System Design & Software Architecture

Jonathan I. Maletic, Ph.D.
Department of Computer Science
Kent State University
A system is decomposed into subsystems.
System Decomposition

- A subsystem provides a set of services to the system.
- A set of related operations that share a common purpose
- The set of services available to other systems form the subsystem interface
- Application Programmer Interface (API) includes name of operations, parameters/types, and return types
- System design focuses on defining services
(sub) System Model

System

Part * 

Class

Subsystem +service()

#parts

1
Software Architecture

• Shaw & Garlan ’ 96

• The use of standard patterns and styles of design is pervasive in many engineering disciplines

• What standard style are used in software at the architectural level?
Architectural Style

- An *architectural style* defines a family of systems in terms of a pattern of structural organization
  - Components (e.g., client, server, DB)
  - Connectors (e.g., procedure call, pipe, event broadcast)
- Vocabulary of components and connectors
- Constraints on how they are combined
Common Architectural Styles

• Dataflow systems
  – Pipe and filter
  – Batch Sequential

• Virtual machines
  – Rule based systems
  – Interpreters

• Repository
  – Databases
  – Hypertext systems
  – Blackboards

• Independent components
  – Peer-to-Peer
  – Client Sever
  – Model / View / Controller
  – Event systems

• Call and return systems
  – Main program and subroutine
  – Layered Systems
  – Object oriented systems
Pipe and Filter Architecture

- Subsystems are called filters and associations between the filters are called pipes.
- Filters only know the content and format of data being received and produced – nothing about the other filters in the system.
- Filters are executed concurrently and synchronization is done via pipes.
- Very reconfigurable.
Pipe and Filter Example

- Unix shell
- `ps auxwww | grep maletic | sort | more`
Batch Sequential Architecture

- Small number of large stand alone subsystems
- Must be executed in a fixed sequential ordering (batch)
- Typically work on large flat files, transforming the file into a new format or ordering so the next subsystem can work on the data
- Subsystems are tightly coupled through the shared file
- No real time feedback, no concurrency
Batch Sequential Example

Subsystem

Validate
Sort
Update
Report

Validate → Sort → Update → Report
Layered Architecture

- A hierarchical decomposition of a system into subsystems (layers) with each providing a higher level of services provided from lower level subsystems
- Closed architecture – each layer can only depend on the layer(s) immediately below
- Open architecture – each layer can access any layer below
Open Architecture: Motif Library

Diagram:
- xlib
- Xt
- Motif
- Application

Connections:
- xlib to Xt
- Xt to Motif
- Motif to Application
Repository Architecture

- Subsystems are independent and interact by a central repository
- Examples: Payroll or banking system, Modern IDE/Compiler, Blackboard
Repository Example
Model / View / Controller

- **Subsystems**
  - Model subsystems are responsible for maintaining domain knowledge.
  - View subsystems are for displaying knowledge to the user.
  - Controller subsystems manage the interactions with the user.
- Model subsystems do not depend on view or controllers.
- Changes in model state is propagated via a subscribe notify protocol.
- Examples: File system, database.
Client/Sever Architecture

• Subsystems:
  – Server provides one or more services to instances of clients
  – Clients ask for services and clients interact with users

• Information system with a central DB is an example

• Web servers (multiple servers)
Client/Server

Client

+requester

* +provider

* +service1()

* +service2()
Peer-to-Peer Architecture

• Generalization of client/server, clients can be servers and vice versa
• The control flow of each subsystem is independent from others except for synchronization of requests.

```
+-----+          +-----+
| Peer| *          | Peer |
|     |            |     |
| +service() | +provider  |
|     |            |     |
| *    | +requester  |
```
Virtual Machine Architecture

```
Virtual Machine --inst fetch 1
                  /  
WorkingMemory 1-store
                /     /  
          1-data fetch
                \     \  
          Program       Interpreter
                   |      |
                   |      |
                   |      |
                   |      |
                   |      |
```
Process Control Architecture
Event-driven Architecture

EventLoop -notifier -handler Subsystem

1 *