Software Design Fundamentals

Introduction

Design Principles

SEN-261 : Software Engineering

Tazeen Muzammil

Software Design

- General definition of design
  - "… the process of applying various techniques and principles for the purpose of defining a device, a process, or a system in sufficient detail to permit its physical realization."
- Goal:
  - To produce a model or representation that will later be built
- Major Areas of concern:
  - Data Design
  - Architecture Design
  - Interfaces Design
  - Components Design

Design and Software Quality

- Quality
  - Design is the place where quality is fostered in software engineering.
  - Design provides us with representation of software that can be assessed for quality.
  - Design is the only way we can translate customer requirements into finished software product or system.
  - Software design serves as the foundation for all the software engineering and software support steps.

Characteristics of a Good Design

- The design must implement all of the explicit requirements contained in the analysis model, and it must accommodate all of the implicit requirements desired by the customer.
- The design must be readable, understandable guide for those who generate code and for those who test and support the software.
- The design should provide a complete picture of the software, addressing the data, functional and behavioral domains from an implementation perspective.

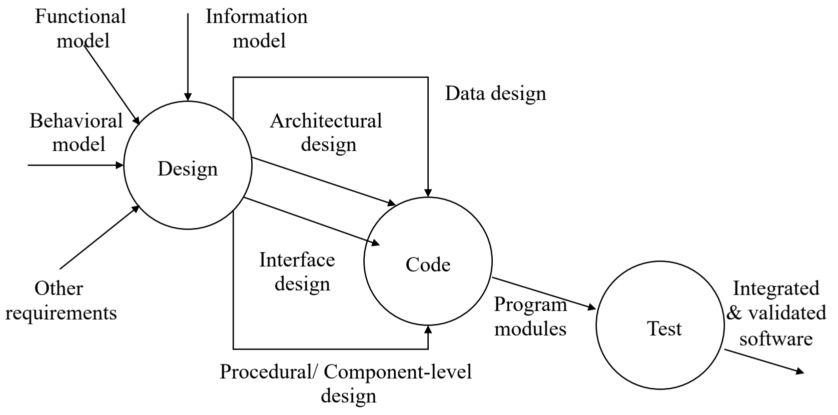
General Design Guidelines

- A design should exhibit architectural structure that
  1) has been created using recognizable design patterns,
  2) is composed of components that exhibit good design characteristics
  3) and can be implemented in an evolutionary fashion.
- Logically partitioned into components that perform specific tasks and subtasks.
- Lead to interfaces that reduce complexity of connections between modules and with the external environment.
- Derived using a repeatable method driven by information gathered during requirements.
- Lead to data structures that are appropriate for the objects to be implemented.

Design Principles

1. Design process should not suffer from tunnel vision
2. Design should be traceable to the analysis model
3. Design should not reinvent the wheel
4. Design should exhibit uniformity and integration
5. Design should be structured to accommodate change
6. Design is not coding and coding is not design
7. Design should be reviewed to minimize conceptual errors

Software Design Model



Data Design

The primary activity during data design is to select logical representations of data objects identified during the requirements definition and specification phase. The selection process may involve algorithmic analysis of alternative structures in order to determine the most efficient design or may simply involve the use of a set of modules that provide the operations upon some representation of an object.
[Wasserman]

- Data design transforms the information domain model created during analysis into the data structures that will be required to implement the software.
- The data objects and relationships defined in ERD and the detailed data content depicted in the data dictionary provide the basis for the data design activities.

Architectural Design

- Objective is to develop a modular program structure and represent the control relationships between modules
- Derived from the system specification, the analysis model, and the interaction of subsystems defined within the analysis model.

