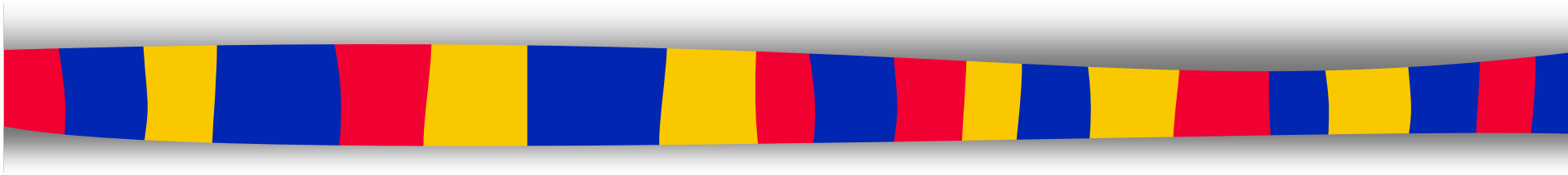
EECS 4314

Advanced Software Engineering



Topic 06:

Architecture Recovery and Analysis

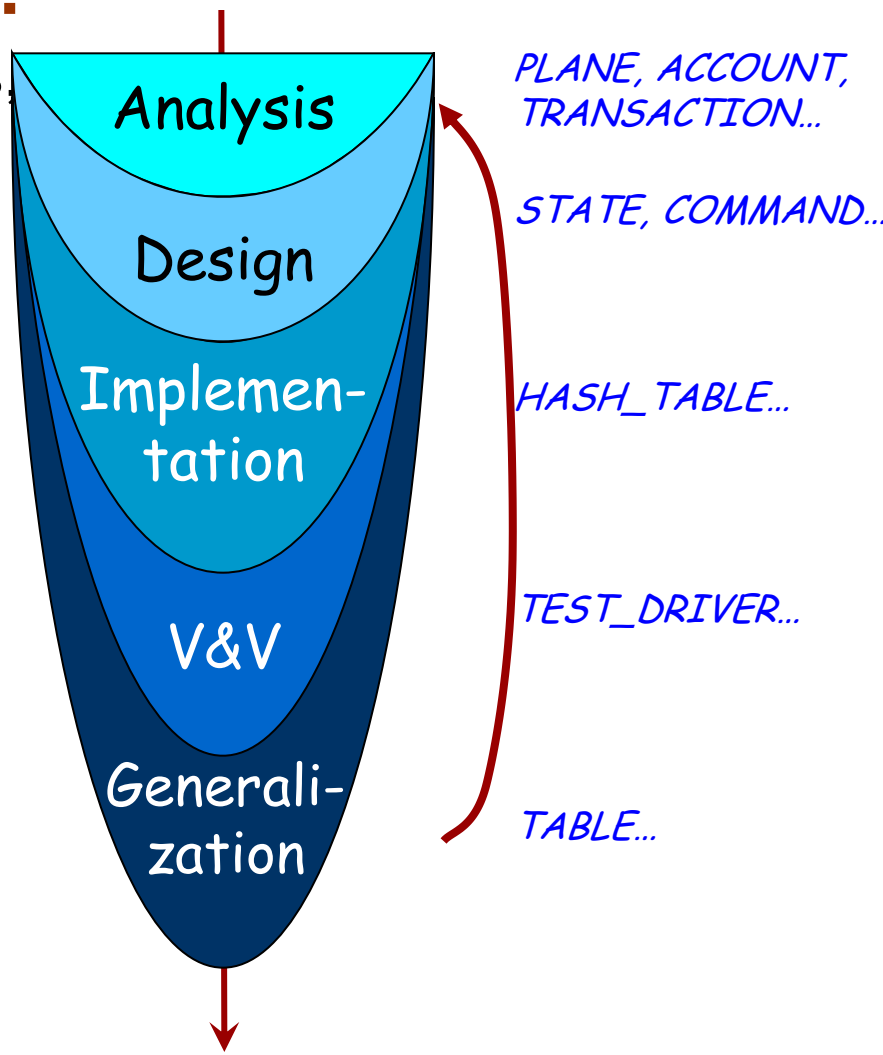
Zhen Ming (Jack) Jiang

References

- Ivan T. Bowman, Richard C. Holt, and Neil V. Brewster. 1999. Linux as a case study: its extracted software architecture. In Proceedings of the 21st international conference on Software engineering (ICSE). 1999.
- John B. Tran, Richard C. Holt: Forward and reverse repair of software architecture. CASCON 1999.
- A few extracted architecture samples (requires Java Applet)
  - http://www.swag.uwaterloo.ca/lsedit/lsapplet.html

The Eiffel model

- **Seamless development:**
  - Single notation, tools, concepts, principles throughout
  - Eiffel is as much for analysis & design as implementation & maintenance
  - Continuous, incremental development
  - Keep model, implementation and documentation consistent
  - Reversibility: go back & forth
  - Saves money: invest in single set of tools
  - Boosts quality



Why do we need architecture recovery?

- Many existing systems do not have documented architecture or they are very likely out-of-date

  - David Parnas. Software Aging. ICSE 1994

- Automated software architectural recovery can help you better understand your system to support software maintenance and evolution

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
- **Reverse Engineering**
  - Extraction of design (or architecture) from implementation and from developers

Reverse Engineering in US

- In the United States even if an artifact or process is protected by trade secrets, reverse-engineering the artifact or process is often lawful as long as it has been legitimately obtained.

- Reverse engineering of computer software in the US often falls under both contract law as a breach of contract as well as any other relevant laws. This is because most EULA's (end user license agreement) specifically prohibit it, and U.S. courts have ruled that if such terms are present, they override the copyright law which expressly permits it ...

