# Software Architecture in Perspective

SENG 480/580

(H. Muller)

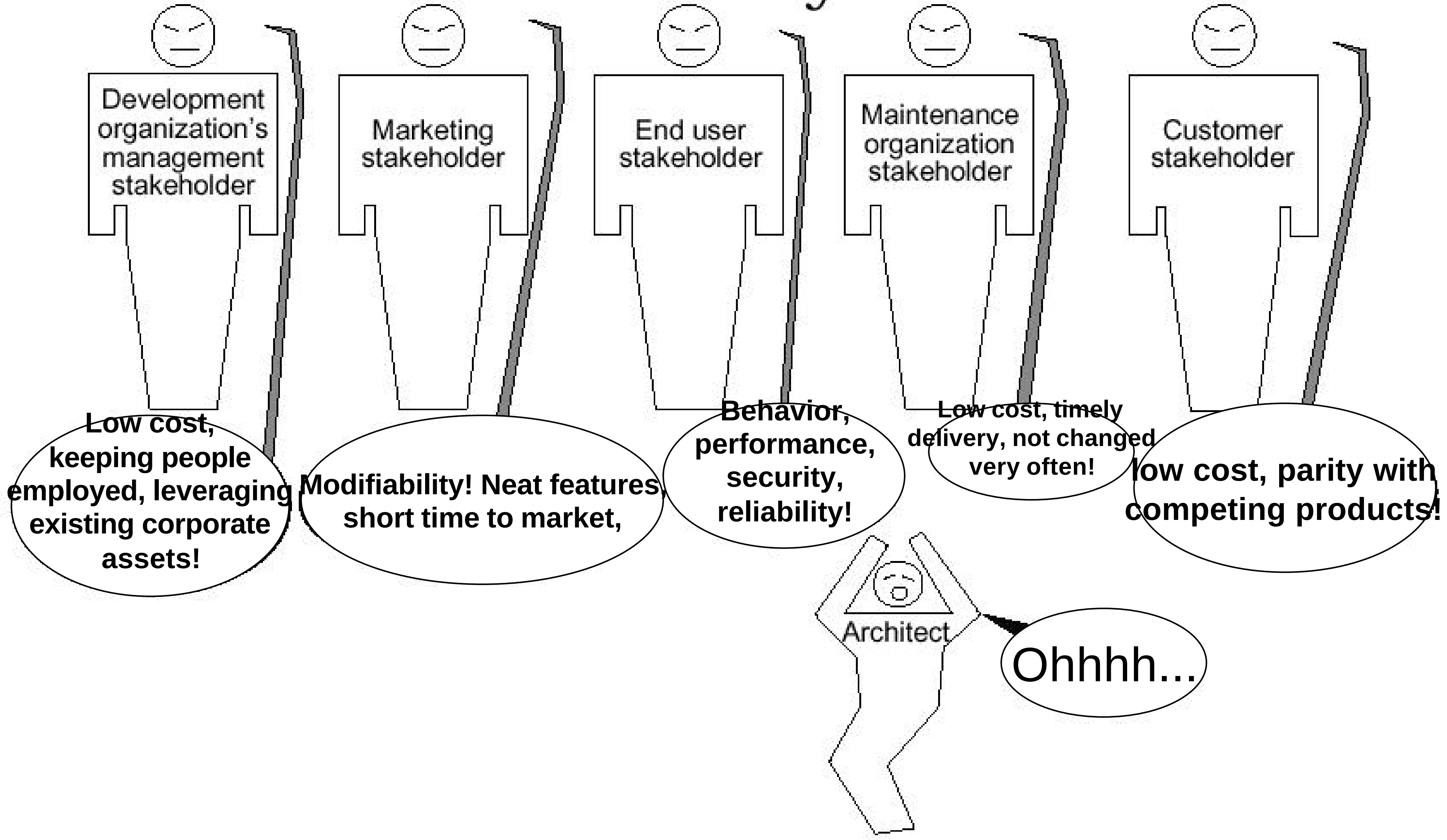
Today: **Margaret-Anne Storey** (mstorey@uvic.ca)

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

## Recap from last lecture:

- Why is software architecture important?
- Business importance of software architecture
- Factors influencing software architecture
- Different stakeholders

# Stakeholders of a System


Development organization's management stakeholder
Marketing stakeholder
End user stakeholder
Maintenance organization stakeholder
Customer stakeholder
Low cost, keeping people employed, leveraging existing corporate assets!
Modifiability! Neat features, short time to market,
Behavior, performance, security, reliability!
Low cost, timely delivery, not changed very often!
low cost, parity with competing products!
Architect
Ohhhh...

