

CS3773 Software Engineering

Lecture 18 Software Architecture

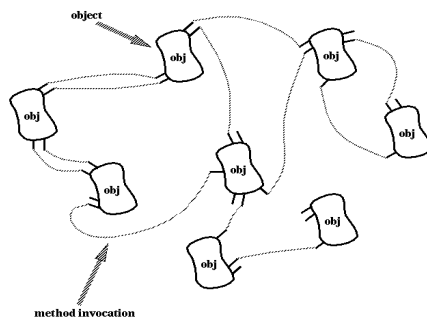
Object-Oriented Style

- Suitable for applications in which a central issue is identifying and protecting related bodies of information (data)
- Data representations and their associated operations are encapsulated in an abstract data type
- Components: are objects
- Connectors: are function and procedure invocations (methods)

2

UTSA CS3773

Object-Oriented Style



3

UTSA CS3773

Object-Oriented Invariants

- Objects are responsible for preserving the integrity (*e.g.*, some invariant) of the data representation
- The data representation is hidden from other objects

4

UTSA CS3773

Object-Oriented Specializations

- Distributed Objects
- Objects with Multiple Interfaces

5

UTSA CS3773

Object-Oriented Advantages

- Because an object hides its data representation from its clients, it is possible to change the implementation without affecting those clients
- Can design systems as collections of autonomous interacting agents

6

UTSA CS3773

Object-Oriented Disadvantages

- In order for one object to interact with another object (via a method invocation) the first object must know the identity of the second object
 - Contrast with Pipe and Filter Style
 - When the identity of an object changes it is necessary to modify all objects that invoke it
- Objects cause side effect problems:
 - e. g., *A* and *B* both use object *C*, then *B*'s affect on *C* look like unexpected side effects to *A*

7

UTSA CS3773

Implicit Invocation Style

- Suitable for applications that involve loosely-coupled collection of components, each of which carries out some operation and may in the process enable other operations
- Particularly useful for applications that must be reconfigured on the fly:
 - Changing a service provider
 - Enabling or disabling capabilities

8

UTSA CS3773

Implicit Invocation Style (Cont'd)

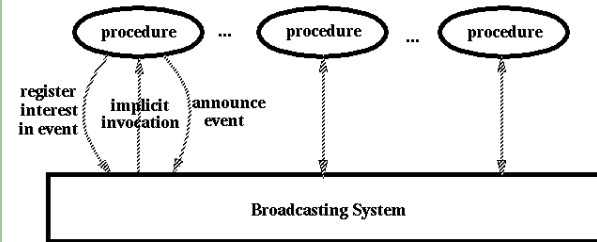
- Instead of invoking a procedure directly ...
 - A component can announce (or broadcast) one or more events
 - Other components in the system can register an interest in an event by associating a procedure with the event
 - When an event is announced, the broadcasting system (connector) itself invokes all of the procedures that have been registered for the event

9

UTSA CS3773

Implicit Invocation Style (Cont'd)

- An event announcement "implicitly" causes the invocation of procedures in other modules.



10

UTSA CS3773

Implicit Invocation Invariants

- Announcers of events do not know which components will be affected by those events
- Components cannot make assumptions about the order of processing
- Components cannot make assumptions about what processing will occur as a result of their events (perhaps no component will respond)

11

UTSA CS3773

Implicit Invocation Specializations

- Often connectors in an implicit invocation system also include the traditional procedure call in addition to the bindings between event announcements and procedure calls

12

UTSA CS3773

Implicit Invocation Examples

- Used in programming environments to integrate tools:
 - Debugger stops at a breakpoint and makes that announcement
 - Editor responds to the announcement by scrolling to the appropriate source line of the program and highlighting that line

13

UTSA CS3773

Implicit Invocation Examples (Cont'd)

- Used to enforce integrity constraints in database management systems (called triggers)
- Used in user interfaces to separate the presentation of data from the applications that manage that data

14

UTSA CS3773

Implicit Invocation Advantages

- Provides strong support for reuse since any component can be introduced into a system simply by registering it for the events of that system
- Eases system evolution since components may be replaced by other components without affecting the interfaces of other components in the system

15

UTSA CS3773

Implicit Invocation Disadvantages

- When a component announces an event:
 - it has no idea what other components will respond to it
 - it cannot rely on the order in which the responses are invoked
 - it cannot know when responses are finished

16

UTSA CS3773

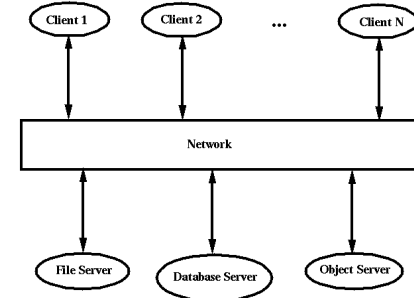
Client-Server Style

- Suitable for applications that involve distributed data and processing across a range of components
- Components:
 - Servers: Stand-alone components that provide specific services such as printing, data management, etc.
 - Clients: Components that call on the services provided by servers
- Connector: The network, which allows clients to access remote servers

17

UTSA CS3773

Client-Server Style



18

UTSA CS3773

Client-Server Style Examples

- File Servers:
 - Primitive form of data service
 - Useful for sharing files across a network
 - The client passes request for files over the network to the file server

19

UTSA CS3773

Client-Server Style Examples (Cont'd)

- Database Servers:
 - More efficient use of distributing power than file servers
 - Client passes SQL requests as messages to the DB server results are returned over the network to the client
 - Query processing done by the server
 - No need for large data transfers
 - Transaction DB servers also available

20

UTSA CS3773

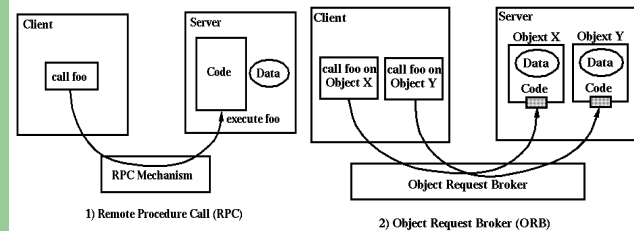
Client-Server Style Examples (Cont'd)

- Object Servers:
 - Objects work together across machine and network boundaries
 - ORBs allow objects to communicate with each other across the network
 - IDLs define interfaces of objects that communicate via the ORB
 - ORBs are the evolution of the RPC

21

UTSA CS3773

RPCs Versus ORBs



22

UTSA CS3773

Client-Server Advantages

- Distribution of data is straightforward
- transparency of location
- mix and match heterogeneous platforms
- easy to add new servers or upgrade existing servers

23

UTSA CS3773

Client-Server Disadvantages

- No central register of names and services -- it may be hard to find out what services are available

24

UTSA CS3773