Interface Design

- Combines program and data structure by defining interfaces that allows data to flow throughout the program
- The interface design describes how the software communicates within itself, with systems that interoperate with it and with humans who use it.
- Data and control flow diagrams provides information required for interface design

Software Design Fundamentals

“The beginning of wisdom for a computer programmer is to recognize the difference between getting a program to work and getting it *right*.” [Jackson]

- Abstraction
- Refinement
- Modularity
- Software Architecture
- Control Hierarchy
- Data Structure
- Software Procedure
- Information Hiding

Design Fundamentals

- Abstraction
  - Is one of the fundamental ways that can be used to cope with complexity.
  - Levels of detail/language used to describe a problem
    - Highest level
    - Lower level
    - Lowest level
  - Types:
    - Procedural abstraction
    - Data abstraction
    - Control abstraction

The psychological notion of “abstraction” permits one to concentrate on a problem at some level of generalization without regard to irrelevant low level details; use of abstraction also permits one to work with concepts and terms that are familiar in the problem environment…
[Wasserman]

Design Fundamentals (cont.)

- Refinement
  - Top-down strategy
  - Abstraction and Refinement are complimentary

In each step, one or several instructions of the given program are decomposed into more detailed instructions. This successive decomposition or refinement of specification terminates when all instructions are expressed in terms of any underlying computer or programming language. *[Wirth]*

Design Fundamentals (cont.)

- Modularity
  - Divide software into separate components often called modules that are integrated to solve problem requirements

Criteria to evaluate the Design methods

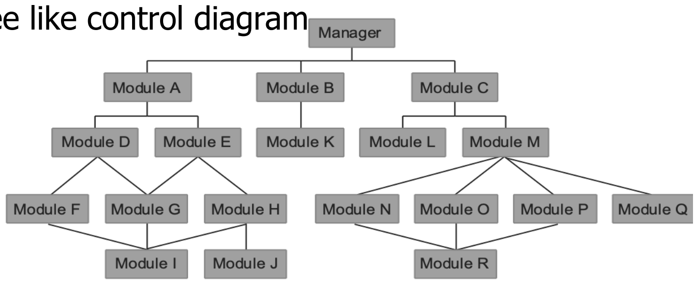
- Modularity Decomposability
- Modular Composability
- Modular Understandability
- Modular Continuity
- Modular Protection

Design Fundamentals (cont.)

- Software Architecture
  - The hierarchical structure of program components, the manner in which these components interact and & the structure of data that are used by the components.
- Properties of an architectural design
  - **Structural properties**
    - Defines the components of the system, and the manner in which those components are packaged and interact with one another.
  - **Extra-functional properties**
    - Should address how the design architecture achieves requirements for performance, capacity, reliability, security, adaptability and other system characteristics.
  - **Families of related properties**
    - Identify the repeatable patterns that are commonly encountered in the design of similar system. The design should have the ability to reuse architectural building blocks.

Design Fundamentals (cont.)

- Control Hierarchy/Program Structure
  - Organization of modules that implies a hierarchy of control
  - Depth, width, fan-out, fan-in
  - - Tree like control diagram



Design Fundamentals (cont.)

- Data Structure
  - Logical representation of the relationship among individual data elements
  - Scalar, vector, array, linked list, stacks etc
- Software Procedure
  - Processing details of each module
  - Precise specification includes sequence of events, decision points, repetitive operations, data organization

Design Fundamentals (cont.)

- Information Hiding
  - Modules should be "characterized by design decisions that each hides from all others"
  - Modules are designed so that information within a module is inaccessible to other modules with no need for the information
  - Defines and enforces access constraints

Modular Design

- Benefits
  - Reduces complexity
  - Facilitates change
  - Easier to develop
  - Easier to maintain and test
  - Easier implementation afforded by parallel development

Functional Independence

- Design software so that each module addresses a specific sub-function of requirements and has a simple interface when viewed from other parts of the program structure
  - Benefits
    - Easier to develop
    - Easier to maintain & test
  - Measures of Independence
    - Cohesion
      - measure of relative functional strength of a module
      - a cohesive module should (ideally) do just one thing/single task
    - Coupling
      - measure of the relative interdependence among modules