## Other Stakeholders

Development organization

Marketers

Maintenance/Reuse organization

## Development Organization Concerns

**Immediate business issues**

- **amortizing the infrastructure**
- **keeping cost of installation low**
- **utilizing personnel**

**Long-term business issues**

- **investing in an infrastructure for strategic goals**
- **investing in personnel**

**Organizational structure issues**

- **furthering vested interests, e.g.,**
  - **maintaining an existing database organization**
  - **supporting specialized expertise**
- **maintaining the standard method of doing business**

## Technical Environment

**Current trends: today's information system will likely employ a**

- **database management system**
- **Web browser for delivery and distribution across platforms**

**This was not true 10 years ago.**

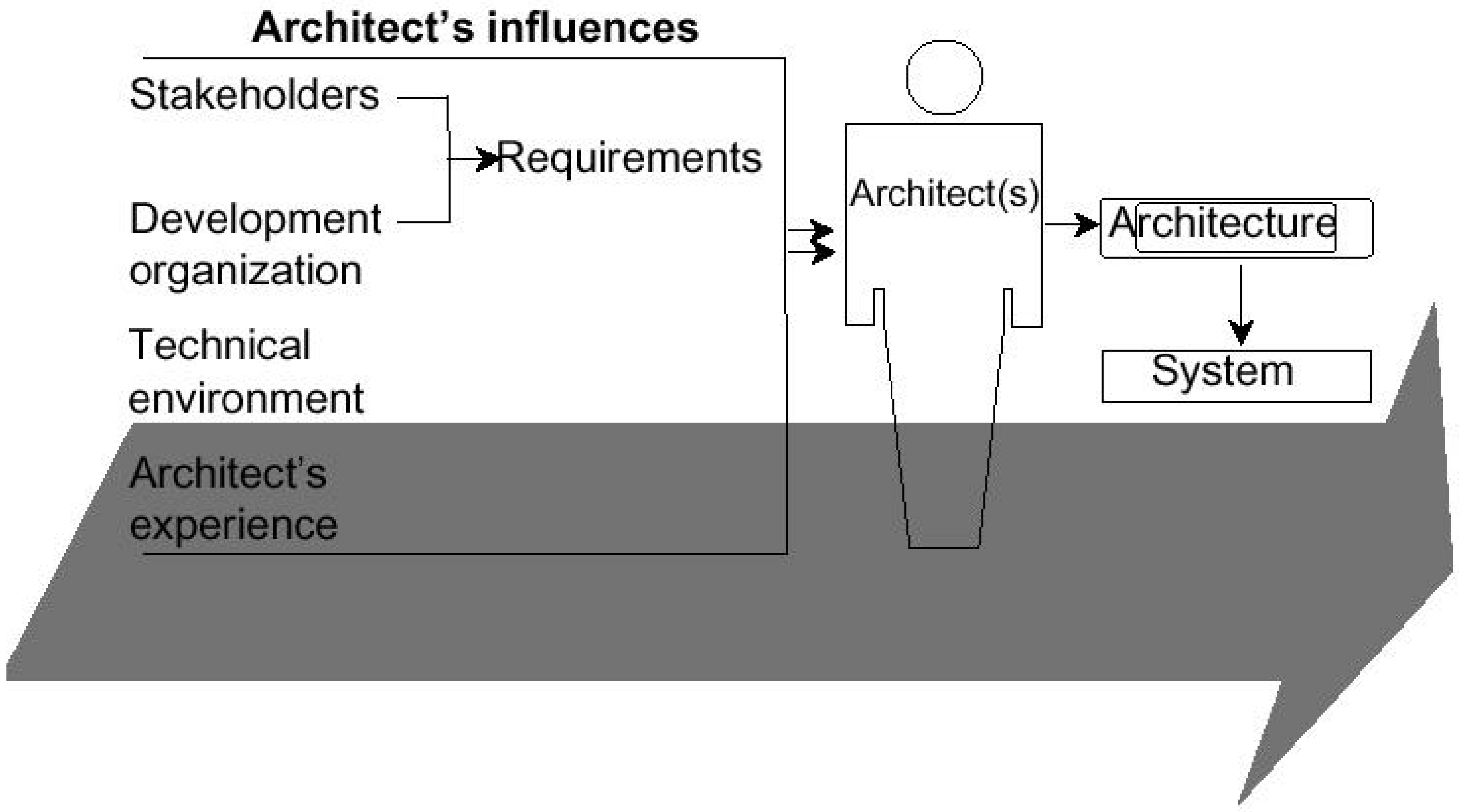
**Available technology: decisions on using a centralized or decentralized system depend on processor cost and communication speed; both are changing quantities.**

## Architect's Background

**Architects develop their mindset from their past experiences.**

- **Prior good experiences will lead to replication of prior designs.**
- **Prior bad experiences will be avoided in the new design.**

# Summary: Influences on the Architect



## Factors Influenced by Architectures

Structure of the development organization

Enterprise goals of the development organization

Customer requirements

Architect's experience

Technical environment

The architecture itself

## Architecture Influences the Developing Organization's Structure

**Short term: Work units are organized around architectural units for a particular system under construction.**

**Long term: When company constructs collection of similar systems, organizational units reflect common components (e.g., operating system unit or database unit).**

