Topics in Software Architecture

SENG 480/580

(H. Muller)

Today: **Jens Jahnke** (jens@acm.org)

The following slides for the course introduction have been taken from a similar course of <u>Rick Kazman</u> at CMU.

Building Systems from Parts

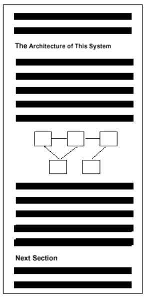
- The hype:

  "... and then we'll be able to construct software systems by picking out parts and plugging them together, just like Tinkertoys ..."

- The hard cold truth:

  It's more like having a bathtub full of Tinkertoy, Lego, Erector set, Lincoln logs, Block City, and six other incompatible kits -- picking out parts that fit specific functions and expecting them to fit together

Typical Descriptions of Software Architectures



- Descriptions of software systems often include a section on "the architecture of this system"
- These are usually informal prose plus box-and-line diagram
- Sometimes these appeal to intuition
- They have little precision, are rarely formal, and rarely analyzable
- So what good are they???

Typical Descriptions of Software Architectures

> "Camelot is based on the client-server model and uses remote procedure calls both locally and remotely to provide communication among applications and servers." [Spector 87]

> "We have chosen a distributed, object-oriented approach to managing information." [Linton 87]

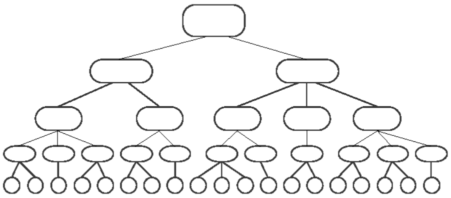
> "The easiest way to make the canonical sequential compiler into a concurrent compiler is to pipeline the execution of the compiler phases over a number of processors." [Seshadri 88]

> "The ARC network [follows] the general network architecture specified by the ISO in the Open Systems Interconnection Reference Model." [Paulk 85]

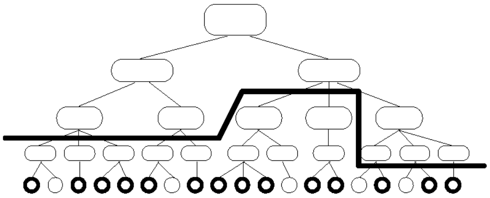
Course Objectives

- Learn to design, understand, and evaluate systems at an architectural level of abstraction.
- By the end of the course, be able to:
  - > Recognize major architectural styles in existing systems.
  - > Describe and present an architecture clearly.
  - > Design architectural alternatives for a problem and choose among them.
  - > Construct a medium-sized software system that satisfies an architectural specification, using existing definitions and development tools to expedite the task.
  - > Analyze software architectures for appropriateness.
  - > Use domain knowledge to specialize an architecture for a particular family of applications.

Software Design Levels

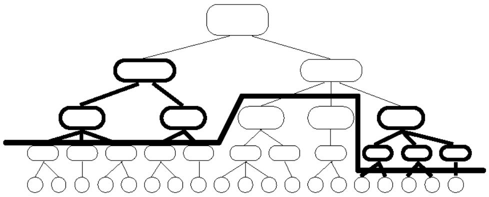


Software Design Levels: Programs



Library Reuse

Software Design Levels: Architecture



Architectural Patterns

Elements of a Complete Software System

User view of problem	User Model
Software view of problem	Requirement
Modules and connections	Architecture
Algorithms & data strs	Code
Data layouts, memory maps	Executable

Observations about Designers

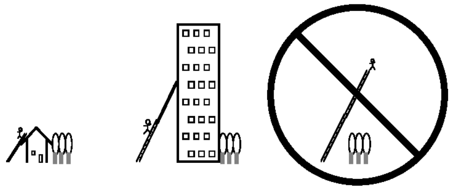
- They freely use informal patterns (idioms)
  - > Very informal, imprecise semantics
  - > Diagrams as well as prose, but no uniform rules
  - > Communication takes place anyhow
- Their vocabulary uses system-level abstractions
  - > Overall organization (styles)
  - > Kinds of components and interactions/interfaces among them
- They compose systems from subsystems
  - > Tend to think about system structure statically
  - > Often select a system organization by *default*, not by design

Software Architecture

- The architecture of a software system
  - > defines the system in terms of components and interactions among components
  - > shows correspondence between requirements and elements of the constructed system
  - > addresses system-level properties such as latency, capacity, throughput, security, availability
- An architectural definition selects
  - > Components: define the locus of computation
    - » Examples: filters, databases, objects, ADTs
  - > Connectors: mediate interactions of components
    - » Examples: procedure call, pipes, event broadcast
  - > Properties: specify info for construction & analysis
    - » Examples: signatures, pre/post conditions, RT specs

Aren’t Programming Languages Good Enough?

