Chapter 11

Distributed Systems Architectures

Distributed Systems Architectures

Published by Justin Suarez

Modified over 9 years ago

Presentation on theme: "Distributed Systems Architectures"— Presentation transcript:

1 **Distributed Systems Architectures**
Chapter 11
Distributed Systems Architectures

2 **Distributed Systems Architectures**

3 Objectives

To explain the advantages and disadvantages of distributed systems architectures

To describe different approaches to the development of client-server systems

To explain the differences between client-server and distributed object architectures

To describe object request brokers and the principles underlying the CORBA standards

4 **Topics covered Multiprocessor architectures**

Client-server architectures

Distributed object architectures

CORBA

5 Distributed systems

Virtually all large computer-based systems are now distributed systems

Information processing is distributed over several computers rather than confined to a single machine

Distributed software engineering is now very important

6 System types

Personal systems that are not distributed and that are designed to run on a personal computer or workstation.

Embedded systems that run on a single processor or on an integrated group of processors.

Distributed systems where the system software runs on a loosely integrated group of cooperating processors linked by a network.

7 **Distributed system characteristics**

Resource sharing

Openness

Concurrency

Scalability

Fault tolerance

Transparency

8 **Distributed system disadvantages**

Complexity

Security

Manageability

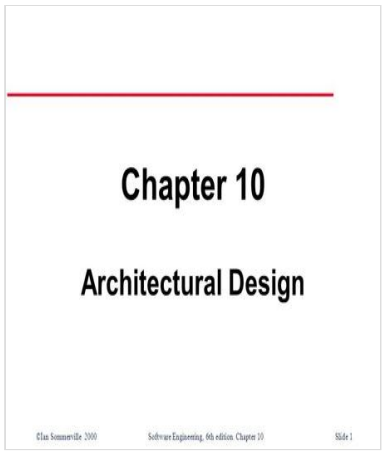
Unpredictability

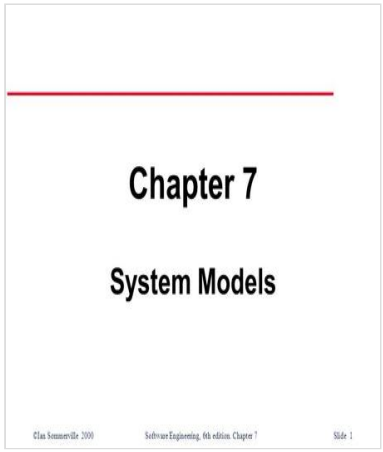
9 **Issues in distributed system design**

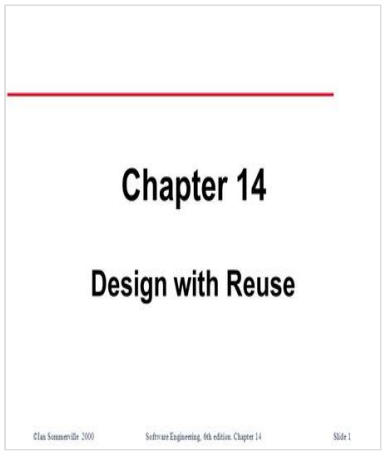
10 **Distributed systems archiectures**

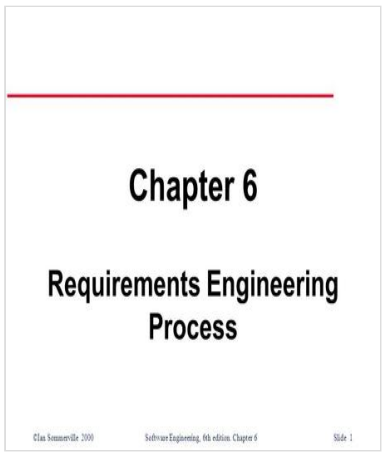
Client-server architectures

Distributed services which are called on by clients. Servers that provide









11 Middleware

Software that manages and supports the different components of a distributed system. In essence, it sits in the middle of the system

Middleware is usually off-the-shelf rather than specially written software

Examples

Transaction processing monitors

Data convertors

Communication controllers

12 **Multiprocessor architectures**

Simplest distributed system model

System composed of multiple processes which may (but need not) execute on different processors