## Architecture Influences the Developing Organization's Enterprise Goals

**Development of a system may establish a foothold in the market niche.**

**Being known for developing particular kinds of systems becomes a marketing device.**

**Architecture becomes a leveraging point for additional market opportunities and networking.**

## Architecture Influences Customer Requirements

**Knowledge of similar fielded systems leads customers to ask for particular features.**

**Customers will alter their requirements on the basis of the availability of existing systems.**

## Architecture Influences the Architect's Experience and Technical Environment

**Creation of a system affects the architect's background.**

**Occasionally, a system or an architecture will affect the technical environment.**

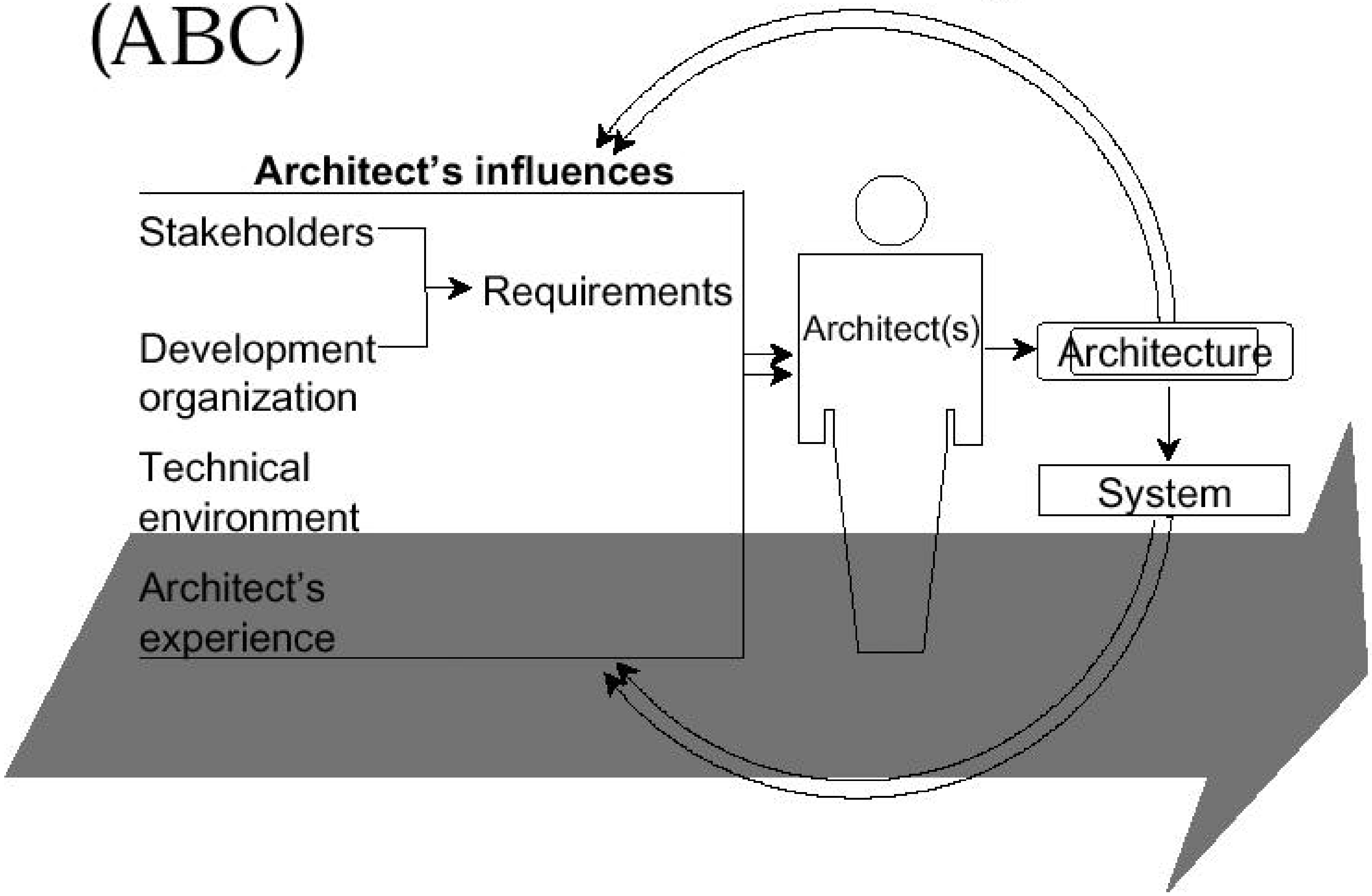
- **the WWW for information systems**
- **the three-tier architecture for database systems**

## A Cycle of Influences

**Architectures and organizations influence each other.**

- **Influences to and from architectures form a cycle.**
- **An organization can manage this cycle to its advantage.**

## Architecture Business Cycle (ABC)



## What Makes a Good Architect? -1

**People skills: must be able to**

- **negotiate competing interests of multiple stakeholders**
- **promote inter-team collaboration**

**Technical skills: must**

- **understand the relationships between qualities and structures**
- **possess a current understanding of technology**
- **understand that most requirements for an architecture are not written down in any requirements document**

## What Makes a Good Architect? -2

**Communication skills: must be able to**

- **clearly convey the architecture to teams (both verbally and in writing)**
- **listen to and understand multiple viewpoints**

