

Strategic
Computing and Communications
Technology

BA 290D, EECS 201, IS 224
Spring 99

Technology introduction

by
David G. Messerschmitt

Goal

- Define terminology that can be used throughout the course
- Basic background that will be useful

2

Definitions

- What is an application?
- What is a technology?
- What is information technology?
- What is the relationship between application and technology?

3

Application

- Application = something that puts technology to use to the benefit of individuals or organizations
 - (Term sometimes used in broader and relative sense -- e.g. Pentium/PC)
- Technology = something that puts scientific principles to use

4

Trend

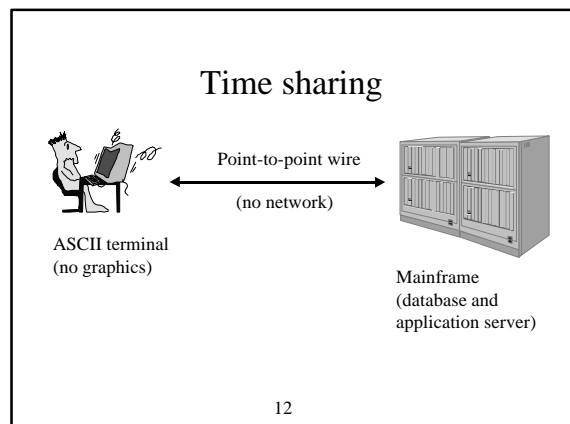
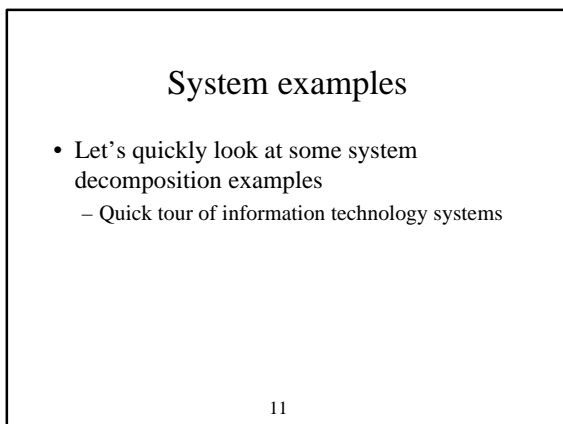
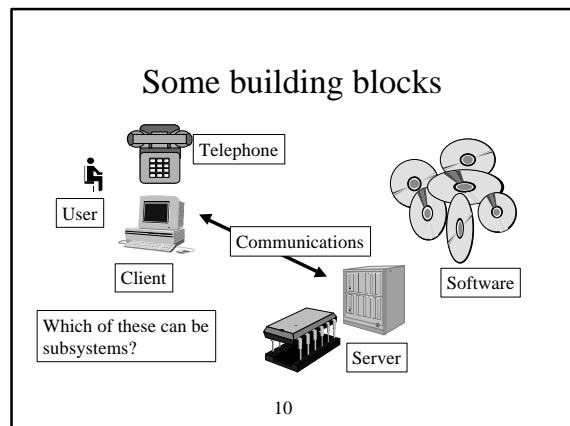
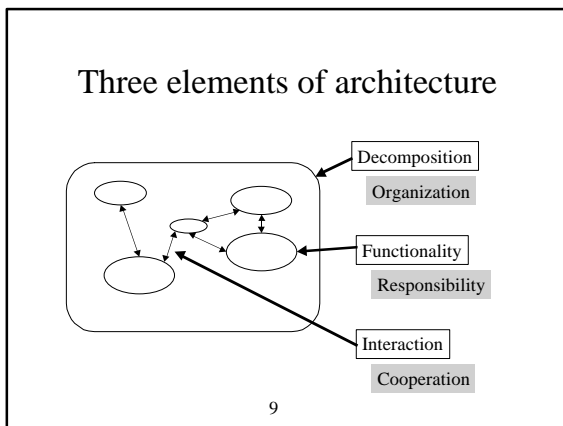
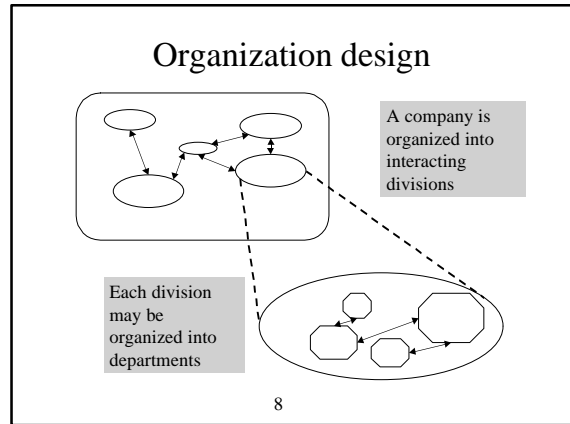
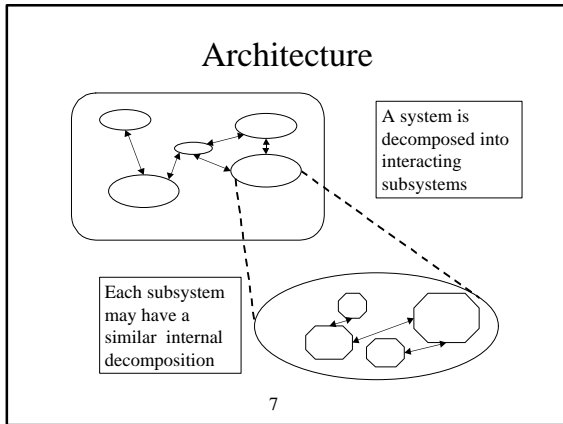
- Relative importance is shifting from technology to applications (ironically due to technology advances)
 - Appreciation of end-user needs and market opportunities (marketing)
 - Appreciation of many non-technical or partly technical factors that contribute to the success or failure of technologies or high-tech products (this course)