Architectural model of many large real-time systems

Distribution of process to processor may be pre-ordered or may be under the control of a despatcher

13 **A multiprocessor traffic control system**

14 **Client-server architectures**

The application is modelled as a set of services that are provided by servers and a set of clients that use these services

Clients know of servers but servers need not know of clients

Clients and servers are logical processes

The mapping of processors to processes is not necessarily 1 : 1

15 **A client-server system**

16 **Computers in a C/S network**

17 **Layered application architecture**

Presentation layer

Concerned with presenting the results of a computation to system users and with collecting user inputs

Application processing layer

Concerned with providing application specific functionality e.g., in a banking system, banking functions such as open account, close account, etc.

Data management layer

Concerned with managing the system databases

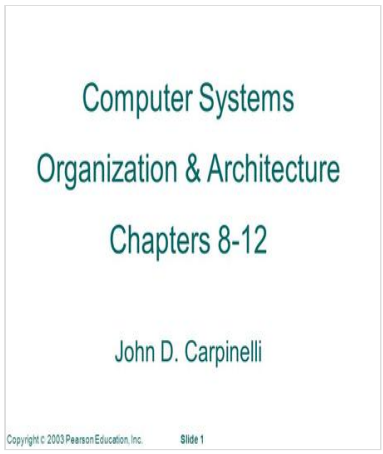
18 Application layers

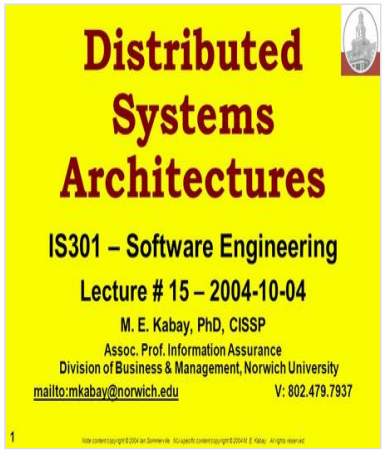
19 **Thin and fat clients Thin-client model Fat-client model**

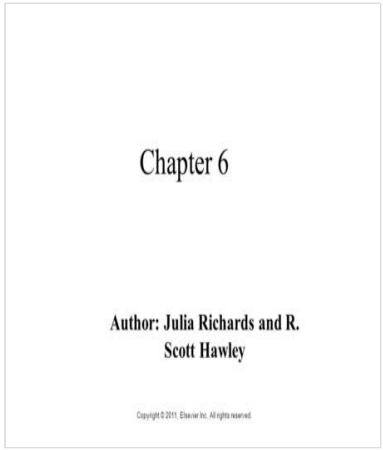
In a thin-client model, all of the application processing and data management is carried out on the server. The client is simply responsible for running the presentation software.

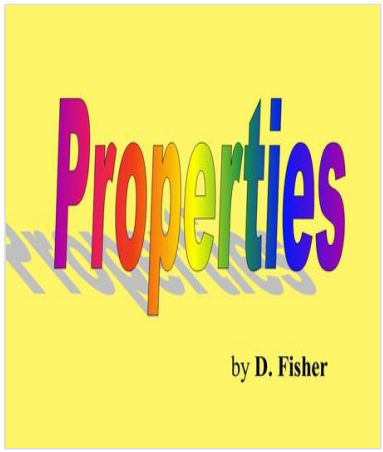
Fat-client model

In this model, the server is only responsible for data management. The software on the client implements the application logic and the interactions with the system user









The legacy system acts as a server in its own right with a graphical interface implemented on a client
A major disadvantage is that it places a heavy processing load on both the server and the network

22 Fat client model

More processing is delegated to the client as the application processing is locally executed
Most suitable for new C/S systems where the capabilities of the client system are known in advance
More complex than a thin client model especially for management. New versions of the application have to be installed on all clients

23 **A client-server ATM system**

24 **Three-tier architectures**

In a three-tier architecture, each of the application architecture layers may execute on a separate processor
Allows for better performance than a thin-client approach and is simpler to manage than a fat-client approach
A more scalable architecture - as demands increase, extra servers can be added