## What Makes a Good Architecture?

**Fitness for purpose**

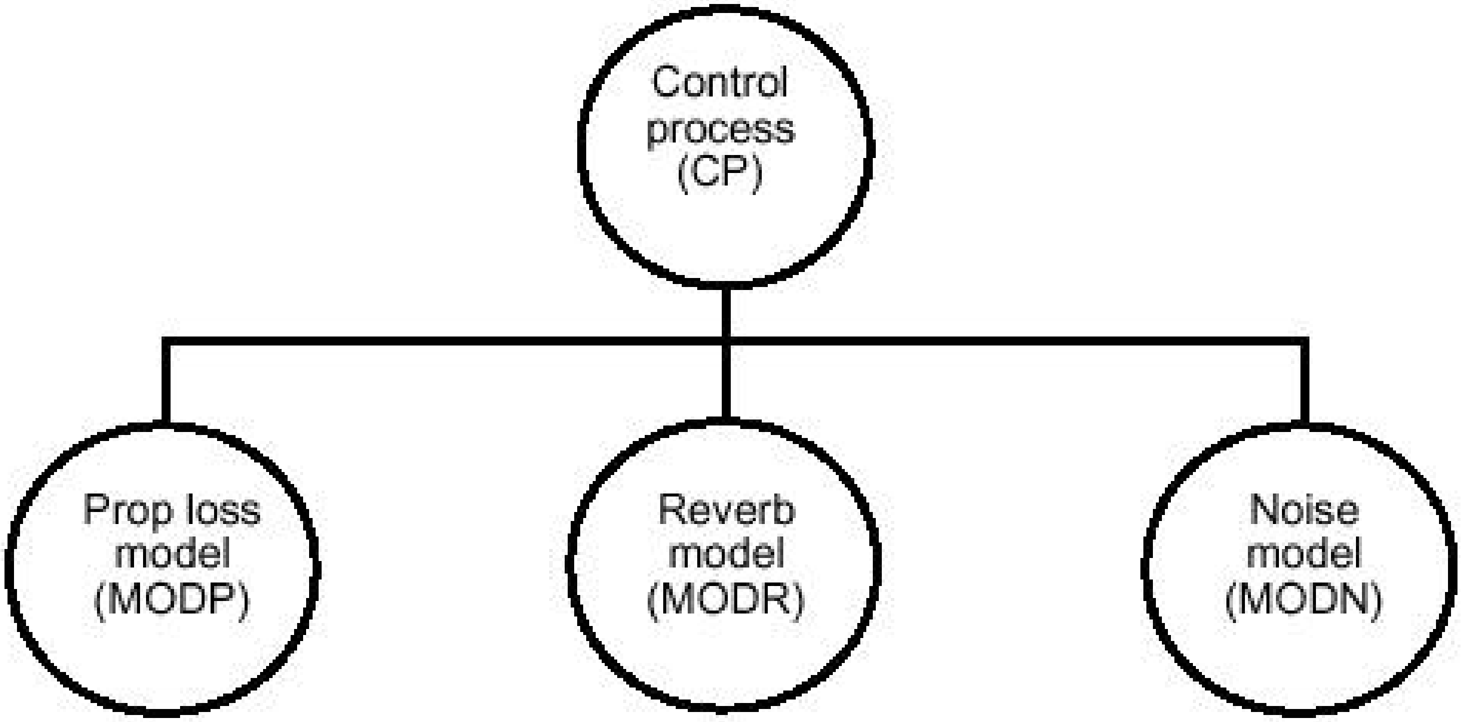
**Achievable within a reasonable budget**

**Achievable within a reasonable time**

**So,**

**… what is software architecture?**

## A Software Architecture (?)



## What’s Wrong with this Diagram?

- Many things are left unspecified:
  - What kind of components?
  - What kind of connectors?
  - What do the boxes and arrows mean?
  - What is the significance of the layout?
  - Why is control process on a higher level?

➢ Box and arrow drawings alone are not architectures; rather, they are a starting point.

## What’s Wrong with the Diagram?

*Which* structure? Software is composed of *many* structures.

- modules
- tasks
- functions
- hardware
- classes

Thus, when seeing boxes and lines, we must ask

- What do the boxes represent?
- What do the arrows mean?

## What is Software Architecture?

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

## Implications of this Definition

Architecture is an abstraction of systems.

- Architecture defines components and how they interact.
- Architecture suppresses purely local information; private component details are not architectural.

Systems have many structures (views).

- No single view can be the architecture.
- The set of candidate views is not fixed or prescribed: whatever is useful for analysis or communication.

## More Implications

Every system *has* an architecture.

- Every system is composed of components and relationships among them.
- In the simplest case, a system is composed of a single component, related only to itself.

Having an architecture is different from having an architecture that is known. Issues:

- precise specification of the architecture
- architecture recovery and conformance
- rationale for the architecture

## Why Is Architecture Important?

Architecture is important for (at least) three reasons:

- It provides a vehicle for communication among stakeholders.
- It is the manifestation of the earliest design decisions about a system.
- It is a transferable, reusable abstraction of a system.

## Communication Vehicle

Architecture provides a common frame of reference in which competing interests may be exposed and negotiated.

- negotiating requirements with users
- keeping the customer informed of progress and cost
- implementing management decisions and allocations

## Result of Early Design Decisions -1

An architecture constrains an implementation.

- implementations must conform to architecture
- (global) resource allocation decisions constrain implementations of individual components
- system tradeoffs are manifested in the architecture

## Result of Early Design Decisions -2

The architecture dictates organizational structure for development/maintenance efforts, e.g.

- division into teams
- units for budgeting, planning
- basis of work breakdown structure
- organization for documentation
- organization for CM libraries
- basis of integration, test plans, testing
- basis of maintenance

➢ Once committed to, an architecture is extremely hard to change!

## Result of Early Design Decisions -3.

Architecture permits/precludes the achievement of a system’s desired quality attributes (modifiability, performance, security, etc.).

