Module 4: Examples of Architectures (Linux)
Ahmed E. Hassan
Linux as a Case Study: Its Extracted Software Architecture

Paper By: Ivan T. Bowman, Richard C. Holt and Neil V. Brewster

Slides By: Jack ZhenMing Jiang
Outline

- Terminology
- Conceptual Architecture
- Concrete Architecture
- Conclusions
Terminology

- **Conceptual Architecture**
  - How developers think of a system; Relations meaningful to developers
  - Analogy: Blue Print of the House
  - By Reviewing Existing Documentation
  - Essential Relations

- **Concrete Architecture**
  - Relations that exists in a system
  - Analogy: Actual Architecture of the House
  - By Examining the Source Code
  - Implementation Specific Knowledge
The Linux Kernel

- Responsible for process, memory, and hardware device management
  - Different from the Linux System
- Linux System: 10 KLOC in 1991 to 1.5 MLOC in 1998
- The studied Linux Kernel is 800 KLOC
- Open Source
Conceptual Architecture – Top Level

Legend: Subsystem —— depends on
Conceptual Architecture – File System

Diagram:
- Memory Manager
- IPC
- Network Interface
- File System
  - System Call Interface
  - Virtual File System
  - Executable File Formats
  - Device Drivers
  - Logical File Systems
  - Buffer Cache
  - File Quota
- Process Scheduler
- Initialization
- Library

Legend:
- Kernel Subsystem
- Subsystem
- depends on →
- depends on all →

Façade Pattern
Object Oriented Style
Concrete Architecture

Extraction

Control Flow

Data Flow

Manually - Group Directory to Subsystems - Naming Conventions

Feedback

Legend:

Subsystem
 contained subsystems
 omitted subsystems
Concrete Architecture – Top Level
Concrete Architecture – File System

Legend:
- Kernel Subsystem
- Subsystem
- depends on
- depends on all

Concrete Architecture – File System

Legend:
- **Kernel Subsystem**
- **Subsystem**

- **depends on**
- **depends on all**
Concrete Architecture – Logical File System
Why Conceptual Architecture and Concrete Architecture Not Match?

- Missing Relations in Conceptual Architecture
- More Functionalities
  - For example, Process Scheduler
- Use Different Mechanisms
- Improve Efficiency by Bypassing Existing Interfaces
- Exist for Developer Expediency
  - “The read-only stuff doesn’t really belong here, but any other place is probably as bad and I don’t want to create yet another include file.”
What To Do Next?

- **Restructure to Remove Unexpected Dependencies**
  - Header Files
  - Lower Coupling

- **Refine Conceptual Architecture**
  - Not Hinder System Understanding
Conclusions

- Conceptual and Concrete Architecture for the Linux Kernel
- Similar Work Needs to Be Done for the Firefox Report
- Sample Reports:
  - Conceptual Architecture: http://plg.uwaterloo.ca/~itbowman/CS746G/a1/
  - Concrete Architecture: http://plg.uwaterloo.ca/~itbowman/CS746G/a2/