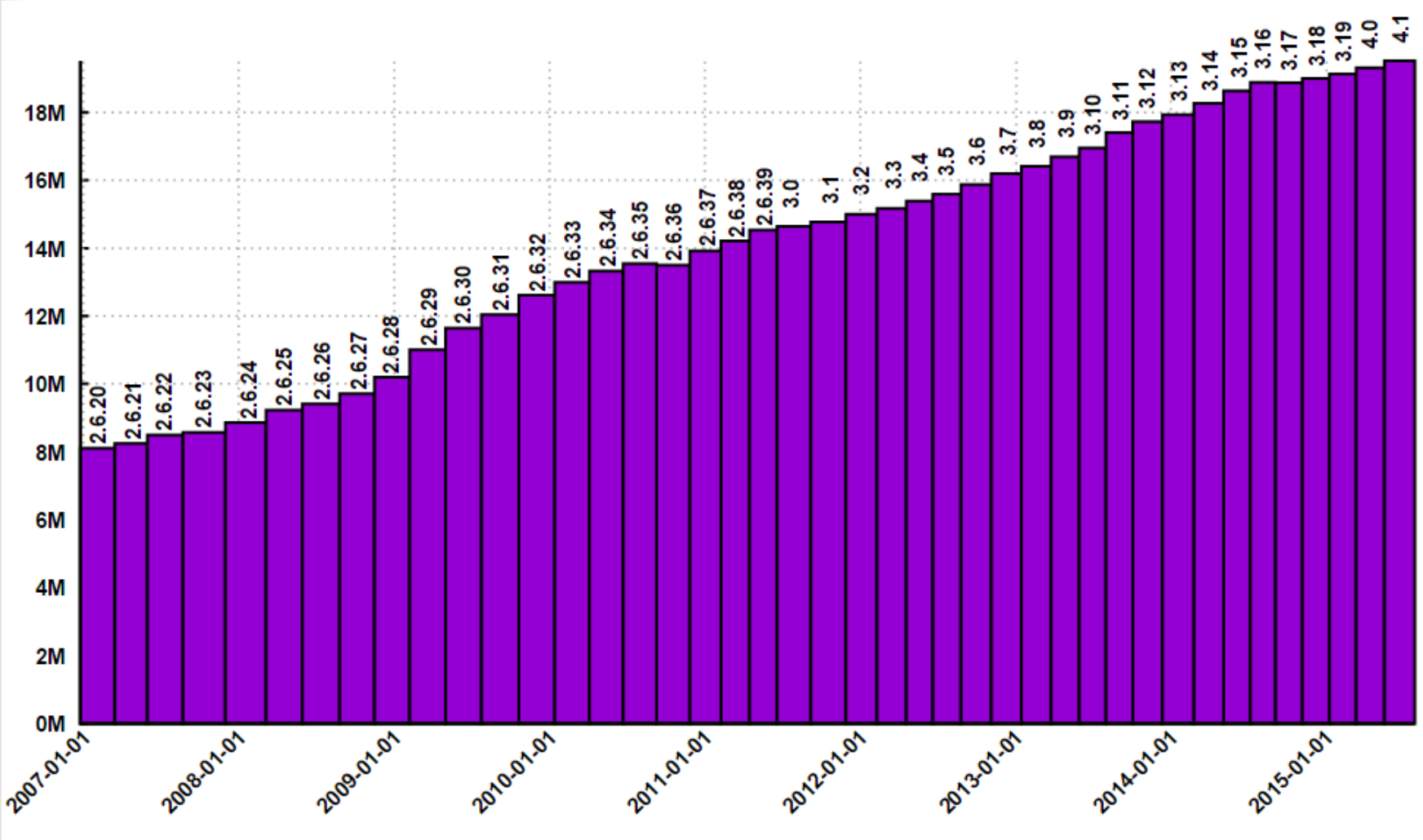
https://en.wikipedia.org/wiki/Reverse_engineering#Legality