25 **A 3-tier C/S architecture**

26 **An internet banking system**

27 **Use of C/S architectures**

28 **Distributed object architectures**

There is no distinction in a distributed object architectures between clients and servers
Each distributable entity is an object that provides services to other objects and receives services from other objects
Object communication is through a middleware system called an object request broker (software bus)
However, more complex to design than C/S systems

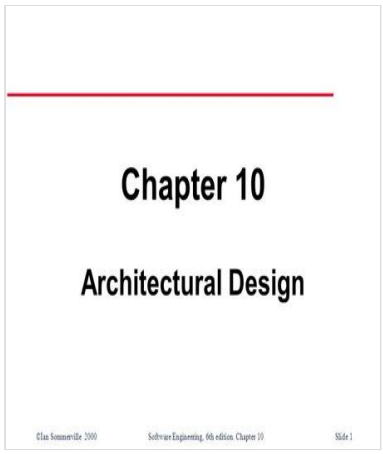
29 **Distributed object architecture**

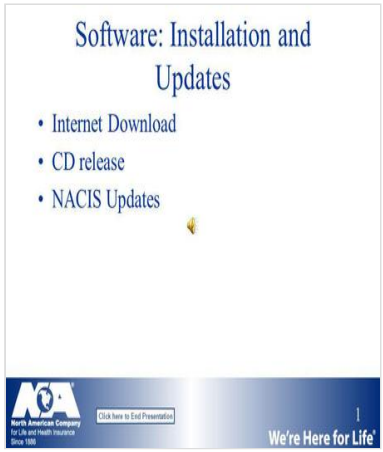
30 **Advantages of distributed object architecture**

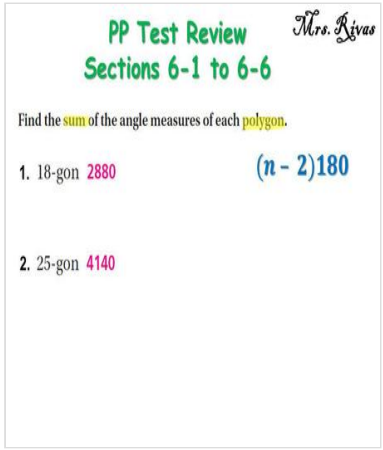
It allows the system designer to delay decisions on where and how services should be provided
It is a very open system architecture that allows new resources to be added to it as required
The system is flexible and scaleable
It is possible to reconfigure the system dynamically with objects migrating across the network as required

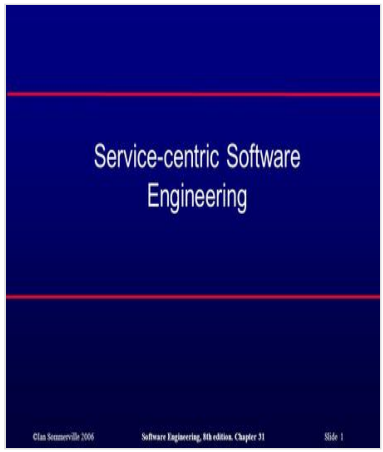
31 **Uses of distributed object architecture**

As a logical model that allows you to structure and organise the system. In









32 A data mining system

33 Data mining system

The logical model of the system is not one of service provision where there are distinguished data management services

It allows the number of databases that are accessed to be increased without disrupting the system

It allows new types of relationship to be mined by adding new integrator objects

34 CORBA

CORBA is an international standard for an Object Request Broker - middleware to manage communications between distributed objects

Several implementation of CORBA are available

DCOM is an alternative approach by Microsoft to object request brokers

CORBA has been defined by the Object Management Group

35 **Application structure**

Application objects

Standard objects, defined by the OMG, for a specific domain e.g. insurance

Fundamental CORBA services such as directories and security management

Horizontal (i.e. cutting across applications) facilities such as user interface facilities

36 **CORBA application structure**

37 **CORBA standards An object model for application objects**

A CORBA object is an encapsulation of state with a well-defined, language-neutral interface defined in an IDL (interface definition language)

