Lecture 13:
Midterm Review

Emad Shihab
Course Content

- Requirements
- Architectural Styles
- Architecture Recovery
- Design Patterns
- Project Scheduling
- Software Estimation
Requirements
Software Requirements

“Requirements are...a specification of what should be implemented. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system.”
(Sommerville and Sawyer 1997, Karl E. Wiegers 1999)
Where Do Requirements Come From?

Requirements come from users and stakeholders who have demands/needs.

An analyst/requirement engineer:

1. **Elicits** these demands/needs (raw requirements)

2. **Analyzes** them for consistency, feasibility, and completeness

3. **Formulates** them as requirements and write down a specification

4. **Validates** that the gathered requirements reflect the needs/demands of stakeholders
Types of Requirements

■ Functional Requirements
  – Specify the function of the system
  – F(input, system state) → (output, new state)

■ Non-Functional Requirements (Constraints)
  – Quality Requirements
    • Specify how well the system performs its intended functions
    • Reliable, Portable, etc…
  – Managerial Requirements
    • Legal responsibilities
  – Context / Environment Requirements
    • Conditions in which the system should operate
Quality Attributes
Quality Attributes

- Architects are often told:
  - “My application must be fast/secureSCALE”

We want **precise/measurable** goals
What are Quality Attributes

- Often know as –ilities
  - Performance
  - Scalability
  - Modifiability
  - Availability
  - …
Performance

Different ways to measure performance:

- **Throughput**
  - Transaction per min, Messages PM)
    - Average? (Video streaming)
    - Peak? (Bidding)

- **Response Time**
  - Delay or Latency
    - Guaranteed? (VOIP)
    - Average? (Search)

- **Deadlines**
  - Payroll task must complete by 2 AM
Scalability

‘How well a solution to some problem will work when the size of the problem increases’

- Request Load (# of simultaneous requests)
- Connections (# of simultaneous connections)
- Data size (text vs. video messages)
- Deployment (installation of new users)
Modifiability

- Modifiability measures how easy it may be to change an application.

- Architect asserts likely change scenarios.

- Some general rules:
  - Minimizing dependencies
  - Avoid ripple effects!
Availability

- The proportion of the required time the system is useable
  - E.g., 100% available during business hours

- Some general rules:
  - Eliminate single points of failure
  - Replication and failover
Security Approaches

- **Authentication**
  - Verify the identity of users

- **Authorization**
  - Access rights

- **Encryption**
  - Messages sent to/from application are encrypted

- **Integrity**
  - Contents are not altered in transit

- **Many others...**
Architectural Styles
Repository Style

Shared Data

Memory Access

Computation

Memory
Repository Style Advantages

- Efficient way to store large amounts of data
- Can easily share data
- Centralized management:
  - Backup, security, etc...
- Solutions to complex problem do not have to be preplanned
Repository Style Disadvantages

- Must agree on a data model a priori
- Difficult to distribute data
- Changing data schema is expensive
Pipe and Filter Architectural Style

filter

pipes
Pipe and Filter Advantages

- Easy to understand the overall input/output behavior of a system
- Support reuse since any two filters can be hooked together, provided they agree on the data that is being transmitted between them
- Systems can be easily maintained and enhanced - new filters can be added or old filters can be replaced
Pipe and Filter Disadvantages

- Not good choice for interactive systems, because of their transformational character

- Excessive parsing and unparsing leads to loss of performance and increased complexity in writing the filters themselves
Object-Oriented Style
Object-Oriented Advantages

- Object can change the implementation without affecting its clients
- Can design systems as collections of autonomous interacting agents
Object-Oriented Disadvantages

- Objects need to identify other objects they want to interact with
  - Contrast with *Pipe and Filter Style*
  - What if identity of an object changes?

- Objects cause side effect problems:
  - *E.g.*, *A* and *B* both use object *C*, then *B*’s effects on *C* look like unexpected side effects to *A*. 
Implicit Invocation Style

Publish-Subscribe

Event Based

Taylor et al. 2010
Implicit Invocation Advantages

- (PS) Efficient dissemination of one-way information
- Provides strong support for reuse
  - Any component can be added, by registering/subscribing for events
- Eases system evolution
  - Components may be replaced without affecting other components in the system
Implicit Invocation
Disadvantages

- (PS) Need special protocols when number of subscribers is very large

- When a component announces an event:
  - it has no idea what other components will respond to it,
  - it cannot rely on the order in which the responses are invoked
  - it cannot know when responses are finished
Layered Style

Adapted from Taylor et al. 2010

Virtual Machine

Client-Server
Layered Style Advantages

- **VM**
  - Clear dependence structure
  - Upper levels immune to changes at lower levels
  - Lower levels are independent of upper levels

- **CS**
  - Centralization of computation and data at server
  - Single powerful server can serve many clients
Layered Style Disadvantages

- **VM**
  - Having too many layers can be inefficient (may need to cross layers)
  - Not easy to divide software systems into layers

- **CS**
  - Heavy dependence on communication network
Architecture Recovery
Conceptual vs. Concrete vs. Reference

- **Conceptual architecture:**
  - shows how developers think about a system

- **Concrete architecture:**
  - shows the actual relationships in the system

- **Reference architecture:**
  - General architecture for a specific domain
Linux Architecture

Conceptual Architecture

Concrete Architecture
Why the Extra Dependencies?

- Developers avoid existing interfaces to achieve better efficiency
- Expediency
Reference Architecture Derivation Process

Reference Architecture for Web Servers

- Conceptual Architecture
  - Concrete Architecture
    - Apache

- Conceptual Architecture
  - Concrete Architecture
    - AOLServer

- Conceptual Architecture
  - Concrete Architecture
    - Jigsaw
Various parts of the architecture have to be modeled using different approaches

**View:** is a set of design decisions related to a common concern (or set of concerns)

**Concern:** is an aspect of the system that a stakeholder cares about
Architectural Views

Stakeholder: End-user
Concern: Functionality

Logical View → Development View

Scenarios

Process View → Physical View

Integrators
Performance
Scalability

System engineers
Topology
Communications
Midterm

- Friday, Oct 14, BIOSCI 1120
- 50 minutes
- 2 questions:
  - 1 short answers with 5 sub-questions
  - 1 architecture design
- 70 marks in total
Midterm

- Topics covered
  - Requirements (~15%)
  - Architectural styles (~55%)
  - Conceptual, Concrete and Reference architecture (~20%)
  - Architectural views (~10%)

- Review notes, readings, EOC questions, and class activities

- Email or come and see me if you have any questions