Recovering the Linux Architecture

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

The Evolution of the Linux Kernel

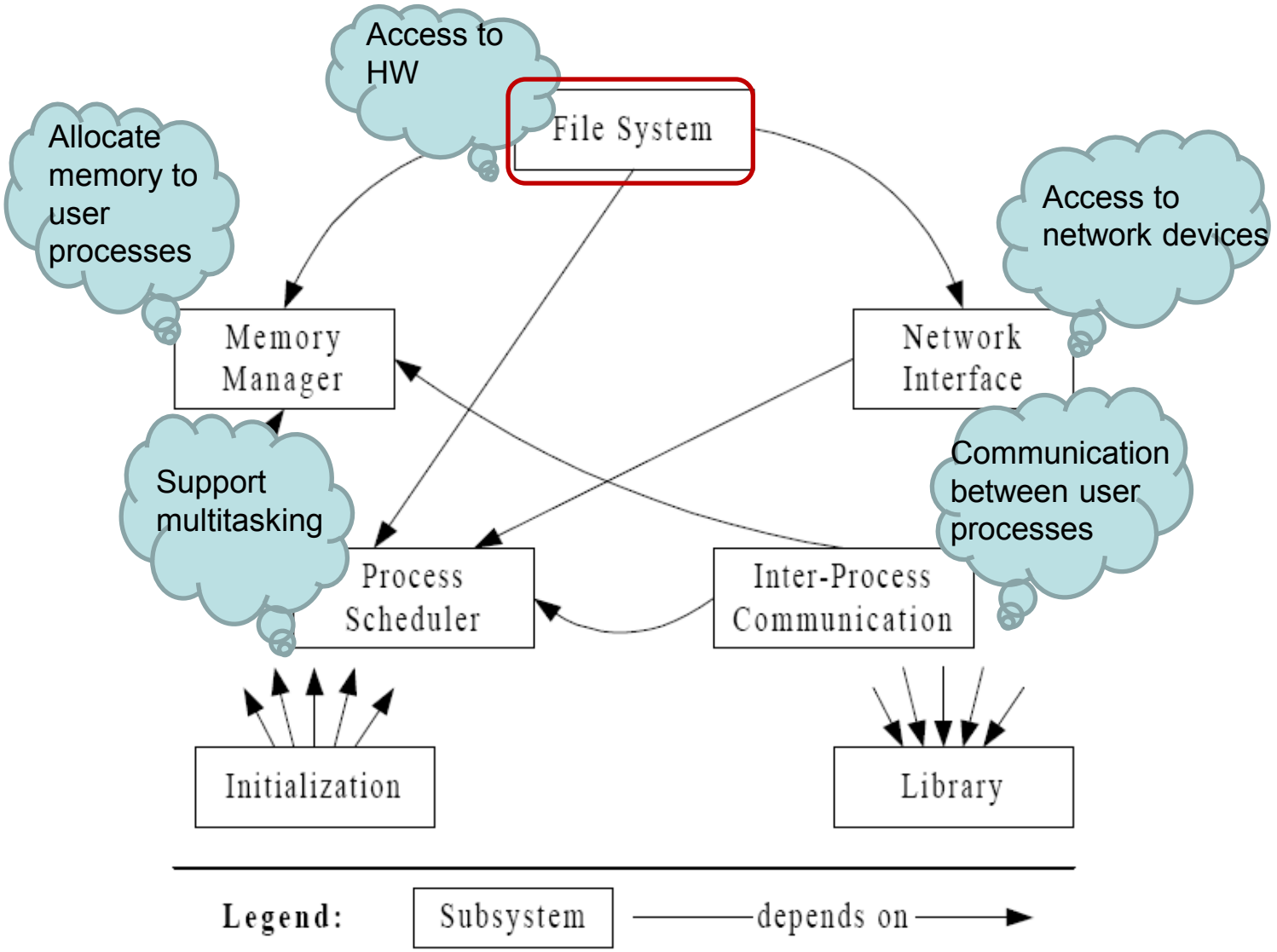


https://en.wikipedia.org/wiki/Linux_kernel

How to Obtain the Conceptual Architecture

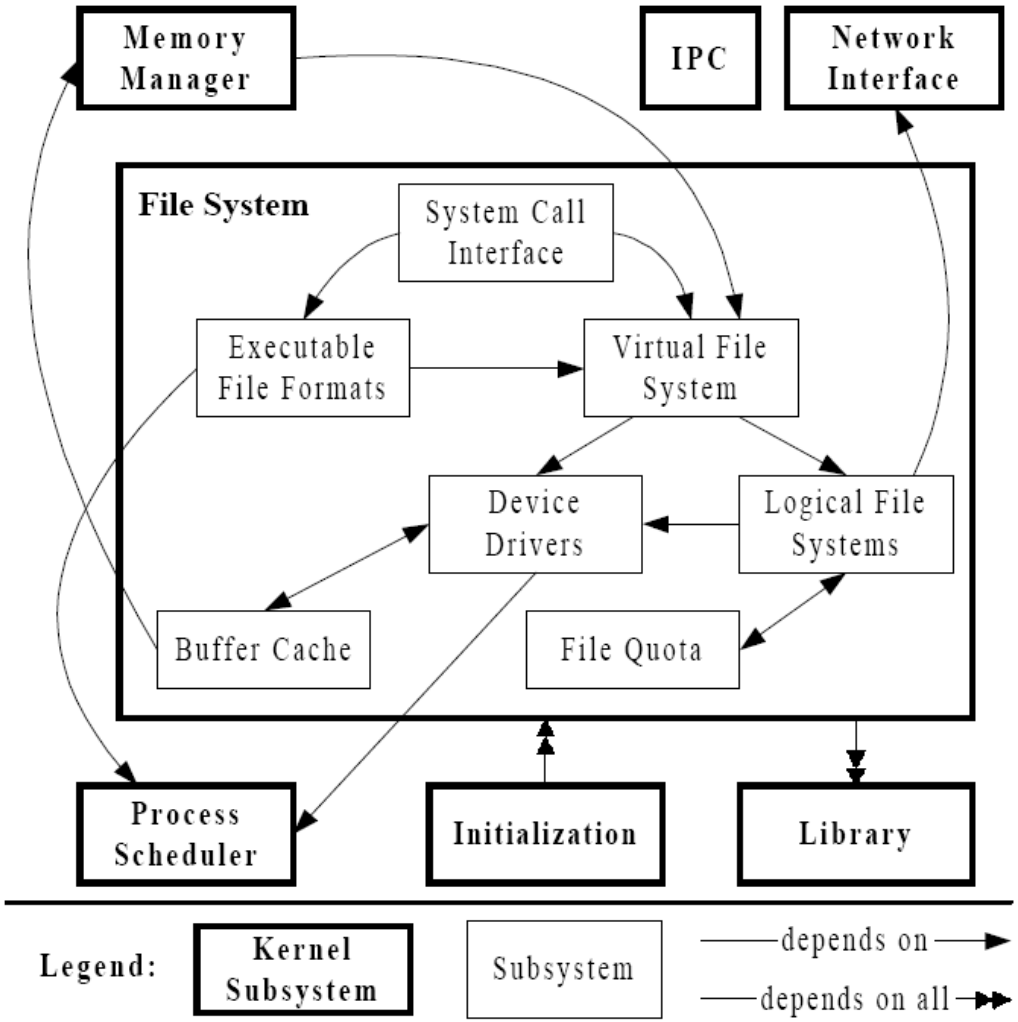
- Examining existing documentation
  - About related or similar systems
  - Documentation provided by the project itself
  - Sometimes more reading is required to iron out inconsistencies