The architecture influences qualities, but does not guarantee them.

An architecture helps with evolutionary prototyping.

- Architecture serves as a skeletal framework into which components can be plugged.
- By segregating functionality into appropriate components, experimentation is easier.

## Reusable Model

An architecture forms a reusable model. It:

- provides a vocabulary of design
- enables template-based component development
- enables product lines
- enables systems to be built from externally developed components
- separates functionality from packaging and interconnection mechanisms

## Architectural Structures - 1

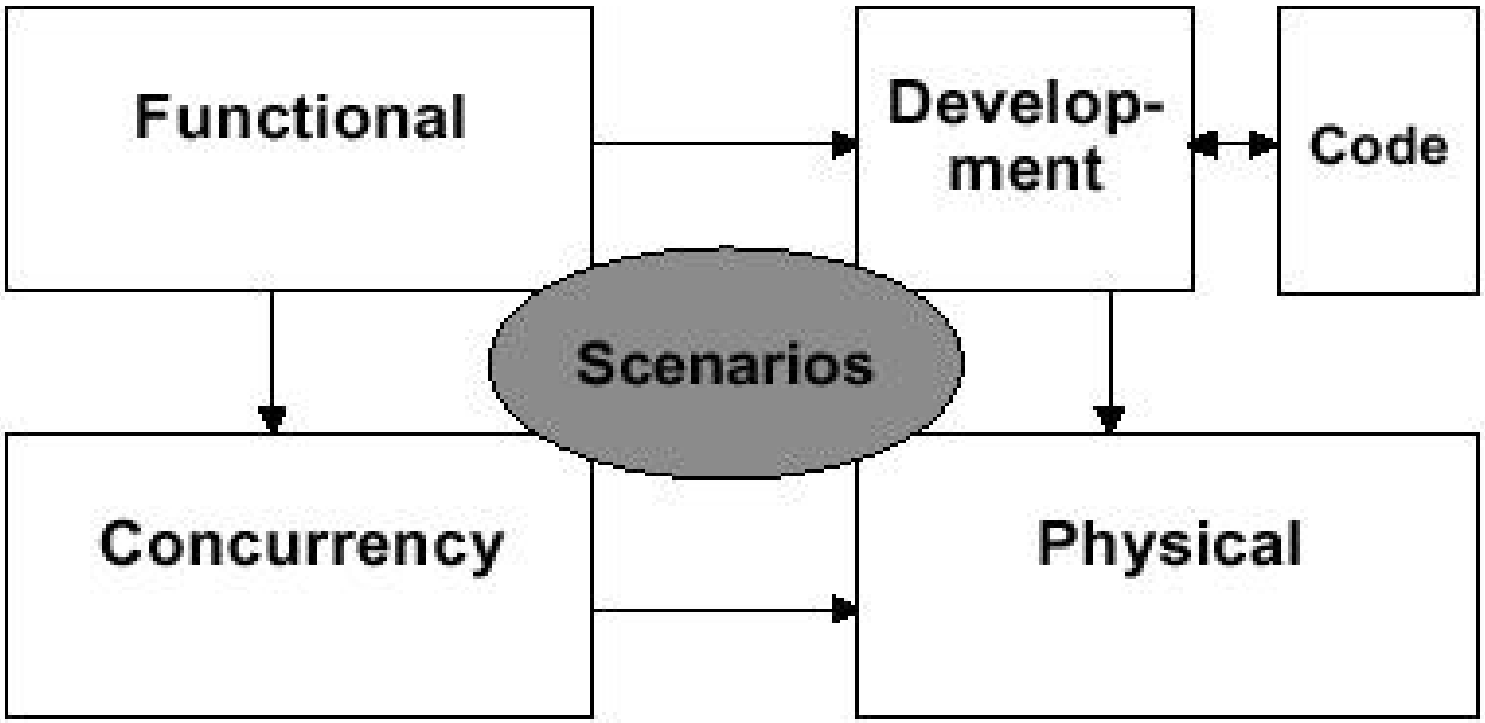
In a house, there are plans for rooms, electrical wiring, plumbing, ventilation, . . .

Each of these constitutes a “view” of the house. These views are

- used by different people
- used to achieve different qualities in the house
- used as a description and prescription

So it is with software architecture.

## Architectural Structures - 2.



⟹ **Similar to Kruchten’s 4+1 View Model**

## Functional View

Components:

- functions, key system abstractions, domain elements

Connectors:

