Material and some slide content from:
- Emerson Murphy-Hill
- Software Architecture: Foundations, Theory, and Practice
- Essential Software Architecture

SE2: Introduction to Software Architecture

Reid Holmes

Friday, January 11 2013.

Quick Announcement

Googler Office Hours

When: Tuesday, January 15th, 11:00am-1:30pm
Where: TC 1112

RSVP: http://goo.gl/hyQ3n

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

Structural example

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

SEI

"The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.

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 you 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.

www Example

Image from [Software Architecture: Foundations, Theory, and Practice] companion materials.

www Example

Image from [Software Architecture: Foundations, Theory, and Practice] companion materials.

www Example

Image from [Software Architecture: Foundations, Theory, and Practice] companion materials.

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. (e.g., ...)
- 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.

[TAILOR ET AL.]

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.

Architecture Analogies

- We live in them.
- We know (approximately) how they are built.
  - Requirements.
  - Blueprints (design).
  - Construction (implementation).
  - Use in practice.

The architect

- Distinctive role.
- Broadly trained.
  - Requirements, design, implementation, & use.
- Has a keen sense of aesthetics.
- Strong understanding of the domain.
  - What are these for buildings?
  - What are these for software?

[TAILOR ET AL.]

The architect

How is building architecture different from software architecture?

What common benefits can software gain from an architect that a building gets from its architect?