Conceptual Architecture



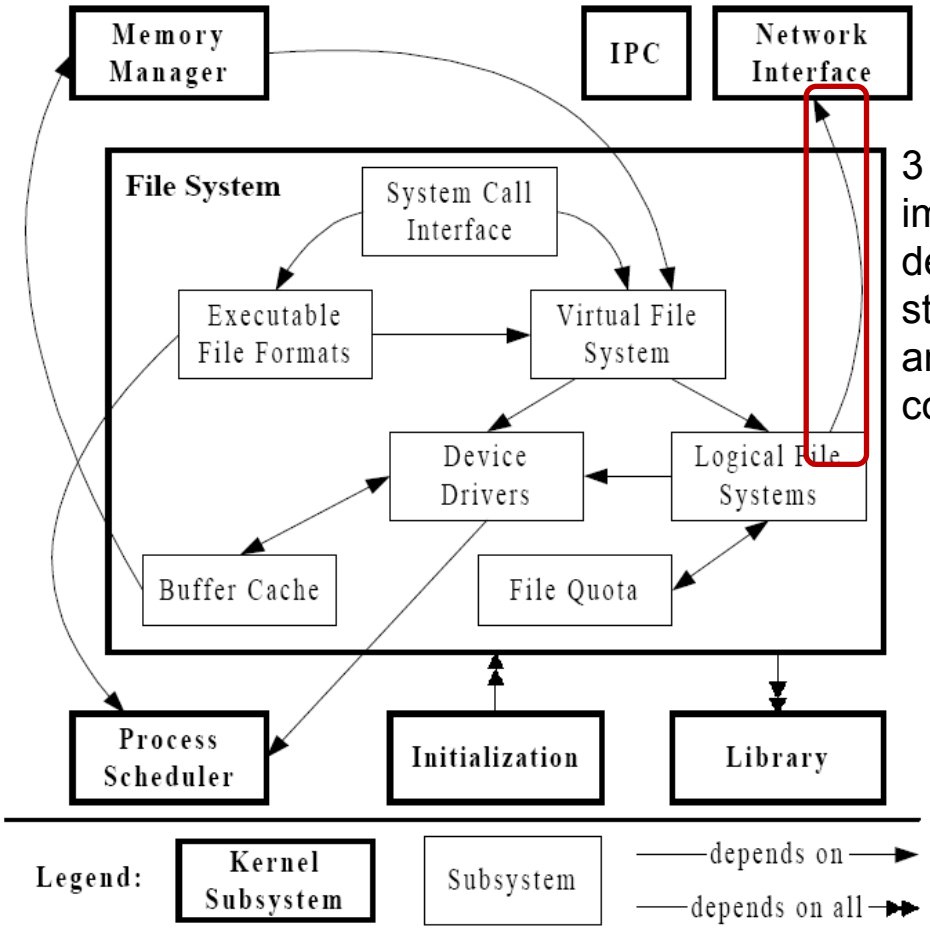
File system conceptual architecture

1. Provide access to hardware devices
2. Support different file formats and their mapping on physical locations
3. Allow programs to be stored in several executable formats



File system conceptual architecture

- Uses façade design pattern
  - Other parts of kernel use File System through a designated interface
  - Minimizes dependencies



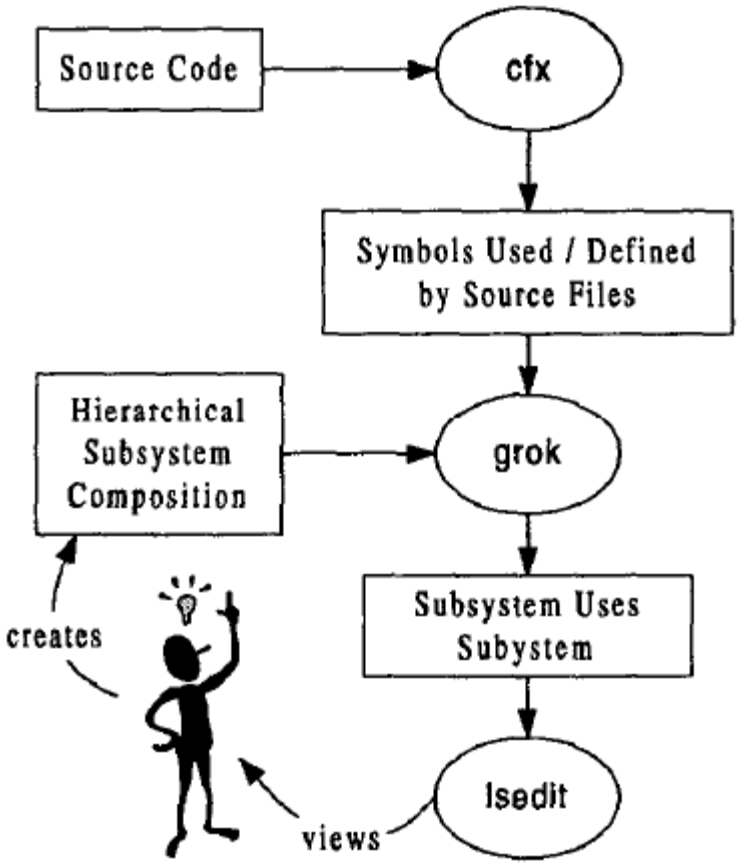
3 file system implementations depend on files stored on another computer

Concrete Architecture

Welcome to the real architecture!