5

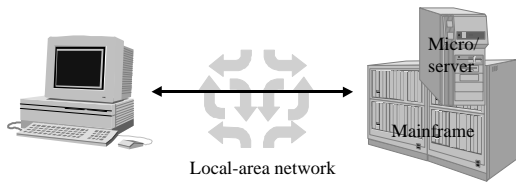
Outline

- Architecture
 - Decomposition
 - Modularity
 - Interfaces
- Hardware
- Software

6

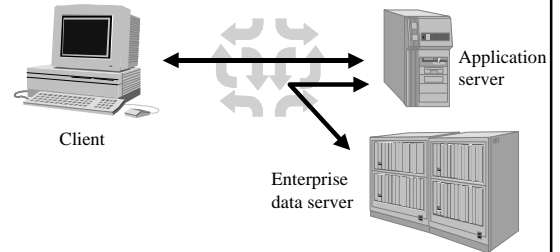


Two-tier client/server



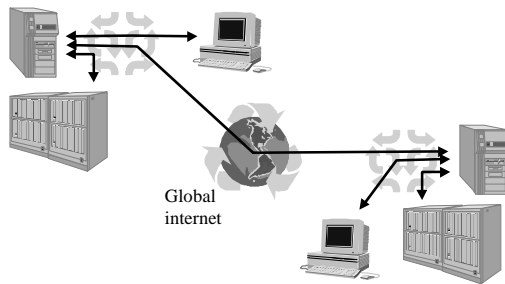
13

Three-tier client/server



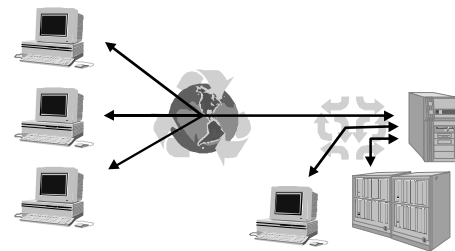
14

Inter-organizational computing



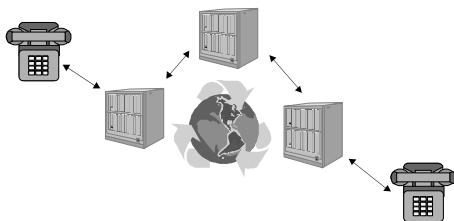
15

Consumer access



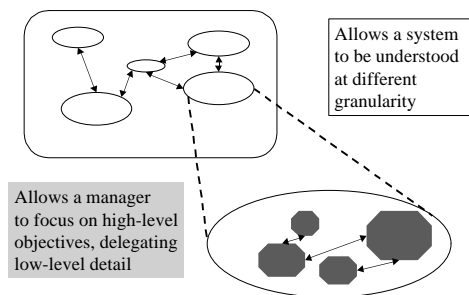
16

Telephone system



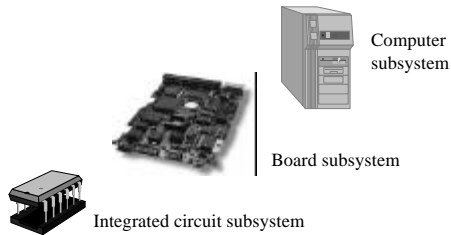
17

Hierarchy



18

Hierarchy in hardware architecture



19

Emergence

- Subsystems are more specialized and simpler functionality
- Higher-level system functionality arises from the interaction of subsystems
- Emergence includes capabilities that arise purely from that interaction
 - e.g. airplane flies, but subsystems can't

20

System integration

Architecture → subsystem implementation → system integration

- Bring together subsystems and make them cooperate properly to achieve desired system functionality
 - Always requires testing
 - May require modifications to architecture and/or subsystem implementation

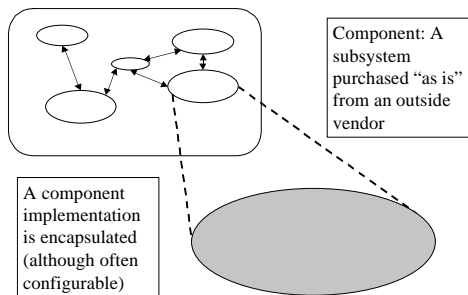
21

Why system decomposition?

- Divide and conquer approach to containing complexity
- Reuse
- Consonant with industry structure (unless system is to be supplied by one company)
- Others?

22

Components



23

Examples of components

- Computer
- Disk drive
- Network
- Network router
- Operating system
- Integrated circuit
