A Warning
If You Ask About COM vs CORBA, You WILL Receive A Beating.

I only show the differences to get everyone to shut up.
COM

Single Vendor

Single Platform

Extensive Library of Desktop Controls

Well-Enriched on Desktop

A Desktop System

An C++ ABI For x86

Only one that really matters
A Standard, not a Product
Enterprise Use
Embedded, Desktop, and
Other Standards
Designed to Mix with
Completely Interoperable
Multi-Platform
Multi-Vendor
CORBA In the Real World

Who Uses It, and How You Can Too
CORBA Users

Or anywhere else in the Real World.

- Global Television News Media
- Oil Reservoir Search Networks
- Medical Data Sharing
- Banking Infrastructure
- Video On Demand, Videoconferencing
- Weapon Systems
- Avionics: Fighter Jets, Commercial Aircraft

CORBA Users
Example Developer Usage

1. Design Your System
2. Write IDL for Your Interfaces
3. Compile the IDL into Stubs and Skeletons
4. Implement the Skeletons
5. Link Your Clients Against the Stubs

Which forward requests to the servers
These become Your Servers
Example Project (1)

Server returns a string, which is always "Hello World"

Yes this is dumb

Client calls server method, and prints out "Hello World"

Server returns a string, which is always
Example Project (2)

```idl
interface DumbExample { string guessWhatIReturn(in long uselessIntegerValue); }
```

Run your IDL compiler on DumbExample.idl, and it creates:

- Stub Files: DumbExample_impl.{h,cpp}
- Skeleton Files: DumbExample-impl.{h,cpp}

In long uselessIntegerValue:

```idl
 chức guesWhatIReturn) } interface DumbExample:

def DumbExample(idl:
```
Example Project (3)  

```cpp
#include "DumbExample.h"

int main(int args, char ** argv) {
  // Init your Object Request Broker
  CORBA::ORB_var orb = ...;
  // call the interface method
  CORBA::String_var s = de->guessWhatIReturn(0);
  // print out results.
  cout << s;
  
  DumbExample_var de = DumbExample::narrow(0);
  Narrow object into a DumbExample
  
  DumbExample_var oo = orb->string_to_object(argv[1]);
  Convert string to object reference to an object
  
  CORBA::ORB_var orb = CORBA::ORB_init( argc, argv );
  // Init your Object Request Broker
  
  cout << sizeof DumbExample "DumbExample.h"
  
  Client.cpp
```

POA-DumbExample is an auto generated base class that handles the memory management for you when you return them. The *var classes have _retn() functions to automatically take care of CORBA work for you.

• POA-DumbExample is an auto generated base class that handles the memory management for you when you return them. The *var classes have _retn() functions to automatically take care of CORBA work for you.

Example Project (4)
Example Project (5)
Example Project (6)

CORBA Server.cpp (Part 2 of 2):

// An instance of the actual serving object
DumbExample_impl servant;

// get a normal object reference to it.
DumbExample_var de = servant._this();

// print out a stringified object reference to our servant
CORBA::String_var s = orb->object_to_string(de);
cout << s << endl;

// start handling requests.
orb->run();
CORBA Quality Of Service

When you care about more than a button's redraw latency
In essence, an ability to specify priorities, deadlines, and routing specifications to object method calls. It’s defined in the CORBA Messaging Standard[1].

What?
Some data is expendable—Like RT video frames for the on-flight movie.

Some data is extremely important—Like flight control instructions to avoid that mountain.

Why?
How?

• Set up policies for the client and/or server side
  - Server side sets up policies in its POA
  - Client can set its policies in a few places
    - The client’s ORB
    - A specific object reference
    - The local thread

• Set up policies for the client and/or server
Definitions

Message Router
- an intermediate object that queues and forwards requests onto other objects.
- Often useful when remote clients aren't immediately ready

One-Way Request
- a request type where the client is told not to wait for a response.
- Queue may be persistent


QoS Policies

• Queue Order
• Request Routing
• Request & Reply timeouts
• Request & Reply priority
• Synchronization for one-way operations
• Rebind policy

QoS Policies
Rebind Policy

What happens when the connection drops or the target object has moved? Do we
rebind explicitly?

NO-RECONNECT: Reconnection and rebindings, but don't do any rebinds to new remote objects if their QoS policies are different.

NO-REBIND: Autoreconnect, but do all the target object has moved?

TRANSPARENT: ORB does all.
Synchronization For One-Way Operations

- When should calls to one-way operations return?

  - `SYNC_NONE` - before passing request to client-side
    - transport layer (e.g., TCP)
  - `SYNC_WITH_TRANSPORT` - after passing request to client-side
  - `SYNC_WITH_Target` - after the remote object has completed the operation (but before the remote object has received the request)
  - `SYNC_WITH_SERVER` - after the server-side ORB finished the request
Request & Reply Priority

- Messages can go through Message Routers on their way to the destination.
- This priority controls their queue ordering.
- [1] Routers

Reply Priority

Request Priority
For synchronous calls, one can specify a maximum timeout to wait before returning.

- A CORBA::TIMEOUT exception is raised in case the call doesn’t complete in time.
- Standard specifications specify an accuracy of 100 ns.
Request Routing

- You can specify whether or not Message Routers are used to deliver the message:
  - ROUTE_NONE - don't use Routers
  - ROUTE_FORWARD - router passes request on
  - ROUTE_STORE_AND_FORWARD - router persists stores until forwarded
Queue Ordering

- ORDER_DEADLINE - sort by shortest timeout
- ORDER_TEMPORAL - in the order requests were specified
- ORDER_PRIORITY - use the priority values were given
- ORDER_ANY - order is not important
- ORDER_TEMPORAL - in the order requests were specified precisely specified

When in a Router's queue, ordering can be
References
CORBA QoS • Chapter 22 of the CORBA Standard

Pretty useful.


Actually all the articles on Douglas C. Schmidt and Steve Vinoski
Object Interconnections, Column 19 by Messaging
[1] Chapter 22 of the CORBA Standard:
CORBA Programming

• Java Programming with CORBA
  Andreas Vogel and Keith Duddy

• Advanced CORBA Programming with C++
  Michi Henninger & Steve Vinoski

• Advanced CORBA Programming with C++

CORBA Programming
CORBA ORBs

- TAO: The ACE ORB
  - Don't bother with ORBit either. Just use TAO.
  - Don't bother, it sucks

  - MICO
    - Good commercial Embedded Realtime ORB
      - http://www.ois.com

  - Objective Interface
    - Good commercial Enterprise and Embedded ORBs
      - http://www.iona.com

  - Iona's ORBIX
    - Good Free Realtime ORB
      - http://www.cs.wustl.edu/~schmidt/TAO.html

  - TAO: The ACE ORB

CORBA ORBS