Email. Id : sumalathagopathoti@gmail.com

Objectives

- Describe various design concepts.
- Discuss the use of design concepts.
- Describe the different levels of software design.
- Discuss the need of decomposition.
- Describe the concepts related to modern software engineering.

Design Concepts

- Fundamental software design concepts are listed as follows

1. Abstraction
2. Architecture
3. Pattern
4. Separation of concerns
5. Modularity
6. Information hiding
7. Functional Independence
8. Refinement
9. Aspects
10. Refactoring
11. Object-oriented design concepts
12. Design classes

Abstraction

- Solution to any software problem represented with many levels of abstraction.
- At Highest level of abstraction state the solution in broad terms.
- At lower level of abstraction more detailed description is added to solution.
- At lowest level of abstraction solution is stated in a manner, that can be directly implemented.
- We can create procedural abstraction and data abstraction with different levels
  - Procedural abstraction: Sequence of instructions with limited function.
  - Data abstraction: it is a collection of data that describes data object.

Architecture

- It is the “the overall structure of the software”.
- In simple terms it is the organization of components, interaction among the components and the structure of data used by those components.
- Architecture gives us framework from which more design details are added.

Architectural design properties :

- Structural properties defines the components of the systems, and how these components are connected and interact with one another.
- Extra-functional properties addresses how the architecture meets non-functional requirements.
- Families of related systems addresses the ability to reuse architectural building blocks

Patterns

- Design pattern is a general repeatable solution to a common problems in software design.
- It is a description or template for how to solve a problem that can be used in many different situations.

Separation of Concerns

- It is suggested that any complex problem can be easily handled by dividing it into pieces and solve them independently.
- A concern is a feature or behaviour that is specified in the requirements model.

Modularity

- Software is divided into separately named and addressable components, sometimes called modules.
- Modules are to integrated later to satisfy system requirements.
- Modularity is the single attribute of a software that permits a program to be managed easily.
- Modularity makes understanding of design modules easier, as a result it reduces the cost of the software to be built.

Information Hiding

- The basic principle of information hiding is that modules must hide from one another.
- Effective modularity can be achieved by defining modules as much as independent as possible.
- Access constraints are enforced on both procedural details and local data structure of a module.
- Information hiding provides the greatest benefits when modifications are required in software as part of maintenance.

Functional Independence

- Modules are to be developed with “single minded” function and little interaction with other modules.
- Modules should address specific requirement and has simple interface with other modules.
- Independent modules are easy to maintain as they have
  - limited modifications.
  - limited error propagations.
  - reusable.

Functional Independence Criteria

Cohesion

- It is an indication of relative strength of a module and is natural extension to information hiding.
- A cohesive module performs a single task and has less interaction with the other modules.
- A good design should always strive to achieve high cohesion.

Coupling

- Coupling is an indication of interconnection between modules.
- A good design should always strive to achieve low coupling.

Refinement

- Refinement is a process of elaboration.
- Abstraction and Refinement are complementary concepts.
- Abstraction suppresses internal details and refinement reveals internal details of modules.

Aspects

- During the requirement model each requirement is considered independently, but in practice requirements can not isolated easily.
- An aspect is a representation of a cross-cutting concern.
- It means that when A and B are two requirements, B can not be satisfied with out considering A.
- During the design process requirements A and B are refined into A* and B*.
- Here the design concern is that B* cross-cuts A*.

Refactoring

- It is a reorganization technique.
- It simplifies the design of the components without changing its functional behaviour.
- It is the process of changing the software system in such a manner that
  - Should not alter its external behaviour.
  - Improve its internal structure.

Object-oriented Design Concepts

- The object-oriented (OO) paradigm is widely used in modern software engineering.
- The OO concepts are
  - Classes and objects
  - Inheritance
  - Abstraction
  - Encapsulation
  - Polymorphism and other.

Design Classes

- The design model evolves, we should define set of design classes.
  - User interface classes
  - Business domain classes
  - Process classes
  - Persistent classes
  - System classes
- Design classes provide more technical details.
- They act as a guide for implementation.

Summary

- Abstraction hides internal details where as refinement reveals the information.
- Architecture is overall structure of the software.
- Patterns provides repeatable solution to common problems.
- Modularity divides the software into modules, functional independence makes the modules “single minded”.
- Aspects deals with the dependent requirements, refactoring improves the structure of a module.
- Object oriented concepts and design classes.

References

Text Books:

1. Roger Pressman S., “Software Engineering: A Practitioner's Approach”, 7th Edition, McGraw Hill, 2010.
2. Sommerville, “Software Engineering”, Eighth Edition, Pearson Education, 2007

Web Links:

1. https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-355j-software-engineering-concepts-fall-2005/lecture-notes/cnotes4.pdf
2. https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-355j-software-engineering-concepts-fall-2005/lecture-notes/cnotes5.pdf
3. https://drive.google.com/file/d/1-e8kYCqYRhk1Dg_JKdSXbcWNNZXxf632/view
4. https://cdn.shopify.com/s/files/1/0457/4009/7694/files/software_engineering_pdf_pressman_7th_edition.pdf

Thank You

Smt. G. Sumalatha MTech, (PhD),
email ID:sumalathagopathoti@gmail.com