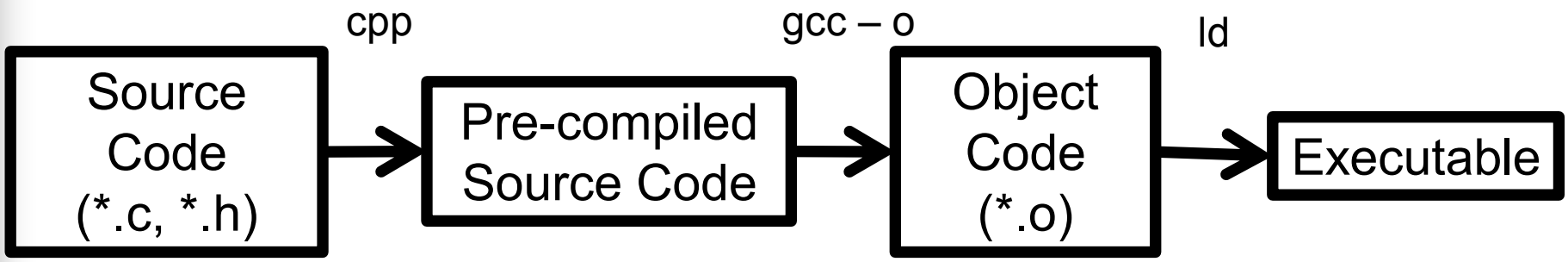
Concrete Architecture – Extraction Methodology

- Download source code
- Source code extractor (cfx) to extract relations
  - Extracts functions calls and variable access relations (e.g. function y calls function x)
  - Control flow and data flow dependencies



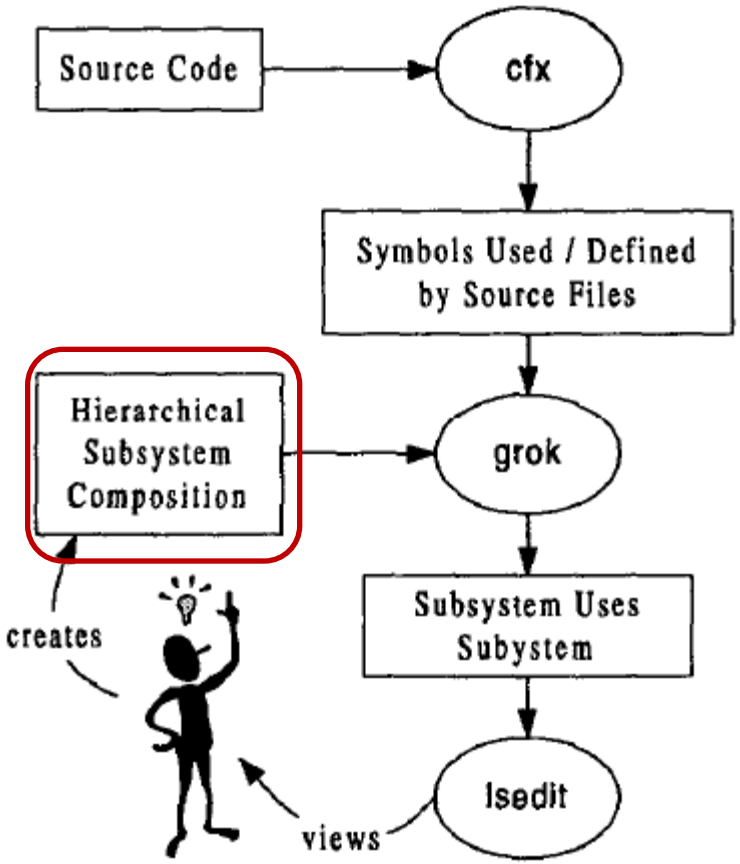
Extracting Code Dependencies

- Extracting code dependency data at different compilation phases
  - Each has its own pros and cons



Concrete Architecture – Extraction Methodology

- Use grok to determine relations between files
  - Each file was manually assigned to a subsystem
  - Used directory structure, file naming, source code comments documentation, source code (last resort)
  - Used grok to determine subsystem relations
- Used lsedit to visualize extracted system architecture

