Material and some slide content from:
- GoF Design Patterns (book)
- Heads up Design Patterns (book)

Final Review

Reid Holmes

Arch Symptoms

- Rigidity:

  Tendency for S/W to be hard to change; small fixes cascade into large changes.

- Fragility:

  Tendency for changes to introduce new faults.

- Immobility:

  Inability to reuse components in new projects because they are too tightly bound to context.

- Viscosity:

  There is always more than one way to make a change; in high-viscosity environments the ‘easier’ change often violates the architecture.

Arch Summary

- Arch: components, connectors, topology
- Enables NFPs, reuse existing knowledge
- Essential difficulties:
  - complexity, conformity, changeability, intangibility
  - Scattering (spread) vs tangling (intertwined)
- Multiple views (diagrams):
  - component, state, physical/deployment, sequence
- Multiple stakeholders (dev, test, ops, sales, etc.)
- NFPs (often conflict, why?)
  - complexity, evolvability, scalability, portability, etc.

Arch Styles

- Client/Server: N-tier separation, clients independent
- Layered: Strong separation between layers, clear interfaces
- Pipe & Filter: Dynamic reordering, best for stream data
- Peer to Peer: No centralized control, security & dist. challenges
- Mobile Code: Dynamically move computation to other nodes
- Blackboard: Independent workers collaborate. BB hard to design
- Interpreter: Dynamic evaluation of provided language
- Event-Based: Implicit evaluation of actions, +obliv, -order
- Publish-Subscribe: Excellent broadcast mech.

Final Course Review

Intended Learning Outcomes:

"Critique an existing architecture or design."

"Differentiate how various architectural styles and design patterns enhance and degrade functional- and non-functional properties."

"Generate and justify and architecture and/or design given a collection of requirements."

"Produce and present concise and unambiguous architecture and design descriptions."

"Create and implement an architecture and design, refining it into a complete system."

ILO 1: Critique

"Critique an existing architecture or design."

So what is architecture?

"The set of principal design decisions"

- Focuses on those decisions that are hard to change once the system is built.

- Components, connectors, topology.

What are design principles?

- A set of guidelines that help improve overall software quality.

ILO 1: Critique

"Critique an existing architecture or design."

Why is architecting software hard?

- Young field, high user expectations, dependent on environment
- Essential vs incidental difficulties.
- Abstraction alone is not enough because of:
    - Complexity (non-linear), conformity (env. changes), changeability (perceived to be easy), intangibility (not constrained by physics).
- Arch drift (not captured) / erosion (violations)
- Concrete vs conceptual

What has improved complexity?

- HLL, tools and environments, development strategies (e.g., spiral model, agile), design emphasis.

ILO 1: Critique

"Critique an existing architecture or design."

Example:

> Given a layered architecture, what is one benefit and one drawback of a strict layered architecture vs. a non-strict layered architecture?

ILO 2: Differentiate

"Differentiate how various architectural styles and design patterns enhance and degrade functional- and non-functional properties."

What is an architectural style?

A set of arch design decisions that are applicable in a given context, constrain design decisions, and elicit beneficial qualities.

What is a design pattern?

A common solution to a recurring problem. Commonly sought qualities: abstraction, flexibility, modularity, and elegance.

ILO 2: Differentiate

"Differentiate how various architectural styles and design patterns enhance and degrade functional- and non-functional properties."

Abstraction: Process by which higher-level *concepts* are *derived* from concrete instances.

Separation of concerns:

- Decomposition of system into independent parts.
- Scattering: concern spread across many parts.
- Tangling: concern interacts with many parts.

ILO 2: Differentiate

"Differentiate how various architectural styles and design patterns enhance and degrade functional- and non-functional properties."

FPs: what the system does.

NFPs: efficiency, complexity, scalability, security, performance, maintainability, evolvability, dependability.

Covered Styles: client-server, layered, pipe & filter, blackboard, event-based, publish / subscribe, interpreter, mobile code, and peer to peer.

Covered Patterns: singleton, facade, decorator, composite, visitor, command, strategy, template method, adapter, and observer.

ILO 2: Differentiate

"Differentiate how various architectural styles and design patterns enhance and degrade functional- and non-functional properties."

Security: Confidentiality, Integrity, and Availability.

Security Arch Principles: least privilege, fail-safe defaults, economy of mechanism, open design, separation of privilege, least common mechanism, psychological acceptability, defence in depth.

ILO 3: Generate and Justify

"Generate and justify and architecture and/or design given a collection of requirements."

Analysis vs design: e.g., modelling 'real world' objects vs. modelling implementation objects.

Example: Apply your knowledge of architectural styles to architect a system that allows the application to dynamically shift computation resources as the system executes. Provide a component diagram. Justify your selection of architectural style.

ILO 3: Generate and Justify

"Generate and justify and architecture and/or design given a collection of requirements."

Example:

The OSI network model has been successfully leveraged for many years; how has its architecture influenced the success of the standard network stack?

ILO 4: Produce and Present

"Produce and present concise and unambiguous architecture and design descriptions."

Alternative views:

- Component diagram
- Sequence diagram
- Deployment diagram

ILO 4: Produce and Present

"Produce and present concise and unambiguous architecture and design descriptions."

Example:

- Create a component diagram for an event-based system that has a Producer component and a Consumer component connected to an event bus.
[concrete vs conceptual]

ILO 5: Create and Implement

"Create and implement an architecture and design, refining it into a complete system."

This is really about the project.