- Database management system

Why is a component implementation encapsulated?

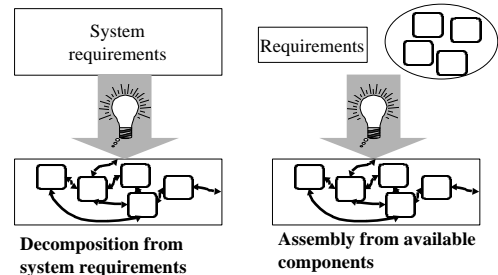
24

Interoperability

- Components are interoperable when they interact properly to achieve some desired functionality
- Increasingly component interoperability cannot be dependent on integration, or is dependent on end-user integration
 - PC and peripherals
 - Enterprise, inter-enterprise, consumer applications
 - Role for standardization

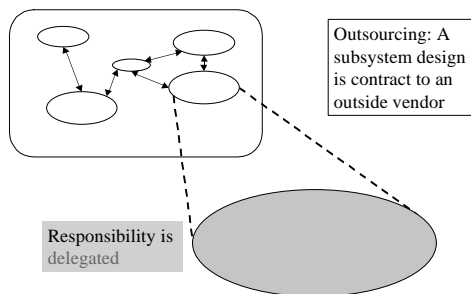
25

Two ways to design a system



26

Outsourcing



27

Interfaces

- Focus of component or subsystem interoperability
- Two purposes:
 - Tells other subsystems how to interact with that component or subsystem
 - Tells component or subsystem implementer what has been promised

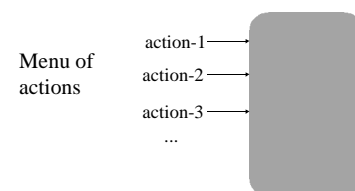
28

Hardware interface

- Physical connection
- Electrical properties
- Data formats (structure and interpretation)

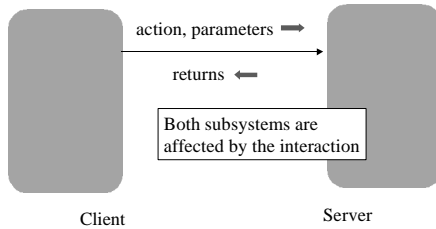
29

Software interface



30

Interaction through interfaces



31

Protocols

- A defined sequence of actions between/among two or more subsystems required to achieve some higher-level functionality
- Interface specification focuses on actions (including formats of parameters and returns) and protocols

32

Encapsulation

- Subsystem implementation details (anything not explicit at interface) should be encapsulated
 - So other subsystems cannot become inadvertently dependent on implementation
 - In the case of components, for proprietary or security reasons