An object request broker that manages requests for object services

A set of general object services of use to many distributed applications

A set of common components built on top of these services

38 CORBA objects

CORBA objects are comparable, in principle, to objects in C++ and Java

They MUST have a separate interface definition that is expressed using a common language (IDL) similar to C++

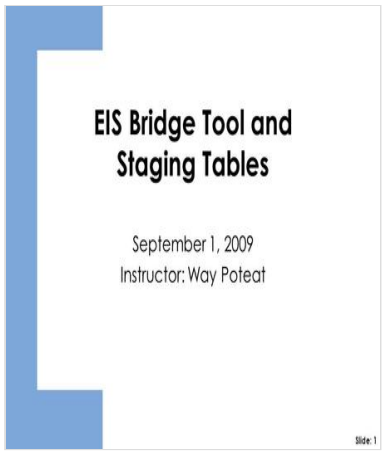
There is a mapping from this IDL to programming languages (C++, Java, etc.)

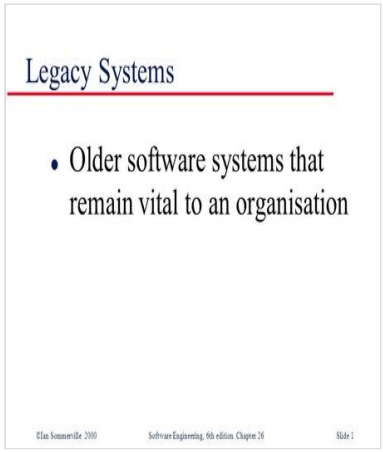
Therefore, objects written in different languages can communicate with each other

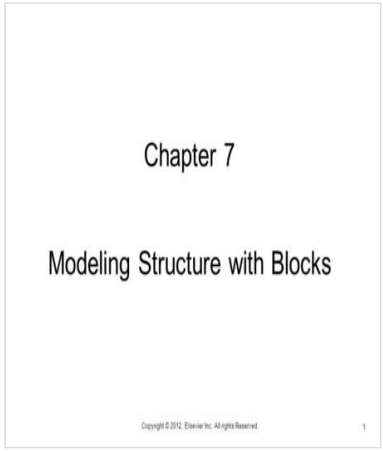
39 **Object request broker (ORB)**

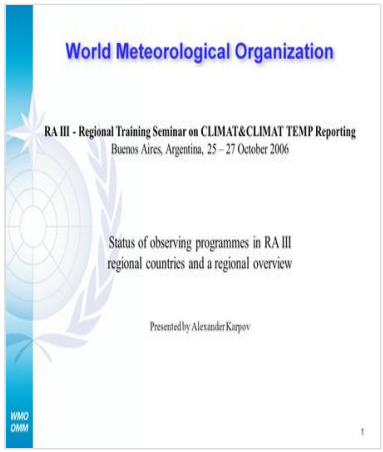
The ORB handles object communications. It knows of all objects in the system and their interfaces

Using an ORB, the calling object binds an IDL stub that defines the interface of the called object









41 **Inter-ORB communications**

ORBs are not usually separate programs but are a set of objects in a library that are linked with an application when it is developed

ORBs handle communications between objects executing on the sane machine

Several ORBS may be available and each computer in a distributed system will have its own ORB

Inter-ORB communications are used for distributed object calls

42 **Inter-ORB communications**

43 **CORBA services Naming and trading services Notification services**

These allow objects to discover and refer to other objects on the network

Notification services

These allow objects to notify other objects that an event has occurred

Transaction services

These support atomic transactions and rollback on failure

44 **Key points Almost all new large systems are distributed systems**

Distributed systems support resource sharing, openness, concurrency, scalability, fault tolerance and transparency

Client-server architectures involve services being delivered by servers to programs operating on clients

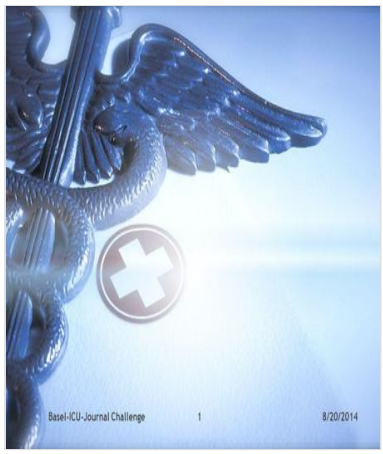
User interface software always runs on the client and data management on the server

