CS 315 - Oct 12, 2015

Chapter 14: Design

- Analysis Model to Design Model
  - Usage based design is mapped onto a structured, well-defined model

- Design and Abstraction
  - Classical Design Activities
    - Architectural Design
      - Input: Specifications
      - Output: Modular Decomposition
      - Architectural Styles
        - Data/database center
          - Clients connect to central repository of information
        - Pipe and Filter
          - Produce an output with a given input
          - Examples
            - Image Manipulation
            - Encryption
            - Transfer protocols
          - Special Case: Batch Sequential
            - Everything happens a certain sequence in a straight line
        - Layered
          - Layers of functionality above a core layer
          - Operating Systems follow this model
          - Layers only have to interact with the layer below them
        - Call and Return
          - Traditional programming
          - Break computations into tiny pieces, calculate results, then combine them
        - Service Oriented
          - Good for distributed business systems
          - Interface/Client tier, Business Logic/Application tier, database tier
    - Detailed Design
      - Each module is designed

        - Specific algorithms, data structures

- Object Oriented Design
    - Aim
        - Design the product in terms of the classes extracted during object oriented analysis

    - Two Steps
        1. Complete the class diagram
            - Determine the formats of the attributes
                - To minimize rework, **never** add an item to a UML diagram until strictly necessary

            - Principle A:
                - Information Hiding

            - Principle B:
                - If an operation is invoked by many clients of an object, assign the method to the object, not the clients

            - Principle C:
                - Responsibility-Driven Design
                - What actions is this object responsible for?
                - What information does this object share?

            - Assign each method, either to a class or to a client that sends a message to an object of that class

        2. Perform the detailed design

    - Package Diagram
        - Package is a group of related classes
        - Primary mechanism to indicate encapsulation in UML

- The Design Workflow
    - Summary
        - The analysis workflow artifacts are iterated and integrated until the programmers can utilize them

    - Decision to be made
        - Implementation Language
        - Reuse
        - Portability

    - The idea of decomposing a large workflow into independent smaller (packages) is

carried forward to the design workflow

- The objective is to break up the upcoming implementation workflow into manageable pieces
    - Subsystems

- It does not make sense to break up small systems into subsystems, the entire system is small enough already
- Why the product is broken into subsystems:
    - It is easier to implement a number of smaller subsystems than one large system
    - If the subsystems are independent, they can be implemented by programming teams working in parallel
        - The software product as a whole can then be delivered sooner

    - The *architecture* of a software product includes
        - The various components
        - How they fit together
        - The allocation of components to subsystems

    - The taks of designing the architecture is specialized
        - It is performed by a *software architect*

    - The architect needs to make *trade-offs*
        - Every software product must satisfy its functional requirements (the use cases)
        - It also must satisfy its nonfunctional requirements, including
            - Portability
            - Reliability
            - Robustness
            - Maintainability
            - Security

        - It must do all these things within budgte and time constraints

    - The architect must assist the client by laying out the trade-offs
    - It is usually impossible to satisfy all the requirements, functional and nonfunctionl, within the cost and time constraints
        - Some sort of compromises have to be made

    - The clients has to
        - Relax some of the requirements
        - Increase the budget; and/or
        - Move the delivery deadline

    - The architecture of a software product is critical

- The requirements workflow can be fixed during the analysis workflow
- The analysis workflow can be fixed during the design workflow
- The design workflow can be fixed during the implementation workflow
- **But there is no way to recover from a suboptimal architecture**
  - The architecture must immediately be redesigned