Software Engineering

Lecture 06: Design — an Overview

Peter Thiemann

University of Freiburg, Germany

SS 2014

The Design Phase

Programming in the large

GOAL:

transform results of analysis (requirements specification, product model) into a **software architecture**

Main Activities

- Decomposition into components
- Development of software architecture

Software Architecture

SW architecture $\hat{=}$ components, connectors, topology

- Component
  - Designated computational unit with specified interface
  - Examples: client, server, filter, layer, database
- Connector
  - Interaction point between components
  - Examples: procedure call, event broadcast, pipe
- Topology
  - Guidelines and restrictions on connecting components

Architectural Styles — Overview

Dataflow systems

Batch sequential, Pipes and filters

Call-and-return systems

Main program/subroutine, OO systems, Hierarchical layers

Independent components

Communicating processes, Event systems, Actors

Virtual machines

Interpreters, Rule-based systems

Data-centered systems (repositories)

Databases, Hypertext systems, Blackboards

(according to Shaw and Garlan, Software Architecture, Prentice Hall)

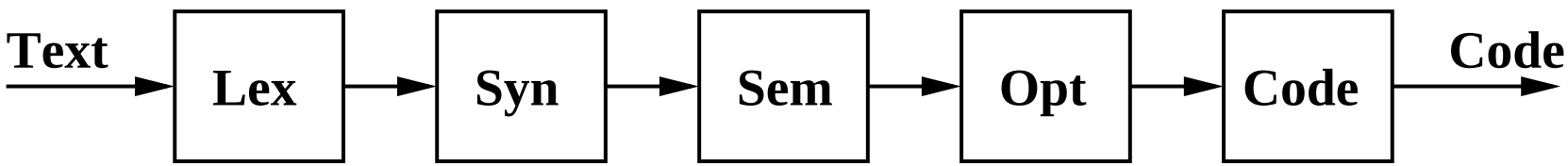
Classification of an Architectural Style

- design vocabulary—types of components and connectors
- allowable structural patterns
- underlying computational model (semantic model)
- essential invariants
- common examples of use
- advantages/disadvantages
- common specializations

Some Example Architectures

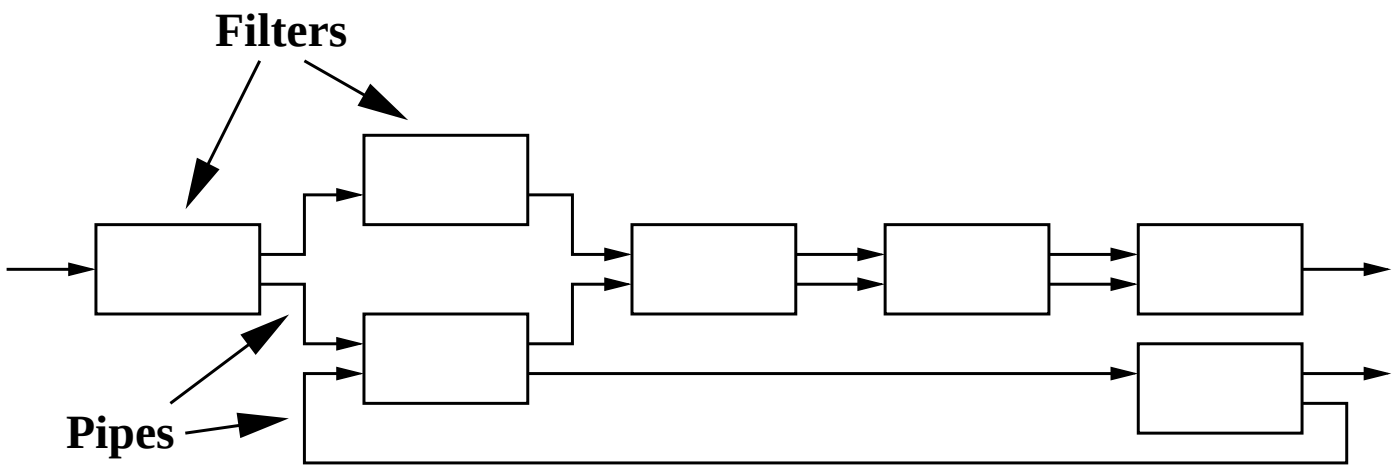
Architecture: Batch Sequential

- Separate, sequential passes
- Data passed linearly
- Each pass runs to completion before the next starts
- Example: traditional compiler architecture



