ECE444: Software Engineering
Architecture2: Patterns, and Tactics

Shurui Zhou

The Edward S. Rogers Sr. Department of Electrical & Computer Engineering
UNIVERSITY OF TORONTO
Learning Goals

• Use notation and views to describe the architecture suitable to the purpose
• Document architectures clearly, without ambiguity
• Understand the benefits and challenges of traceability.
• Understand Architecture in Agile
Architecture vs Object-level Design
Design

• Design process (analysis, design, implementation)
• Design goals (cohesion, coupling, information hiding, design for reuse, …)
• Design patterns (what they are, for what they are useful, how they are described)
Levels of Abstraction

- **Requirements**
  - high-level “what” needs to be done

- **Architecture** (High-level design)
  - high-level “how”, mid-level “what”

- **OO-Design** (Low-level design, e.g. design patterns)
  - mid-level “how”, low-level “what”

- **Code**
  - low-level “how”
Design vs. Architecture

Design Questions

• How do I add a menu item in Eclipse?
• How can I make it easy to add menu items in Eclipse?
• What lock protects this data?
• How does Google rank pages?
• What encoder should I use for secure communication?
• What is the interface between objects?

Architectural Questions

• How do I extend Eclipse with a plugin?
• What threads exist and how do they coordinate?
• How does Google scale to billions of hits per day?
• Where should I put my firewalls?
• What is the interface between subsystems?
Objects

Model
Design Patterns

- Factory
- View
- Observer
- Model/Subject
- Controller
- Command
Design Patterns
Design Patterns
Architecture
Architecture
Architecture Documentation & Views
Every engineered artifact has an architecture
Architecture Disentangled

Architecture as structures and relations
(the actual system)

Architecture as documentation
(representations of the system)

Architecture as (design) process
(activities around the other two)
Why Document Architecture?

• Blueprint for the system
  • Artifact for early analysis
  • Primary carrier of quality attributes
  • Key to post-deployment maintenance and enhancement

• Documentation speaks for the architect, today and 20 years from today
  • As long as the system is built, maintained, and evolved according to its documented architecture

• Support traceability.
Common Views in Documenting Software Architecture

• Static View
  • Modules (subsystems, structures) and their relations (dependencies, …)

• Dynamic View
  • Components (processes, runnable entities) and connectors (messages, data flow, …)

• Physical View (Deployment)
  • Hardware structures and their connections
Common Views in Documenting Software Architecture

• **Modules (Static)**

Modules are assigned specific computational responsibilities, and are the basis of work assignments for programming teams.
Architecture Is a Set of Software Structures

• **Modules** (Static)
  Modules are assigned specific computational responsibilities, and are the basis of work assignments for programming teams

• **Dynamic** (Component-and-connector **C&C**)
  Focus on the way the elements interact with each other at runtime to carry out the system’s functions.
Two views of a client-server system
Common Views in Documenting Software Architecture

• **Modules (Static)**
  Modules are assigned specific computational responsibilities, and are the basis of work assignments for programming teams

• **Dynamic** (Component-and-connector **C&C**)
  Focus on the way the elements interact with each other at runtime to carry out the system’s functions.

• **Allocation** (Physical, Deployment)
  Mapping from software structures to the system’s organizational, developmental, installation, and execution environments.
Selecting a Notation

• Suitable for purpose
• Often visual for compact representation
• Usually boxes and arrows
• UML possible (semi-formal), but possibly constraining
  • Note the different abstraction level – Subsystems or processes, not classes or objects
• Formal notations available
• Decompose diagrams hierarchically and in views
Guidelines: Avoiding Ambiguity

• Always include a legend
• Define precisely what the boxes mean
• Define precisely what the lines mean
• Supplement graphics with explanation
  • Very important: rationale (architectural intent)
• Do not try to do too much in one diagram
  • Each view of architecture should fit on a page
  • Use hierarchy
What could the arrow mean?

- Many possibilities
  - A passes control to B
  - A passes data to B
  - A gets a value from B
  - A streams data to B
  - A sends a message to B
  - A creates B
  - A occurs before B
  - B gets its electricity from A
  - ...

A → B