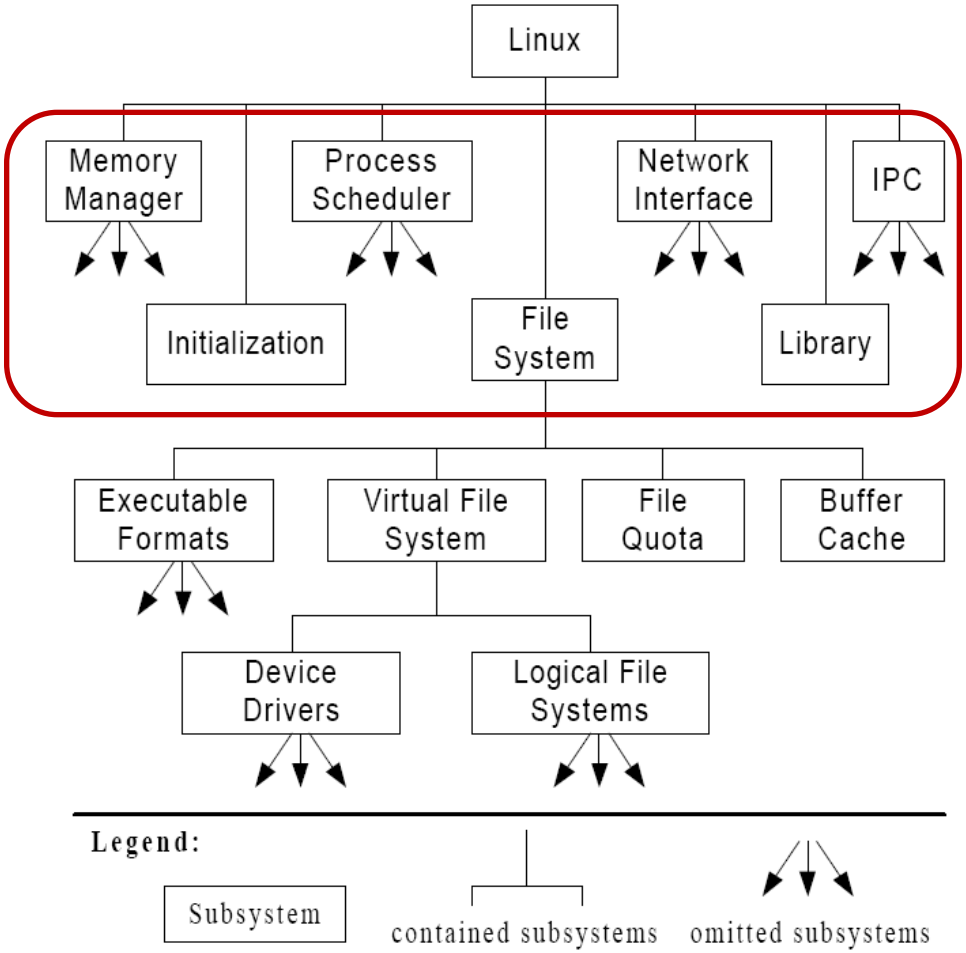


Concrete Architecture – Hierarchical Decomposition

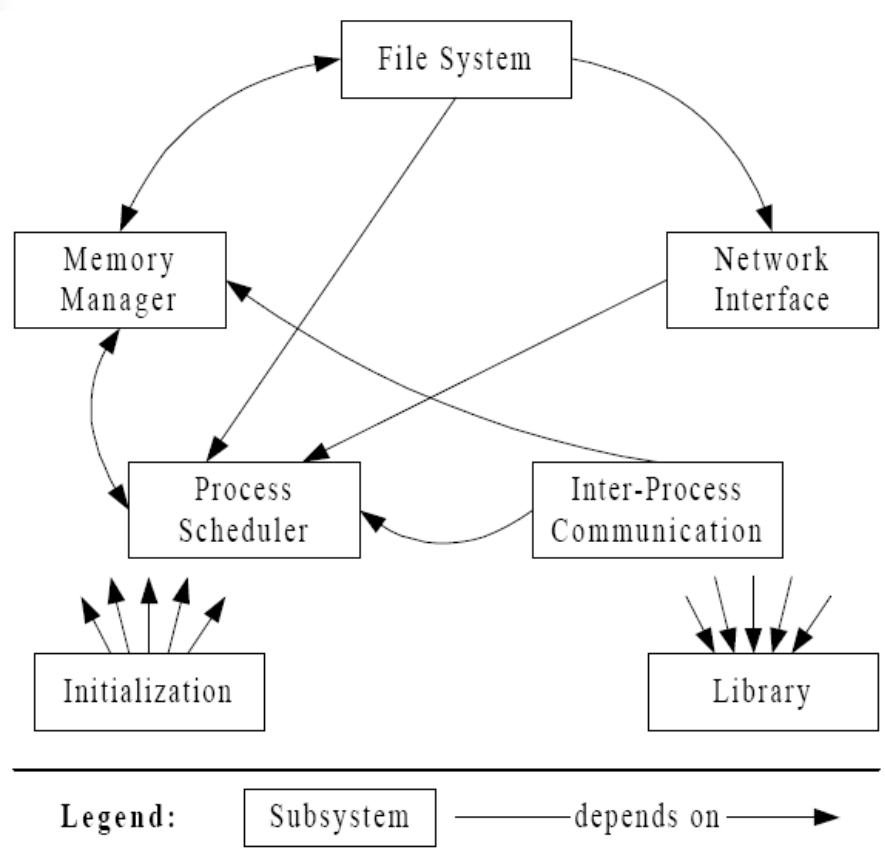
- Manually created
- Source files are assigned to subsystems
- Subsystems hierarchically assigned to (parent) subsystems
- Used to view relationships between subsystems instead of files

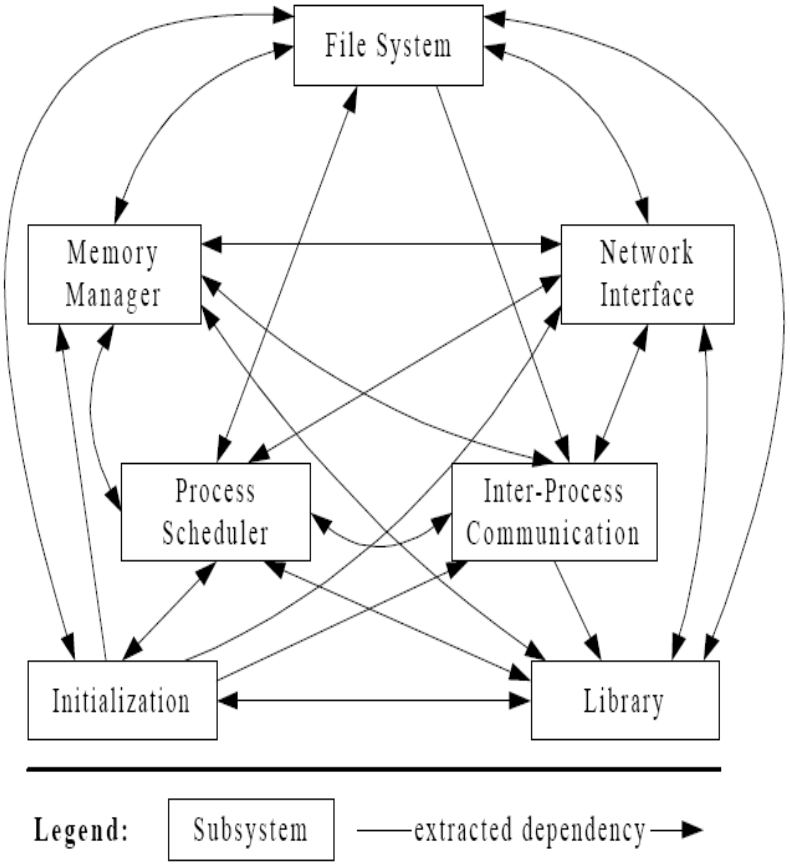
Concrete Architecture – Hierarchical Decomposition