Archtecture: Pipes and Filters

- Data passes continually through the system
- Each component (filter) transforms input streams to output streams incrementally
- Buffered channels (pipes) connect inputs to outputs
- Filters are independent entities
- Common specializations: pipeline (linear sequence of filters), bounded pipes, typed pipes



Properties of Pipes and Filters

+ global understanding
+ reuse
+ easy to maintain and enhance
+ specialized analysis
+ potential for concurrent execution
– interactive applications
– correpondences between streams
– common format for data transmission

Architecture: Event-based, Implicit Invocation

- Also: *reactive integration* or *selective broadcast*
- Each component may
  - announce events
  - register an interest in certain events, associated with a callback
- When event occurs, the system invokes all registered callbacks

$\Rightarrow$ Announcer of event does not know which components are registered

- Order of callback invocation cannot be assumed
- Applications: integration of tools, maintaining consistency constraints, incremental checking

Properties of Implicit Invocation

+ reuse
+ system evolution
– lack of control
– data passed through shared repository
– correctness?

Architecture: Layered Systems

- Hierarchy of system components, grouped in layers
- Inside of layer: arbitrary access between components
- Between layers
  - access restricted to lower layers: linear, strict, treeshaped
  - small interfaces
- **Advantages:** clarity, reusability, maintainability, testability
- **Disadvantages:** not always appropriate, loss of efficiency, no restrictions inside layers
- Examples: communication protocols (OSI), database systems, operating systems