- dependencies, data flow

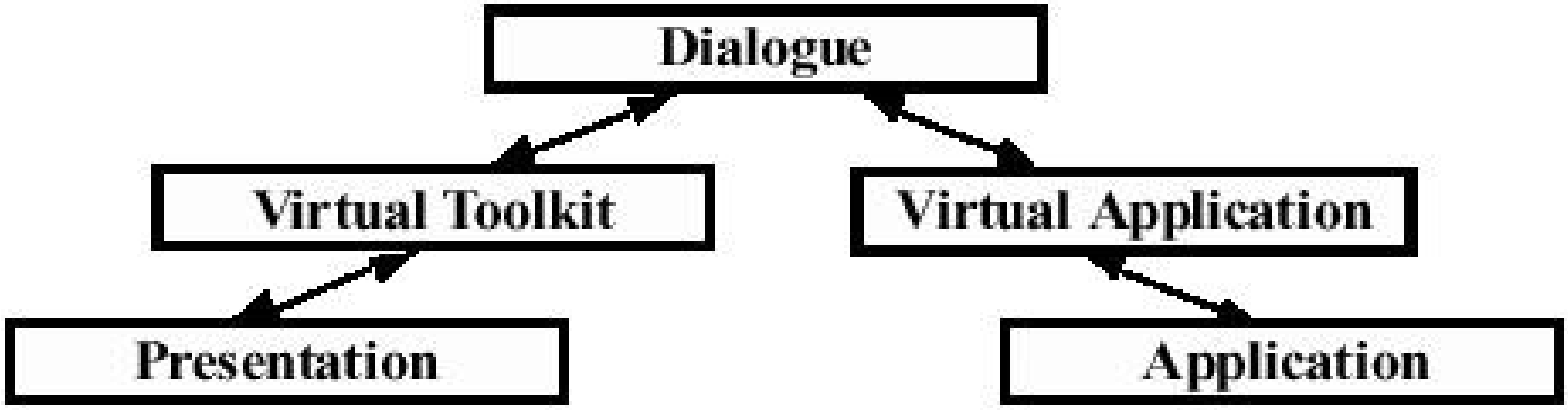
Users:

- domain engineers, product-line designers, end users

Reasoning:

- functionality, modifiability, product lines/reusability, tool support, work allocation

## Functional View Example



## Code View

Components:

- classes, objects, procedures, functions
- subsystems, layers, modules

Connectors:

- calls, invokes
- is-a-sub-module-of

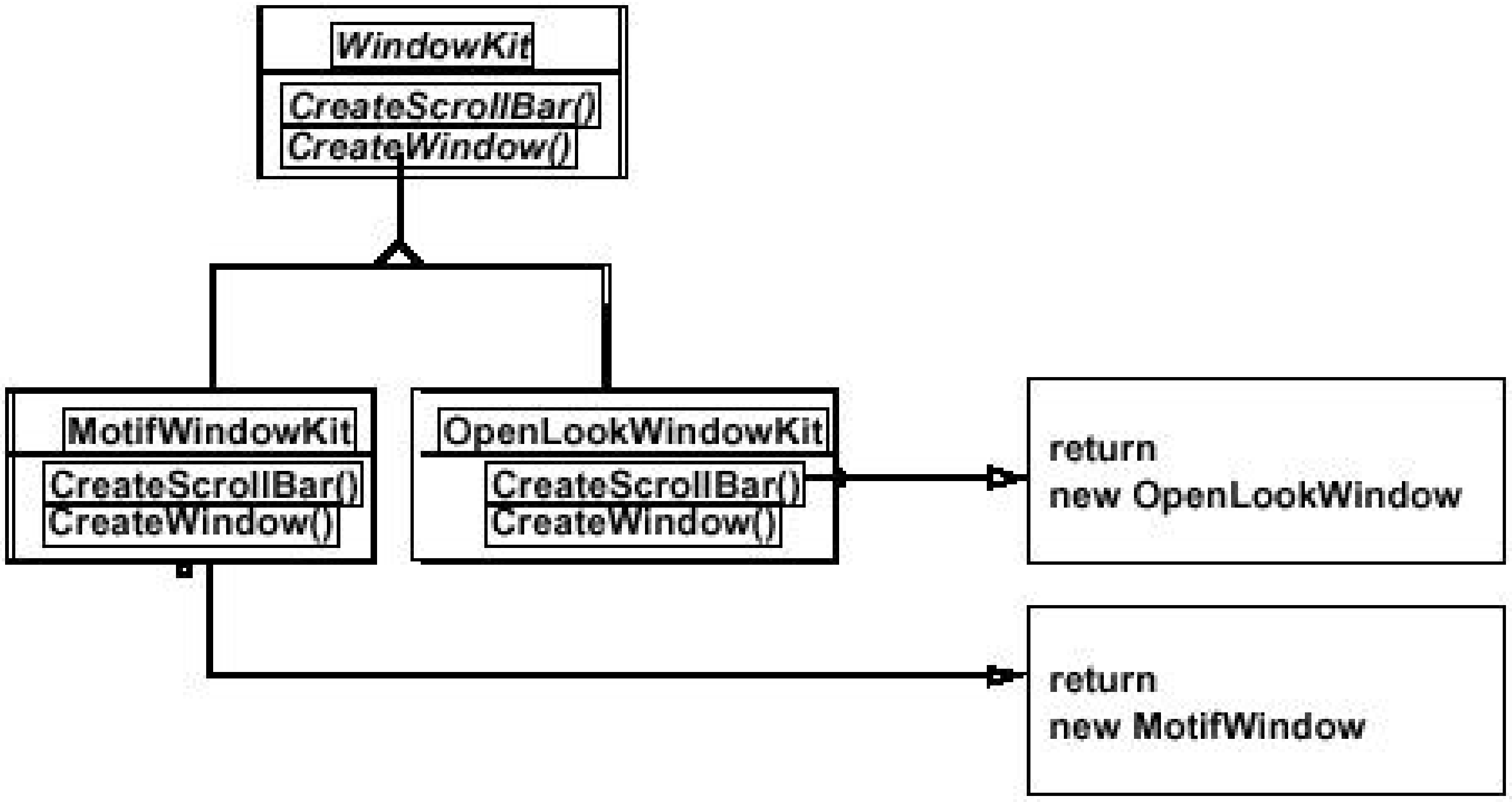
Users:

- programmers, designers, reusers

Reasoning:

- modifiability/maintainability, portability, subsetability

## Code View Example



## Development View

Components:

- files, directories

Connectors:

- contains

Users:

- managers, programmers, configuration managers

Reasoning:

- modifiability/maintainability
- testing
- configuration management/version control

## Concurrency View

Components:

- processes, threads

Connectors:

- synchronization, data flow, events

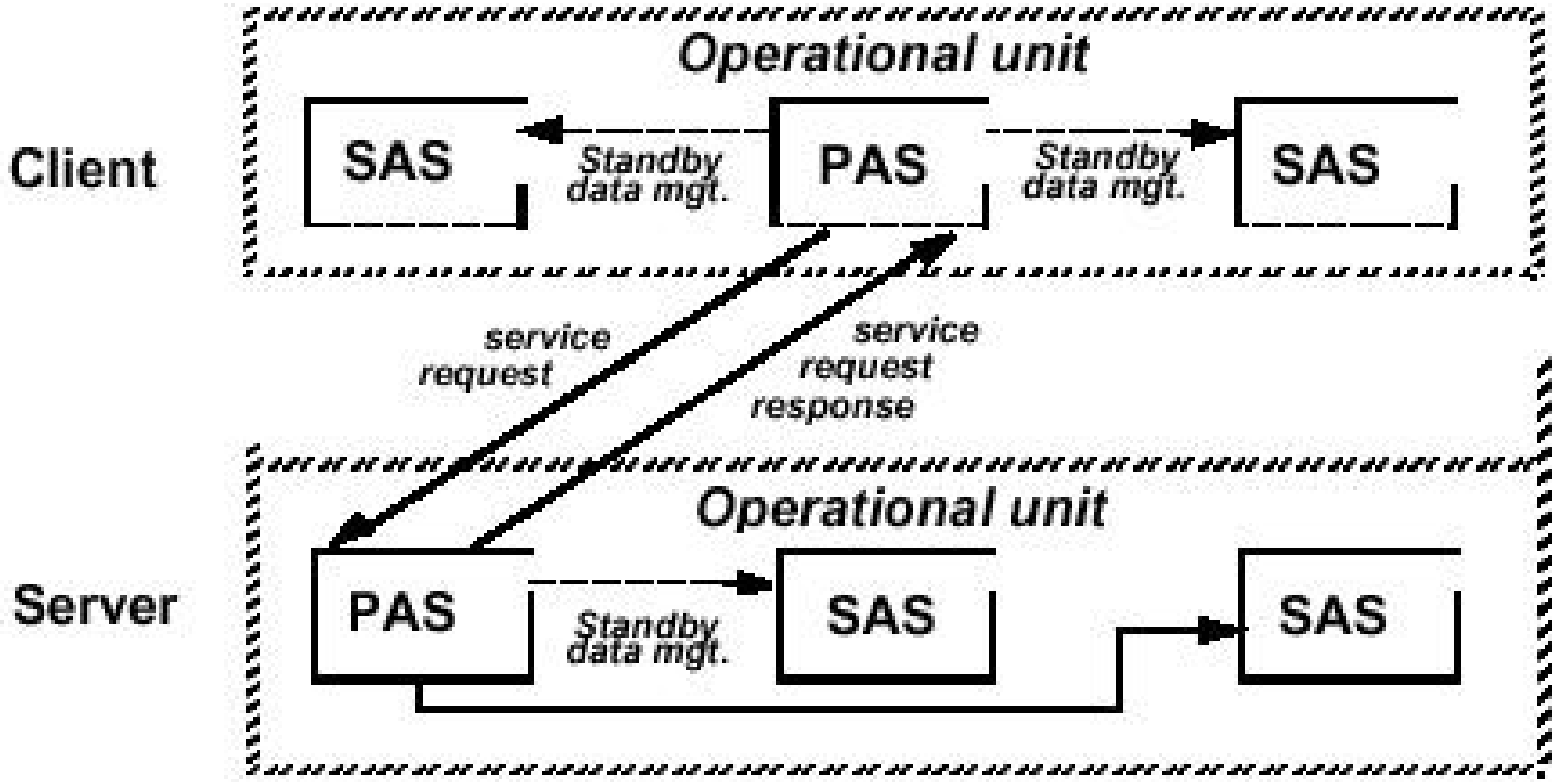
Users:

- performance engineers, integrators, testers

Reasoning:

- performance, availability

# Concurrency View Example



## Physical View

Components:

- CPUs, sensors, storage

Connectors:

- networks, communication devices

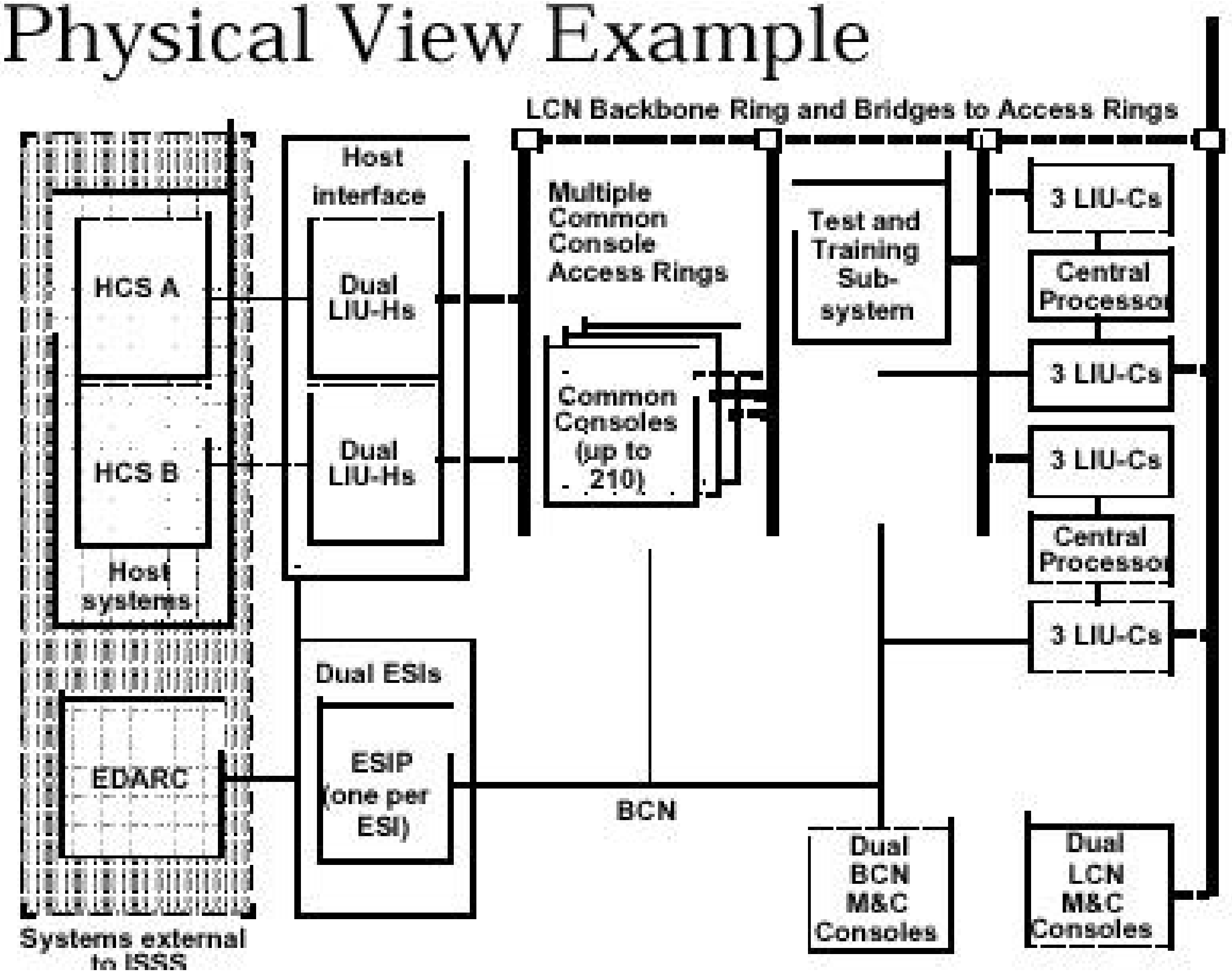
Users:

- hardware engineers, system engineers

Reasoning:

- system delivery and installation, performance, availability, scalability, security

# Physical View Example



## Scenarios

What are scenarios?

- use cases: sequences of responsibilities
- change cases: changes to the system

Why use scenarios?

- to understand and validate the design
- to communicate the design
- to tie the views together
- to animate the design
- to understand the limits of the design

## What Are Views Used For?

Each view provides an *engineering handle* on certain quality attributes.

Views are an engineering tool to help achieve desired system qualities.

In some systems, distinct views collapse into one. (E.g., the concurrency and physical views may be the same for small systems.)

## What Are Views Used For?

Documentation vehicle for

- current development and future development
- managers and customers

Thus, views must be *annotated* to support analysis.  Scenarios aid in annotating views with *design rational*e**.**

# Hierarchical Views

Every view is potentially hierarchical, e.g.:

- *functiona*l: functions contain sub-functions
- *developmen*t: directories contain files
- *cod*e: modules contain sub-modules; systems contain sub-systems
- *concurrenc*y: processes contain threads
- *physica*l: clusters contain computers contain processors

Because views are complex and hierarchical, they need to be navigable.

## View Navigability - 1

A skilled software engineer with some domain knowledge should be able to read the documentation and navigate through it.

There should be an obvious starting point, portraying the system as a collection of interconnected subsystems.

Subsystems should be named, with their responsibilities, functionality, and interconnections identified.

## View Navigability - 2.

Pointers should direct the reader to more detailed documentation of sub-structures.

Tool support can and should aid in navigation.

At every stage, the nature of the connections among the parts should be clearly identified.

## Lecture Summary

Software architecture is about components, connectors, annotations and views. It exists to support analysis and hence decision making.

Software architecture the key to a cycle of influences called the Architecture Business Cycle.

Architectural views are related to each other in complicated ways.

*Lesso*n: Choose the views that are useful to the system being built and to the achievement of qualities that are important to you.