A Tutorial on CORBA

Mark Plesko
J. Stefan Institute, Ljubljana, Slovenia

presented at the ESO
Garching, December 16-th, 1999
Summary

• Introduction
  – Justification and History (=blah, blah)
  – What is CORBA (Executive summary)
  – How does CORBA work (Programmer summary)
  – CORBA Features

• Concepts of CORBA
  – What are Objects in CORBA
  – Data Flow in CORBA
  – Definitions

• CORBA details
  – Request Invocation
  – Object References
  – The Portable Object Adapter (POA)

• More About CORBA
Justification and History (=blah, blah)

- Distributed Applications are heterogeneous
  - layers, applications, libraries glued together
  - can all components really work together?

- Two key rules
  - build platform-independent models and abstraction
  - hide as much low-level complexity without sacrificing too much performance

- CORBA provides a well thought **balanced** set of abstractions and concrete services
  - Object Services
  - Domain Interfaces
  - Application Interfaces

- Object Management Group (OMG) since 1989 - now over 800 members
What is CORBA (Executive summary)

- **ORB**: Object Request Broker = manages remote access to objects
- **CORBA**: Common ORB Architecture = software bus for distributed objects
- CORBA provides a framework for distributed OO programming
  - remote objects are (nearly) transparently accessible from the local program
  - uses the client-server paradigm
  - platform and language independent
- “an OO version of RPC”
  - but a framework rather than a technology => lot of theory
How does CORBA work (Programmer summary)

**Interface** PS {
    attribute double current;
    readonly attribute Ulong status;
    octet on(in octet value);
    void test(out long result);
}

try {
    PS aPS = PSHelper.bind(ORB,“PS1”);
    if (!aPS.on(1)) return;
    aPS.set_current(3.1415);
    print(aPS.status());
} catch (CORBAexception) { … }

class PSimp extends PSImlBase{…};
…
PS thisPS = new PSimp(“PS1”);
BOA.obj_is_ready(thisPS);
BOA.impl_is_ready();
CORBA Features

• Don’t worry about unique terminology - these are just words!
  – CORBA object
  – request, target object, object reference
  – client, server, servant

• Features
  – Interface Definition Language (IDL)
  – language mapping
    • official: C, C++, Smalltalk, COBOL, Ada, Java
    • also: Eiffel, Modula 3, Perl, Tcl, Objective-C, Python
  – Operation invocation and dispatch facilities
    • static (known at compile-time)
    • dynamic (determined at run-time)
  – Object adapters
    • Design pattern: adapt CORBA object interface to servant
  – Inter-ORB Protocol
What are Objects in CORBA

- Objects are abstract: not realized by any particular technology
  - An object system is a collection of objects that isolates the requestor of services (clients) from the providers of services by a well-defined **encapsulating interface**
- Objects “talk” through requests: operation, target object, zero or more parameters, optional request context
- Objects are described with interfaces
  - operations (methods)
  - attributes (properties)
  - Standard data types are supported
    - object references
    - Any
Data Flow in CORBA

[Diagram showing data flow between a client and an object implementation through IIOP.]
Some Definitions

- **ORB:**
  - find the object implementation for the request, prepare the object implementation to receive the request and communicate the data making up the request.
  - ORB throws exceptions
  - ORB implementation is not defined in CORBA

- **Object Adapter (POA, BOA, ...)**
  - provides ORB services to particular groups of object implementations
  - generation and interpretation of object references, method invocation, security of interactions, object and implementation activation and deactivation, mapping object references to implementations, and registration of implementations.

- **IIOP: Internet Inter-ORB Protocol**
  - ORB’s of different vendors can talk
  - TCP/IP implementation of GIOP
More Definitions

- **IDL: Interface Definition Language**
  - IDL is the means by which a particular object implementation tells its potential clients what operations are available and how they should be invoked.

- **Language mapping: recipe how to generate stubs&skeletons from IDL**
  - Clients see objects and ORB interfaces through the perspective of a language mapping, bringing the object right up to the programmer’s level.