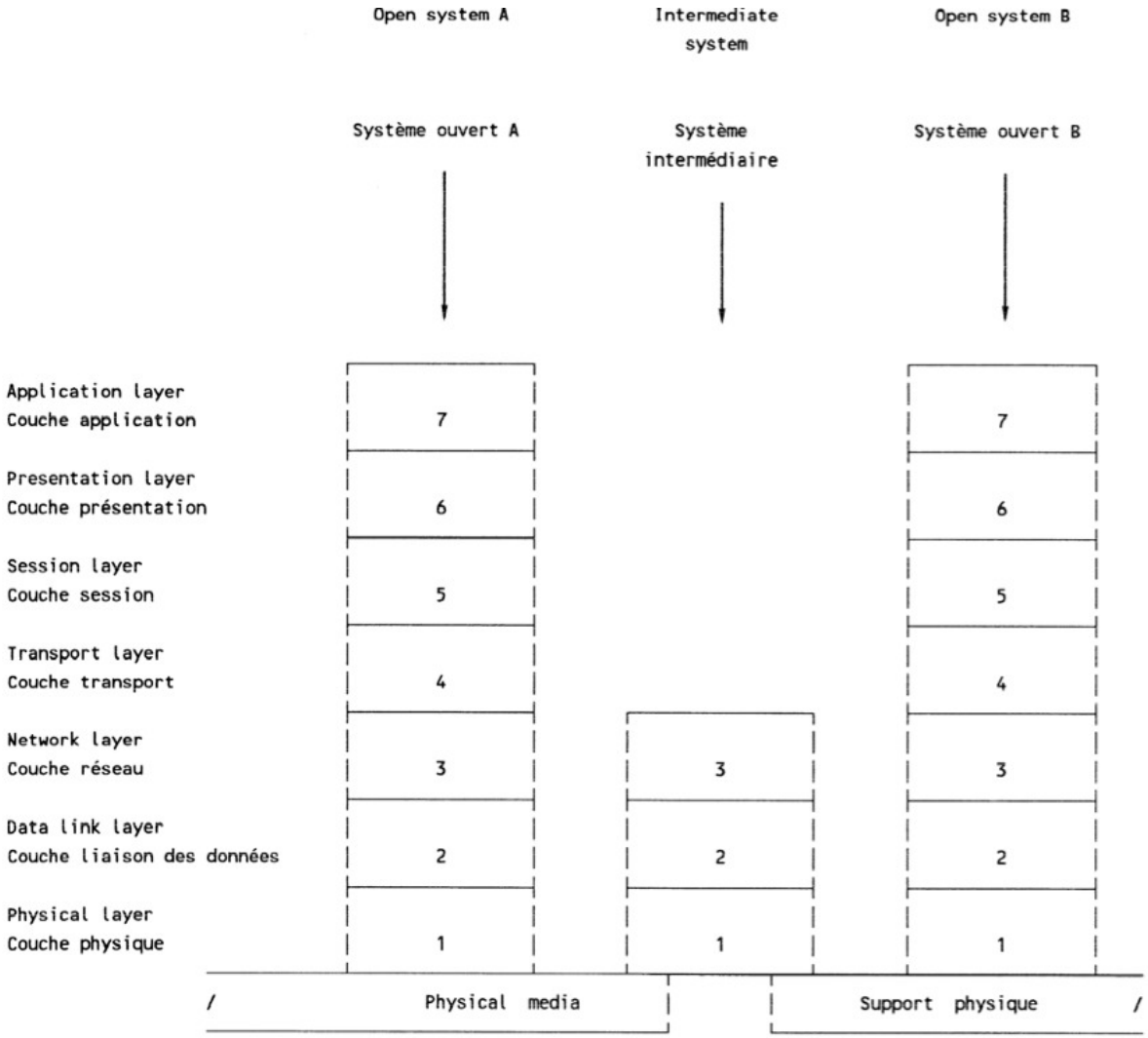
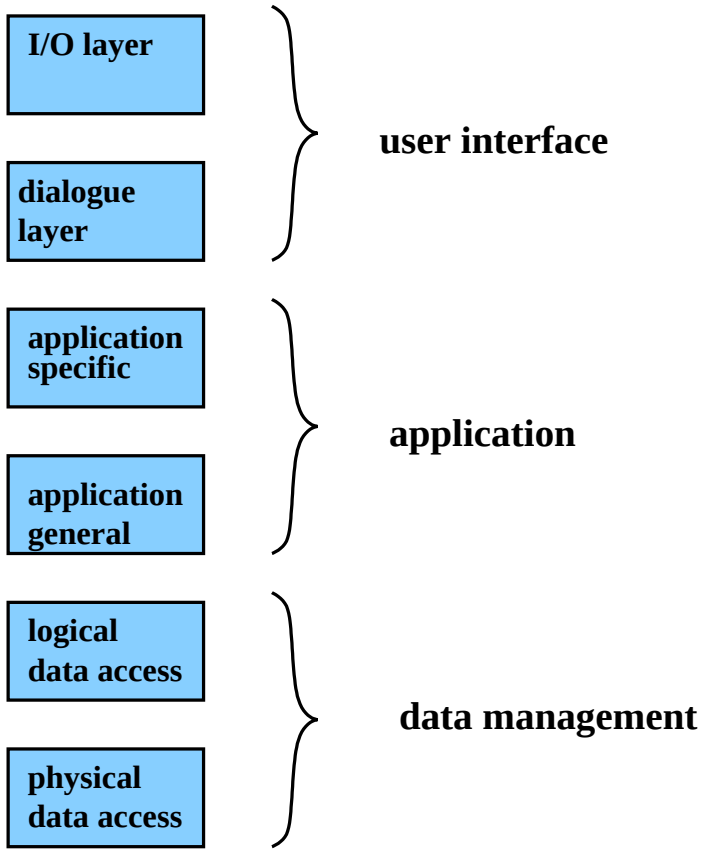