- Seven major subsystems from conceptual arch
- Subsystems had corresponding directories in code implementation
- Used directory structure, naming convention to assign files to subsystems



Conceptual vs. Concrete Architecture



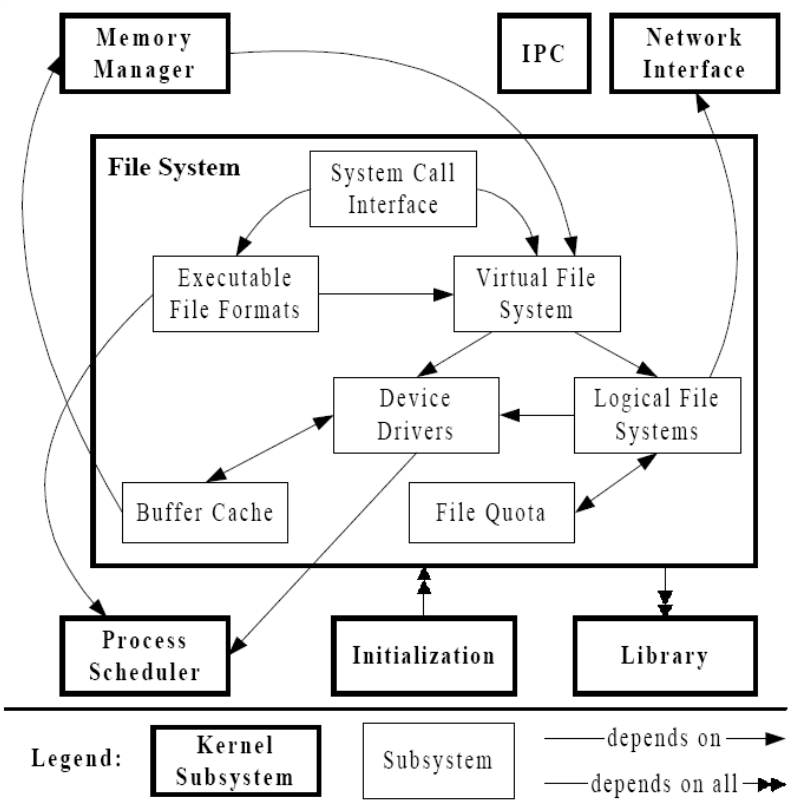


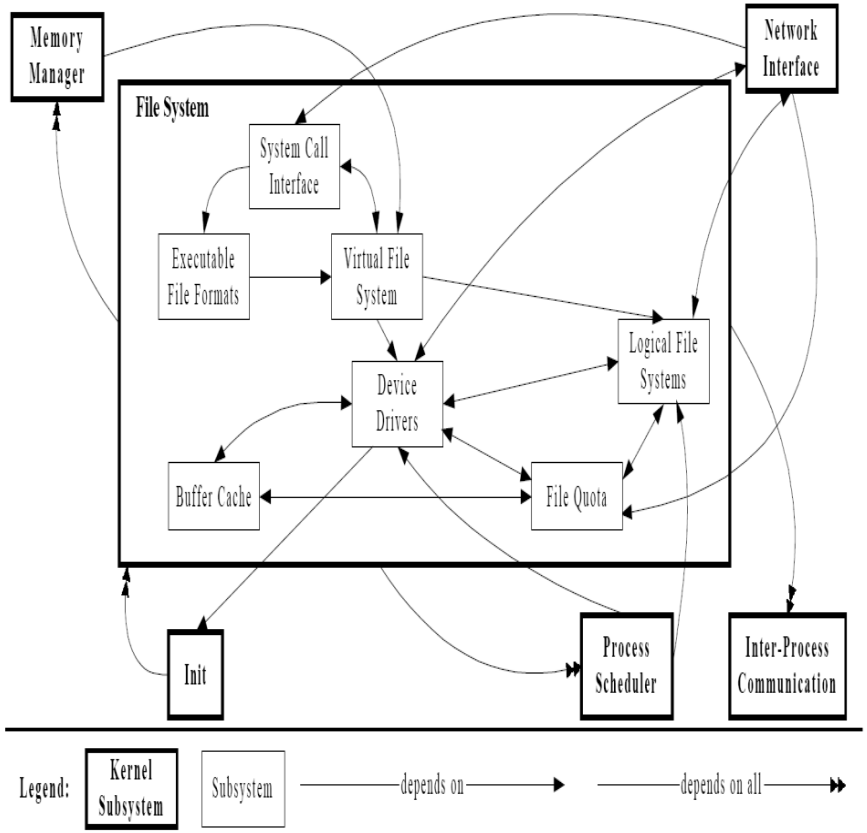
Same subsystems, more dependencies (19 vs. 37)

Why the Extra Dependencies?

- Developers avoid existing interfaces to achieve better efficiency
- Expediency
- (Almost) always expect more dependencies in concrete architecture