- **Interface Repository: where all interfaces are stored network-wide**
  - provides information on interfaces at run-time

- **DII: Dynamic Invocation Interface**
  - construct a remote method call at run-time without the use of stubs
Request Invocation

This is transparently handled by the ORB

- Locate target object
- activate server application if not yet running
- transmit any arguments
- activate a servant if necessary
- wait for request to complete
- return any out/inout parameters and return value
- return exception if call fails
Object References

- Several references to one object
- Can point to nowhere (death undetected)
- Are strongly typed (at compile&run time)
- Support late binding
- Implemented by proxies

- But how do you get a reference?
  - Bootstrap
    - via well known entry point (Naming service)
    - via reference-to-string (known URL, filename)
  - from a Object method call
The Portable Object Adapter (POA)

- Provides object creation, servant registration and mapping, request dispatching
- Intended for scalable, high-performance applications
  - different POAs for 1 object or millions of objects
- Is a locally-constrained object, multiple may exist
- Policies
  - Object life span: persistent/transient
  - Object Id: system_ID/user_ID
  - Mapping objects to servants: unique_ID/multiple_ID
  - Object activation: implicit/no_implicit
  - Matching requests to servants: object_map/default_servant/manager
  - Object to servant association: retain/non_retain
  - allocation of threads: ORB_control/single_thread
CORBA Services

- Some 20+ defined services
- check vendor for implementation and limitations!
- Some interesting services
  - Naming Service
    - “directory-based”
    - single or federated
  - Event Service
    - decouples suppliers from consumers
    - push or pull models
    - uses Any for event data
    - Notification Service?
    - Messaging Service?
More About CORBA

• Other features of CORBA
  – vendor specific implementations - check performance you need!
  – Gateways to DCOM and OLE automation exist
  – CORBA Components (futureware)

• Some buzzwords to know (and use)
  – thin client
  – three tier architecture
  – legacy systems

• Alternatives to CORBA:
  – sockets low level, used by CORBA
  – RPC not OO
  – RMI language dependent
  – DCOM maybe someday
Meta IDL - MIDL

```idl
#parameter P<type>|<type>{
    #accessors{#sync, #async, #history};
    #monitorable;
    #static{default_value, graph_min, graph_max, min_step,
    resolution|pattern, description|string, format|string, units|string};
};
#parameter RW<type>|<type>: P<<type>>{
    #eventable{Alarm<<type>>};
    #mutators{#sync, #async, #nonblocking, #step};
    #static{min_value, max_value};
};

#device PowerSupply{
    #actions{on, off, reset, start_ramp(in CBRamp cb, in RampData data)};
    #methods{double sync_method_test(in double input, out double output)};
    #parameters{current|RW<double>, readback|RO<double>, status|ROpattern};
    #static{model|PowerSupplyModel};
};
```
Callbacks in BACI: device.property.get(CB)

- Asynchronous completion notification

```java
interface CB<type> : CB {
    oneway void execute(in <type> value, in Completion c, in CBDescOut desc);
    oneway void cb_done(in <type> value, in Completion c, in CBDescOut desc);
}
```

- monitoring

- events

```java
interface CB<event_set_name> : CB {
    oneway void <event_1_name>(..., in CBDescOut desc);
    oneway void <event_2_name>(..., in CBDescOut desc);
    ...
}
```

```java
... void subscribe_<event_set_name>(in CB<event_set_name> cb, in CBDescIn desc);
void unsubscribe_<event_set_name>(in CB<event_set_name> cb);
```
CoCoS Runtime

Diagram showing the relationships between DB, ADMIN, MANAGER, COB SERVER, and various COB modules.

- DB linked to ADMIN, MANAGER, and COB SERVER
- ADMIN linked to MANAGER, COB, and COB type A
- MANAGER linked to COB, COB type A, and COB type B
- COB SERVER linked to COB, COB type A, and COB type B
- COB linked to COB type A and COB type B
- COB type A and COB type B are separate modules.
CoCoS Startup and Management

[Diagram showing CoCoS components and interactions]
CoCoS on Pharlap/TNT Real-Time Operating System