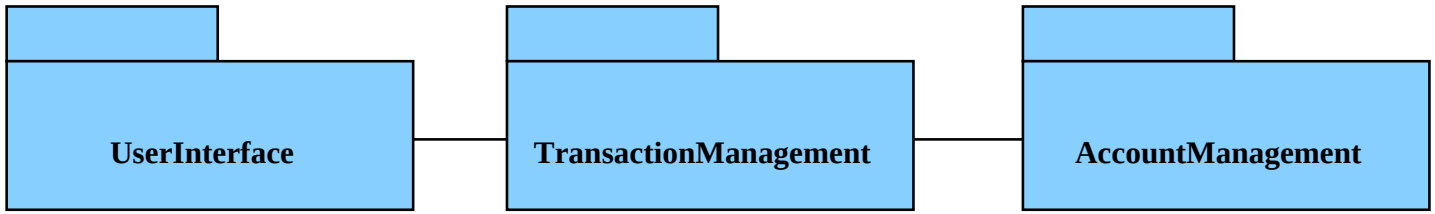
Figure 1

The seven-layer reference model for open systems interconnection.

Typical Setup



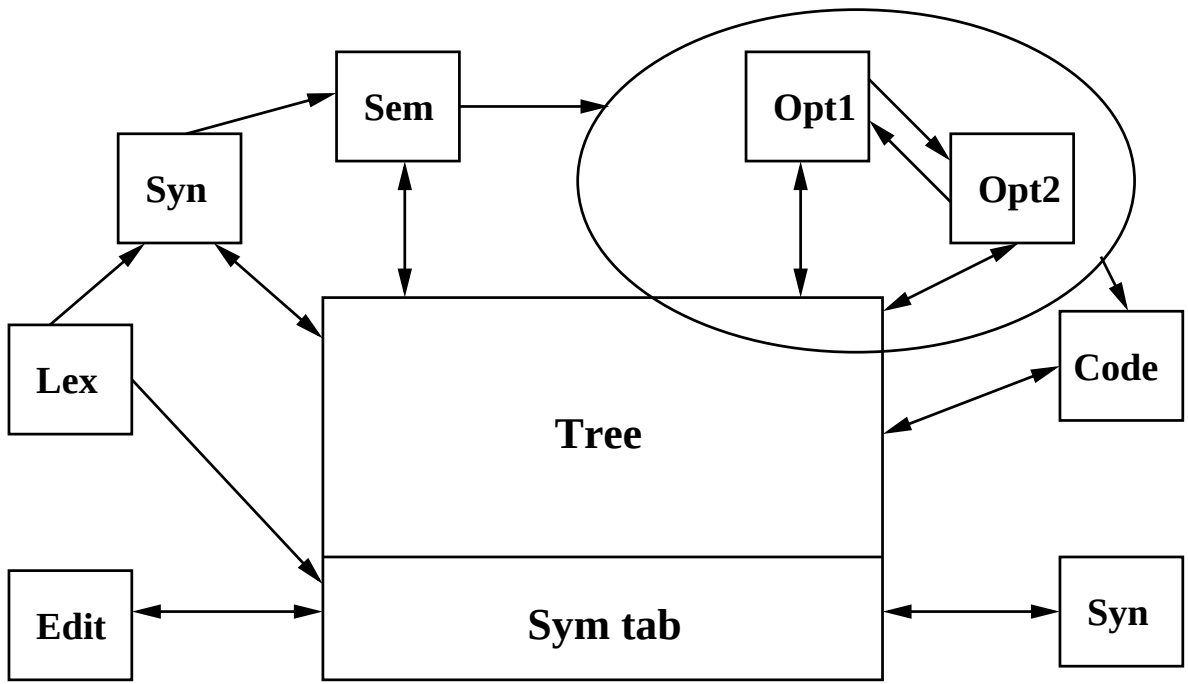
Example: Three-Tier Architecture



- Three kinds of subsystems
  - user interface
  - control — transaction management
  - database — account management
- Enables consistent look-and-feel
- Useful with single data repository
- Web architecture
  - each tier runs on different location
  - browser, web server, application server