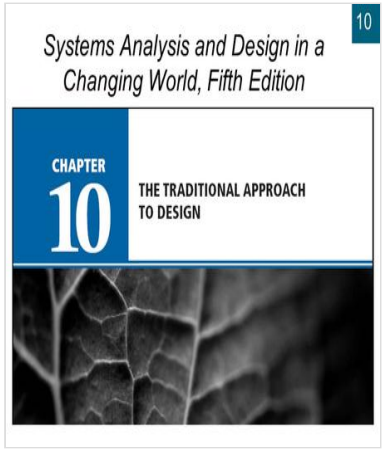
45 Key points

In a distributed object architecture, there is no distinction between clients and servers

Distributed object systems require middleware to handle object communications

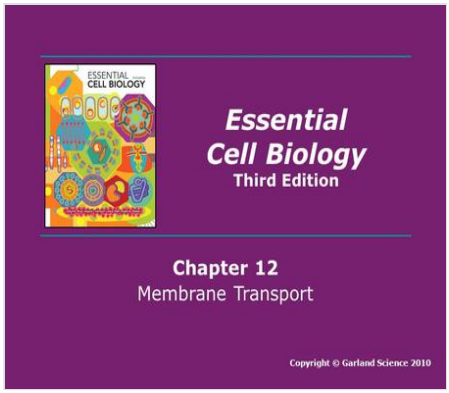
The CORBA standards are a set of middleware standards that support distributed object architectures

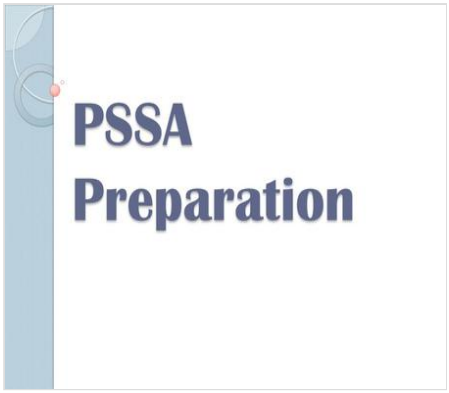


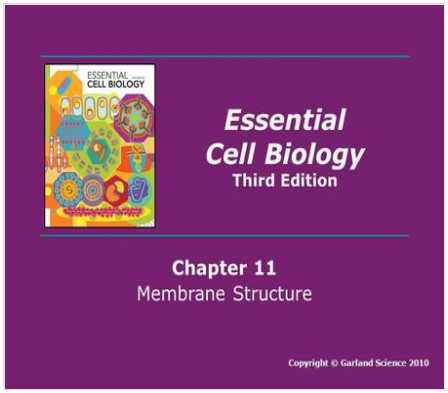


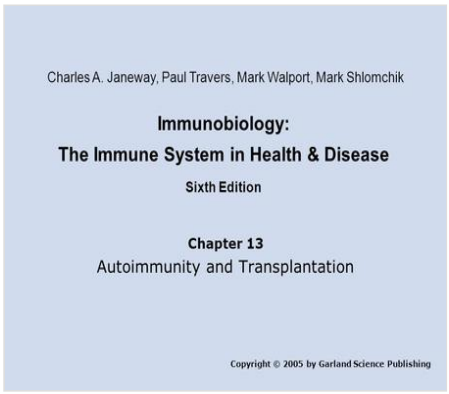
Download ppt "Distributed Systems Architectures"