When orders-of-magnitude improvement are required, new technology may be necessary



Architectural Design Task

Different issues for architecture & programs

Architecture	*Programs*
interactions among parts	implementations of parts
structural properties	computational properties
declarative	operational
mostly static	mostly dynamic
system-level performance	algorithmic performance
outside module boundary	Inside module boundary

Analogy to Building Architecture

Architectural styles: Colonial, Victorian, Greek Revival

- Software system organization paradigms: pipes, layers, events

Building codes: electrical, structural, etc

- Formal specifications: functionality, capacity
- Standards (code, documentation, interfaces, etc.)

Special expertise for given style: balloon frames, slate roofs

- Domain-specific architectures
- Attribute-based architectural styles

Major Topics

1. Introduction to Software Architecture
2. Understanding the Problem Space
   - Problem frames and types
   - Applications to complex systems
3. Classical Architectural Styles
   - Dataflow systems
   - Procedure call systems
   - Event-based systems
   - Repository-oriented systems
   - Independent processes
   - Others (client-server systems, component-based architectures.)
4. Techniques/Tools
   - Architecture documentation
   - Architecture design and analysis
   - Design assistance, patterns, taxonomies
   - Notations and tools

Questions

- What is a software architecture? How is it best represented?
- What kinds of issues does software architecture address?
- Why is this a worthwhile field of study?
- How does architectural design and analysis relate to other software development activities?
- How is software architecture different than programming?

2. Understanding the Problem Space

- Understand that
  - > there are different kinds of problems
  - > different kinds of problems require different kinds of solutions
- Problem Types and Problem Frames
  - > The idea of a problem type/frame
  - > Classical problem frames
  - > The need to combine multiple frames to solve real problems
- Case study
  - > London Ambulance example to illustrate these ideas

Questions

- What kinds of problems are there? How do we generalize these?
- How can one identify the important parts of a problem frame?
- How can one recognize when a problem frame is a good/bad fit?
- When a problem frame is a bad fit, what do you do?
- How do we deal with a situation in which multiple problem frames may apply to at the same time?

3. Classical Architectural Styles

- Common architectural idioms, taxonomies, and patterns
- Issues:
  - > Detailed look at specific architectural styles
  - > Pure forms first; later heterogeneous systems
  - > Distinguishing characteristics & specializations
  - > Heuristics for choosing a style
  - > Implementation techniques
  - > Formal models and analysis
  - > Case studies

Questions

- What are the common architectural styles used by experienced system builders?
- What does it mean to be a style and what properties does each style have?
- What kinds of applications are best matched with certain architectural styles?
- Can one implement one architectural style by another?
- How can one precisely characterize an architectural style?
- What kinds of analyses are made easier when you know the style?

Subtopics

- Dataflow Systems
  - > batch sequential, pipe & filter
- Procedure Call Systems
  - > information hiding, ADTs, objects
- Event-based Systems
  - > multi-cast organization, implicit invocation
- Repository-oriented Systems
  - > blackboards, databases, client-server
- Processes
  - > communicating processes, message passing
- Others
  - > client-server systems, component-based architectures

4. Techniques

- Supporting the architectural design task
- Issues:
  - > Notations for representing architectural designs
  - > Techniques for choosing a good architecture
  - > Techniques for analyzing these representations
  - > Tools for representing architectures, carrying out these analyses, and for guiding choice of architectural style
  - > Making an effective architectural presentation
  - > Incorporating architecture into other development activities
  - > Coping with heterogeneity and mismatched parts

Subtopics

- Design assistance
  - > Concepts for choosing architectural design
  - > Classification of architectural constructs
  - > Patterns
  - > Selection and evaluation of architectures
- Notations and tools
  - > Architectural description languages and tool support
  - > Architectural specification
  - > Effective architectural representation and presentation
- Coping with legacy, evolution, business aspects
  - > Reverse engineering
  - > Architecture analysis
  - > Product Lines

Questions

- How can one connect components that were not designed to work together?
- How can one define an architectural product line?
- Is it possible to analyze an architectural description and predict the properties of the resulting system?
- How can we exploit the wisdom of virtuosos to help less-skilled engineers?
- What are the elements of an effective architectural pitch?
- What role do architectural design reviews play?

Course Outline and Organization

The Final Word

Software architecture is like teenage sex:

- It is on everyone's mind all the time.
- Everyone talks about it all the time.
- Everyone thinks everyone else is doing it.
- Almost no one is really doing it.
- The few who are doing it are:
  - > Doing it poorly.
  - > Sure it will be better next time.
  - > Not practicing it safely.