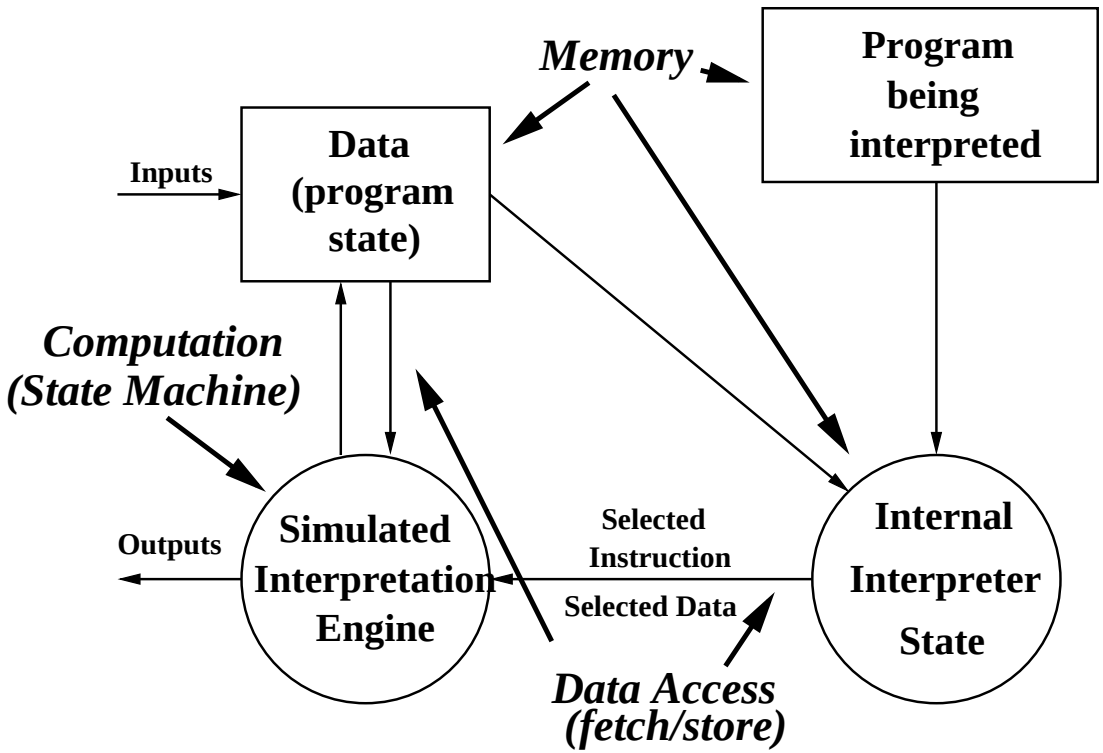
Architecture: Repository

- Central data structure (current state, blackboard)
- Independent components acting on it
- Example: architecture of modern compilers, theorem provers



Architecture: Interpreter

- Virtual machine in software
- Bytecode program + interpretation engine
- Examples: programming language, malware packers



Further Architectural Styles

- Distributed processes
  - topological features
  - interprocess protocols
  - client-server organization
- Main program/subroutine: mirroring the programming language
- Domain specific SW architectures
  - tailored to family of applications
  - structured, e.g., according to hardware requirements
  - Examples: avionics, vehicle management, ...
- State transition systems
- Combinations of architectural styles
  - hierarchically
  - mixture of connectors