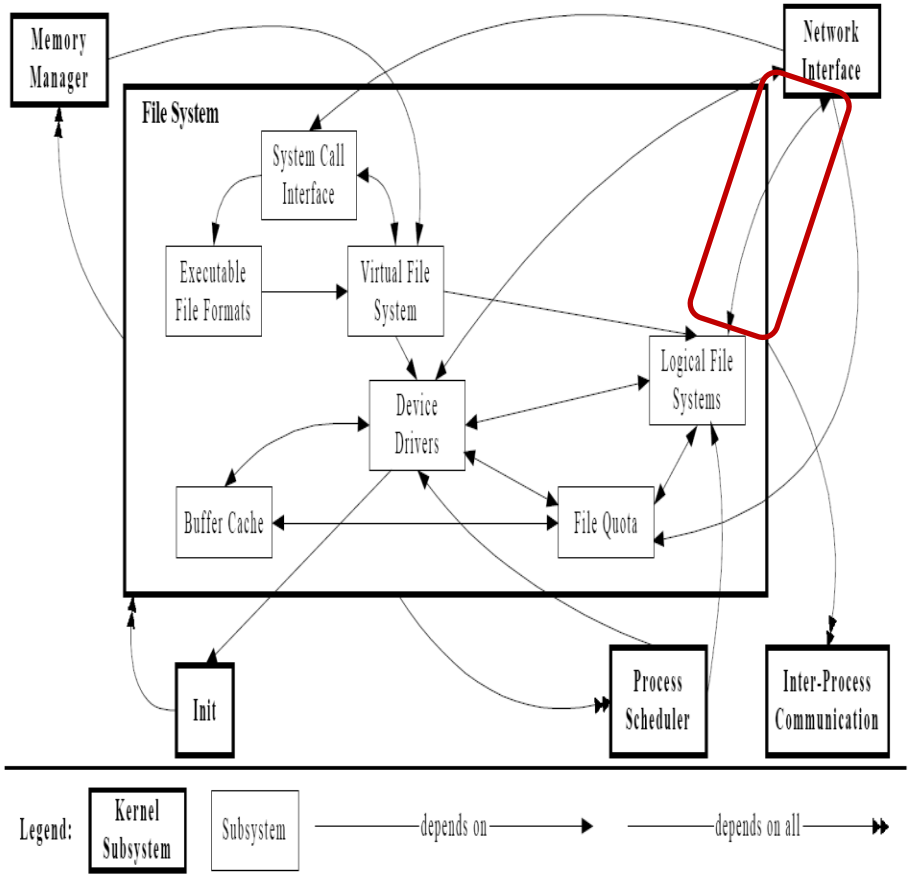
File System Concrete Architecture





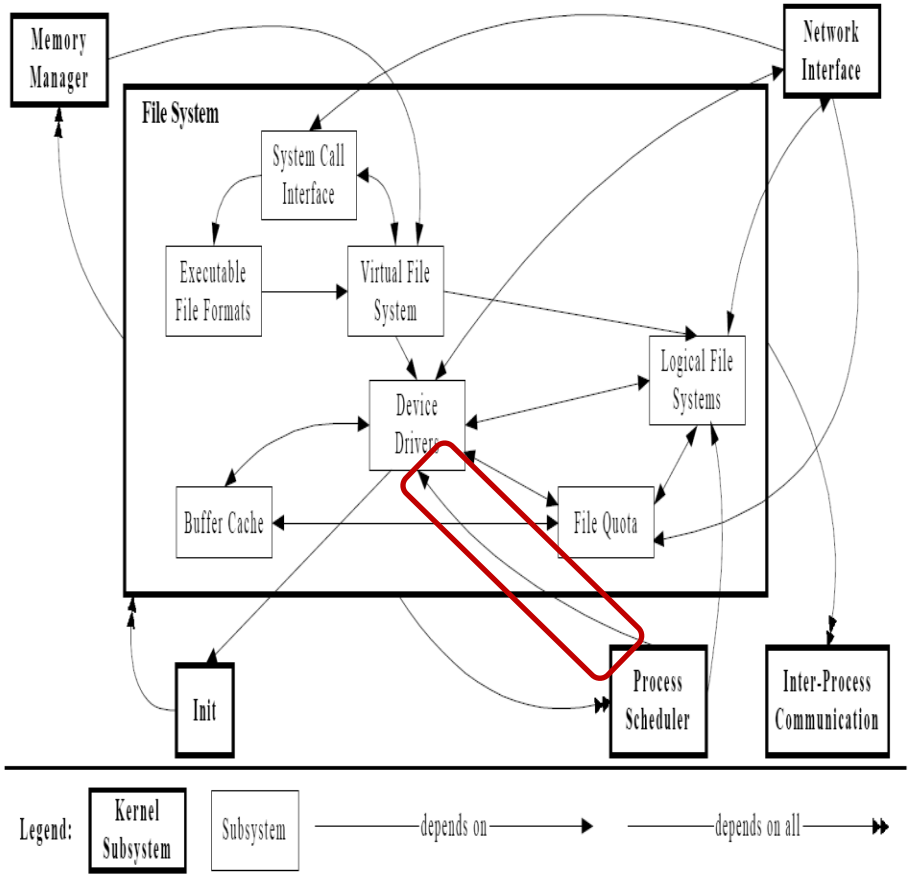
File System Concrete Architecture

- Network interface depends on logical file systems
  - File systems (NCPFS and SMBFS) that use the network were implemented by having the Network Interface directly call functions in the implementation of these logical file systems



File System Concrete Architecture

- Process Scheduler depends on Device Driver
  - (printk) in Process scheduler prints msgs to console. Calls routines implemented in device drivers subsystem



Why Conceptual Architecture and Concrete Architecture Not Match?

- Missing relations in the conceptual architecture
- More functionalities or use of 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?
- Software Reengineering

- Restructure to Remove Unexpected Dependencies
  - Header Files
  - Lower Coupling
- Refine Conceptual Architecture
  - Not Hinder System Understanding

Lessons Learned

- Human judgment is needed to determine appropriate hierarchical decomposition
- Conceptual Architecture shows control flow in one manner, but concrete showed implementation might use a different mechanism (e.g., network interface)
- Developers improved system efficiency by bypassing existing interfaces
  - "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."
- Concrete architecture should be used to refine conceptual architecture
  - Not desirable to add all relations. May hinder understanding
- Best to develop concrete and conceptual architecture in an iterative manner

Concrete Architecture Recovery Assignment

- Claim subsystem of your choice
  - “First come, first serve”
- Scripting to find out the dependency between subsystems
  - Which function(s) caused the dependency from subsystem A to subsystem B
- Iterative process to refine your concrete architecture