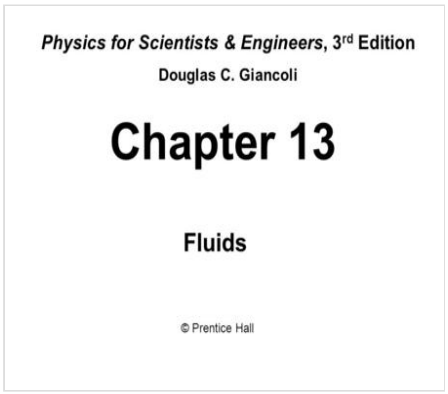
Similar presentations

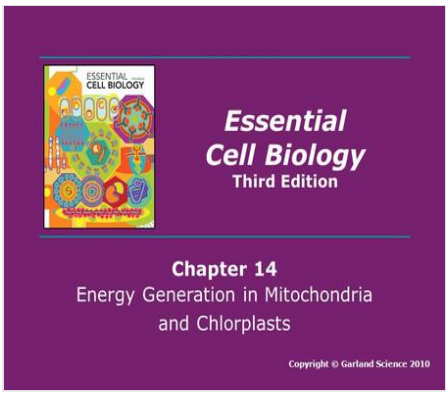




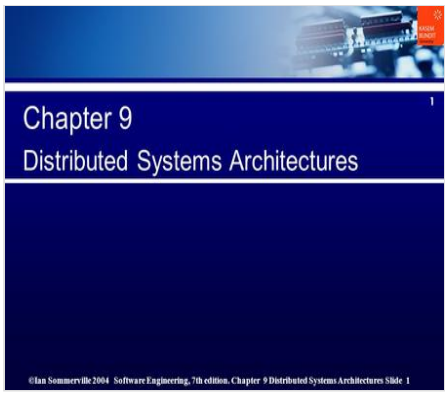







ESSENTIAL CELL BIOLOGY
Essential Cell Biology
Third Edition
Chapter 14
Energy Generation in Mitochondria and Chlorplasts
Copyright © Garland Science 2010

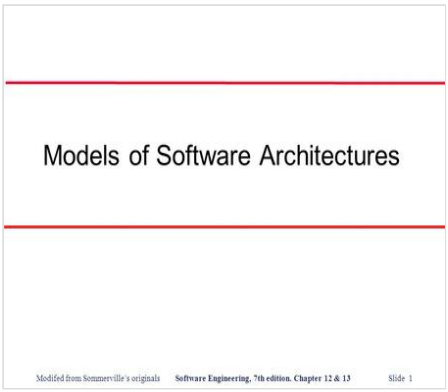

Distributed Systems Design 2
©Ian Sommerville 2004 Software Engineering, 7th edition. Chapter 12 Slide 1

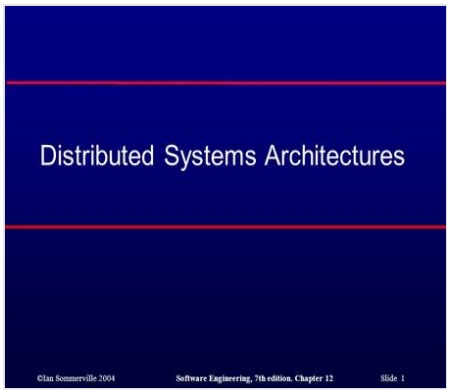

Chapter 9
Distributed Systems Architectures
1
©Ian Sommerville 2004 Software Engineering, 7th edition. Chapter 9 Distributed Systems Architectures Slide 1

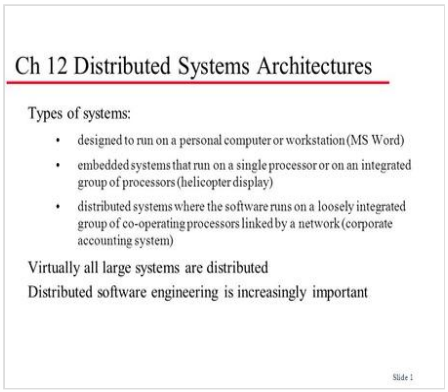

Software Engineering
COMP 201
Lecturer: Sebastian Coope
Ashton Building, Room G.18
E-mail: coopes@liverpool.ac.uk
COMP 201 web-page:
http://www.csc.liv.ac.uk/~coopes/comp201
Lecture 16 – Distributed System Architectures