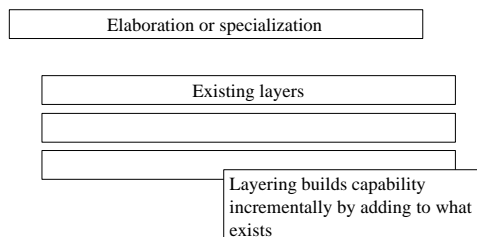
33

“Good” architecture: modularity

- Divide and conquer: decomposition of the system into modules
- Separation of concerns: functional grouping within modules results in great dependency internally, little dependency externally
- Encapsulation: internal implementation details hidden so that other modules cannot become dependent on that implementation
- Reusability: meet generalized needs, configurable

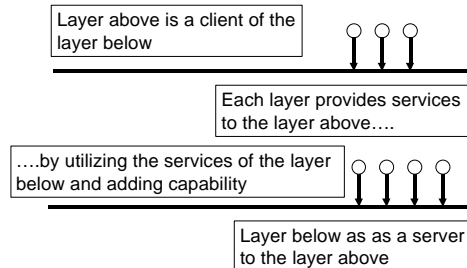
34

Layering



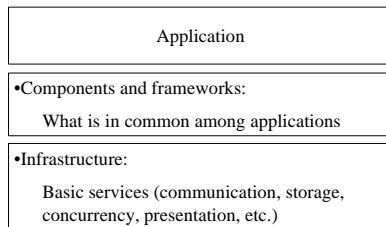
35

Interaction of layers



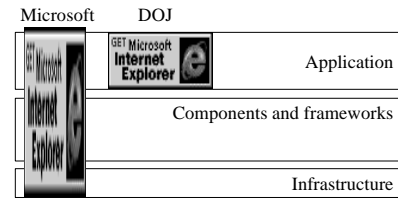
36

Three types of software



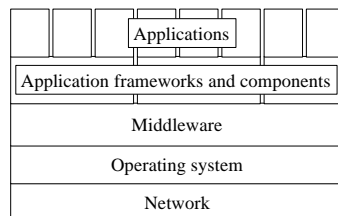
37

Part of Microsoft vs. DOJ dispute



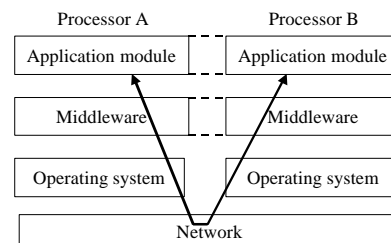
38

Major layers



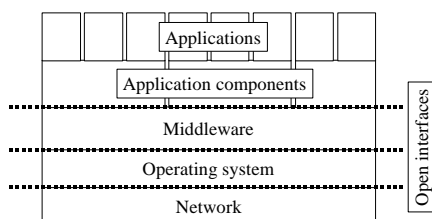
39

Communication



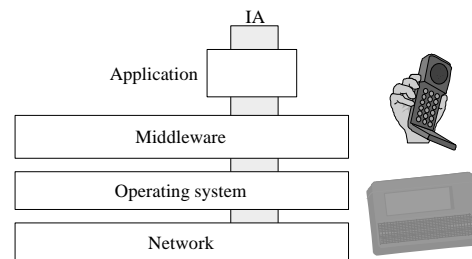
40

Open layer interfaces



41

Information appliances



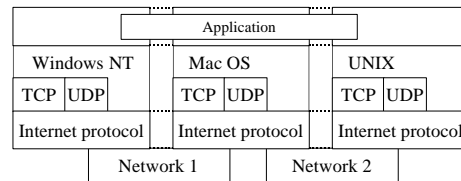
42

Question

- What advantages and disadvantages do you see for the information appliance?

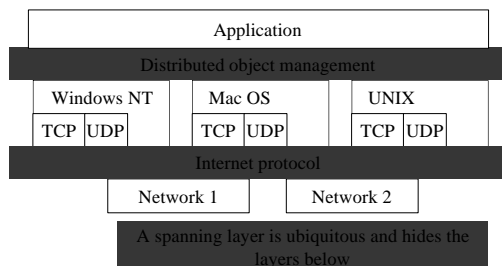
43

Horizontal structure in layers



44

Spanning layer



A spanning layer is ubiquitous and hides the layers below

45