Software Architecture

Software Decomposition
- Subsystem
- Partition
- Software Design Quality
  - Coupling
  - Cohesion

What is Subsystem
- Collection of software elements such as software modules and their relations
- The subsystem boundary is defined explicitly
- The dependency relations among subsystems are defined
- Using “Partition” and “Layer” techniques to minimize the dependency relations

Sample of Subsystem Decomposition

Partitions and Layers
- Partitioning and layering are techniques to achieve low coupling
- A large system is usually decomposed into subsystems using both, layers and partitions
- Partitions vertically divide a system into several independent (or weakly-coupled) subsystems that provide services on the same level of abstraction
- A layer is a subsystem that provides subsystem services to a higher layers (level of abstraction)
  - A layer can only depend on lower layers
  - A layer has no knowledge of higher layers

Subsystem Decomposition into Layers
- Subsystem Decomposition Heuristics:
  - No more than 7+/−2 subsystems
  - More subsystems increase cohesion but also complexity (more services)
  - No more than 4+/−2 layers, use 3 layers (good)
Closed Architecture (Opaque Layering)
- Any layer can only invoke operations from the immediate layer below
- Design goal: High maintainability, flexibility

Open Architecture (Transparent Layering)
- Any layer can invoke operations from any layers below
- Design goal: Runtime efficiency

Coupling and Cohesion
- Goal: Reduction of complexity while change occurs
- Cohesion measures the dependence among modules
  - High cohesion: The modules in the subsystem perform similar tasks and are related to each other (via associations)
  - Low cohesion: Lots of miscellaneous and auxiliary modules, no associations

Coupling measures dependencies between subsystems
- High coupling: Changes to one subsystem will have high impact on the other subsystem (change of model, massive recompilation, etc.)
- Low coupling: A change in one subsystem does not affect any other subsystem
- Subsystems should have as maximum cohesion and minimum coupling as possible:
  - How can we achieve high cohesion?
  - How can we achieve loose coupling?

Sample of Cohesion
- Subsystem A
  - M1
  - M2
  - M3
  - M4
  - M5
- Subsystem B
  - M1
  - M2
  - M3
  - M4
  - M5

High Cohesion
Low Cohesion

Sample of Coupling
- Subsystem A
  - M1
  - M2
  - M3
  - M4
  - M5
- Subsystem B
  - M1
  - M2
  - M3
  - M4
  - M5

High Coupling
Low Coupling
Software Architecture Design

- Software Architectural Styles
  - Client/Server
  - Peer-to-Peer
  - Repository
  - Model/View/Control
  - Pipe and Filter

Client/Server Architectural Style

- One or many servers provides services to instances of subsystems, called clients.
- Client calls on the server, which performs some service and returns the result
- Client knows the interface of the server (its service)
- Server does not need to know the interface of the client
- Response in general immediately
- Users interact only with the client

Peer-to-Peer Architectural Style

- Generalization of Client/Server Architecture
- Clients can be servers and servers can be clients
- More difficult because of possibility of deadlocks

Model/View/Controller

- Subsystems are classified into 3 different types
  - Model subsystem: Responsible for application domain knowledge
  - View subsystem: Responsible for displaying application domain objects to the user
  - Controller subsystem: Responsible for sequence of interactions with the user and notifying views of changes in the model
- MVC is a special case of a repository architecture:
  - Model subsystem implements the central data structure, the Controller subsystem explicitly dictates the control flow

Software Architectural Styles

- Subsystem decomposition
  - Identification of subsystems, services, and their relationship to each other.
  - Specification of the system decomposition is critical.
- Patterns for software architecture
  - Client/Server
  - Peer-To-Peer
  - Repository
  - Model/View/Controller
  - Pipes and Filters

Problems with Client/Server Architectural Styles

- Layered systems do not provide peer-to-peer communication
- Peer-to-peer communication is often needed
- Example: Database receives queries from application but also sends notifications to application when data have changed
**Repository Architectural Style (Blackboard Architecture)**

- Subsystems access and modify data from a single data structure.
- Subsystems are loosely coupled (interact only through the repository).
- Control flow is dictated by central repository (triggers) or by the subsystems (locks, synchronization primitives).

![Diagram showing subsystems and repository interaction](attachment:image1)

**Pipe and Filter**

- Subsystems process data received from a set of inputs and send results to other subsystems via a set of outputs.
- The subsystems are called “Filters.”
- The associations between the subsystems are called “Pipes.”

![Diagram showing pipe and filter](attachment:image2)

**Sample of Subsystem Decomposition**

- IPO2
- IP1
- IIV1
- IGR1
- IP01
- IP02
- IGR2

Decomposing a set of Subsystems

![Diagram showing subsystem decomposition](attachment:image3)

**Sample of Subsystem with Interfaces**

- IPO2
- IP1
- IGR1

Decomposing a set of Subsystems

![Diagram showing subsystem with interfaces](attachment:image4)

**Sample of Hierarchy of Subsystems**

- Bank System
- Finances
- Inventory
- Purchasing
- Vendor Maintenance

![Diagram showing hierarchy of subsystems](attachment:image5)
Define a Component Diagram

Deployment of Client-Server

Deployment of Web Application

Tracing from Use Cases to Design

Tracing from Use Cases to Design