CORBA: Object Adapter, Services, Inter-ORB Protocols

- Balaprasuna Chennupati
Object Adapter

- Glue between CORBA object implementation and the ORB
- Adapts the interface of an object to the interface expected by the ORB
- Caller can invoke requests on an object without knowing its true interface
- Caller can invoke requests on an object without knowing its true interface
- Delegation
Object Adapter adapts interface X to interface A

Caller expects interface A

Object provides interface X

Object Adapter adapts

Object
Object Adapter

- Object upcalls
- Request demultiplexing
- Object activation
- Server process activation
- Object reference generation
- Object registration
- Responsibilities

Object Adapter
Object Adapter

- Single OA... (hence, obsolete)
- Basic Object Adapter (BOA)
- Without OAs:
  - No diverse object implementation styles
  - Unnecessary size and complexity of ORB
  - No diverse object implementation styles

Need for Multiple OAs

- A different OA for each programming language
- A different OA for each programming language

Example: C object -> pointers, whereas, C++ object adapter base class

Object Adapter
CORBA object vs. Servant
Portable Object Adapter (POA)

- Allows servants to be portable among ORBs by different vendors
- Allows multiple POA instances to support ORBs by different vendors
- Allows servants to be portable among

Activation Styles:

- Default servant
- Implicit activation (invoke this method)
- On-demand activation (servant manager)
- Explicit activation (direct calls on the POA)

Activation Styles: Multiple servant implementation styles
Naming Service

- Provides a mapping from names to object references
- Access to the initial references in an application
- By changing the value of a reference, clients can use a different implementation without changing the source code
- Clients can deal with meaningful names instead of object references
- Provides a mapping from names to objects
Trading Service

- Dynamic object discovery facility
  - Allows clients to locate objects with help of a trader
  - Trader stores object reference and the description of the service provided by the object
  - Clients perform a dynamic lookup of services based on queries over the service descriptors

Dynamic object discovery facility
Asynchronous request invocations

Decoupled communication between objects

Asyncronous request invocations

Push model vs. Pull model
To create a transaction:

1. Call `transaction factory: create()`
2. Create a transaction
3. Control object
4. Terminator object
5. Coordinator object

Transaction Service
Inter-ORB Protocols

• No interoperability before CORBA 2.0
  – Lack of particular data formats or protocols
  – Direct ORB-to-ORB interoperability

Bridge based interoperability

  (type system, ...)

When two ORBs reside in the same domain

Inter-ORB Protocols
Inter-ORB Protocols

- **General Inter-ORB Protocol (GIOP)**
  - Over any connection-oriented transport
  - Simple and easy to implement, Reasonable scalability and performance

- **Common Data Representation (CDR)**
  - Format for each IDL data type

- **Transport assumptions**
  - Assumptions about the underlying transport layer
  - Over any connection-oriented transport
Inter-ORB Protocols

- Internet Inter-ORB Protocol (IIOP)
  - Specifies how GIOP is implemented over TCP/IP

- Interoperable Object Reference (IOR)
  - Standard object reference format
  - Stores information needed to locate and communicate with an object over one or more protocols
  - Ex: IOR containing IIOP information stores the Hostname & TCP/IP Port number

Inter-ORB Protocols
References

- Distributed computing
- CORBA – An industrial approach to open environments
- CORBA Programming with C++
- Michi Henning, Steve Vinoski, Advanced
- IEE: Steve Vinoski, CORBA: Integrating diverse applications within distributed heterogeneous environments
- IEEE: Steve Vinoski, CORBA; Integrating diverse
Thank You