SE2: Introduction to Software Architecture

Reid Holmes
Architecture

- Architecture is:
  - All about communication.
  - What ‘parts’ are there?
  - How do the ‘parts’ fit together?

- Architecture is not:
  - About development.
  - About algorithms.
  - About data structures.
What is Software Architecture?

- The conceptual fabric that defines a system
  - All architecture is design but not all design is architecture.
- Architecture focuses on those aspects of a system that would be difficult to change once the system is built.
- Architectures capture three primary dimensions:
  - Structure
  - Communication
  - Nonfunctional requirements
Logical Web Architecture
Physical Web Architecture
Dynamic Web Architecture

- GET
- 200

Chrome → uwaterloo.ca
- GET
- 200

Chrome → friendster.com
- GET
- 200

Chrome → Google
- GET
- 200

- cs446.html
- scholar.html
- google.ca
Non-functional requirements

- Technical constraints: restrictions made for technical reasons

- Business constraints: restrictions made for business reasons

- Quality attributes: e.g., the ‘ilities’
  - Scalability
  - Security
  - Performance
  - Maintainability
  - Evolvability
  - Reliability/Dependability
  - Deployability
ANSI/IEEE 1471-200

“Architecture is the **fundamental organization** of a system, embodied in its **components**, their **relationships** to each other and the environment, and the principles governing its design and evolution”
Eoin Woods

“Software architecture is the set of design decisions which, if made incorrectly, may cause your project to be cancelled.”
Philippe Krutchen

“The life of a software architect is long (and sometimes painful) succession of sub-optimal decisions made partly in the dark.”
So what?

- What makes building systems so hard?
  - Young field.
  - High user expectations.
  - Software cannot execute independently.
- Incidental difficulties [Brooks MMM].
  - Problems that can be overcome.
- Essential difficulties [Brooks MMM].
  - Those problems that cannot be easily overcome.
Essential Difficulties

- Abstraction alone cannot help.
  - Complexity
    - Grows non-linearly with program size.
  - Conformity
    - System is dependent on its environment.
  - Changeability
    - Perception that software is easily modified.
  - Intangibility
    - Not constrained by physical laws.
Attacks on Complexity

- High-level languages.
- Development tools & environments.
- Component-based reuse.
- Development strategies.
  - Incremental, evolutionary, spiral models.
- Emphasis on design.
  - Design-centric approach taken from outset.