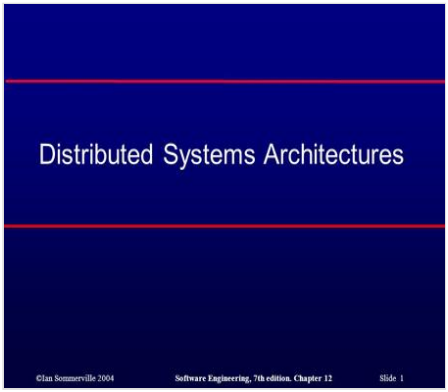

Distributed Systems Architectures

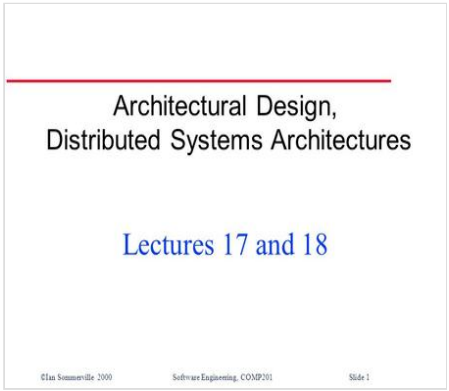

Distributed Systems Architectures
Architectural design for software that executes on more than one processor
Computer Science
CS425/CS625
8/23/2001 1


Models of Software Architectures
Modified from Sommerville's originals Software Engineering, 7th edition. Chapter 12 & 13 Slide 1

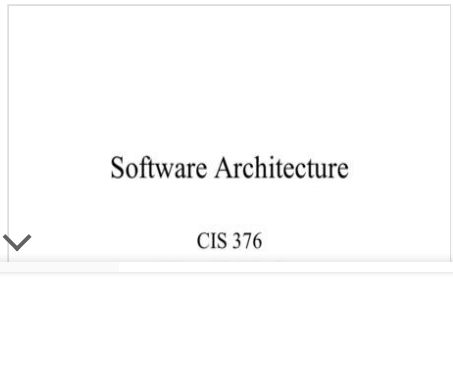

Distributed Systems Architectures
©Ian Sommerville 2004 Software Engineering, 7th edition. Chapter 12 Slide 1

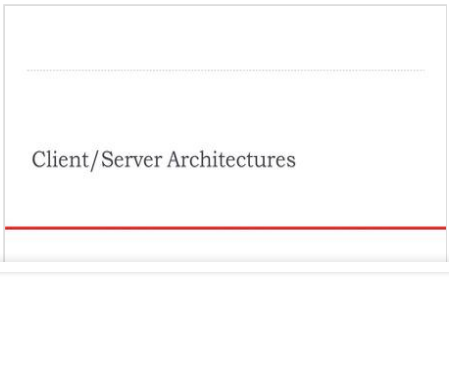

Ch 12 Distributed Systems Architectures
Types of systems:
designed to run on a personal computer or workstation (MS Word)
embedded systems that run on a single processor or on an integrated group of processors (helicopter display)
distributed systems where the software runs on a loosely integrated group of co-operating processors linked by a network (corporate accounting system)
Virtually all large systems are distributed
Distributed software engineering is increasingly important
Slide 1

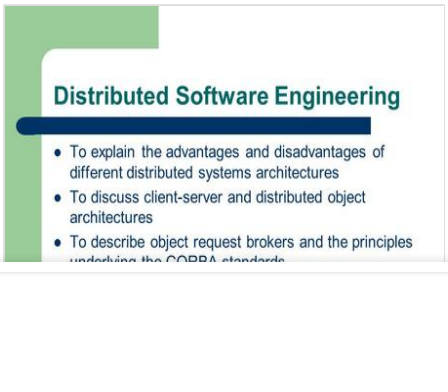

Distributed Systems Architectures
©Ian Sommerville 2004 Software Engineering, 7th edition. Chapter 12 Slide 1


Architectural Design,
Distributed Systems Architectures
Lectures 17 and 18
©Ian Sommerville 2000 Software Engineering, COMP201 Slide 1


Distributed Systems Design 1
©Ian Sommerville 2004 Software Engineering, 7th edition. Chapter 12 Slide 1


Software Architecture
CIS 376


Client/Server Architectures


Distributed Software Engineering
To explain the advantages and disadvantages of different distributed systems architectures
To discuss client-server and distributed object architectures
To describe object request brokers and the principles