The OMG Approach (continued)

CORBA, CCM, OMA, and MDA

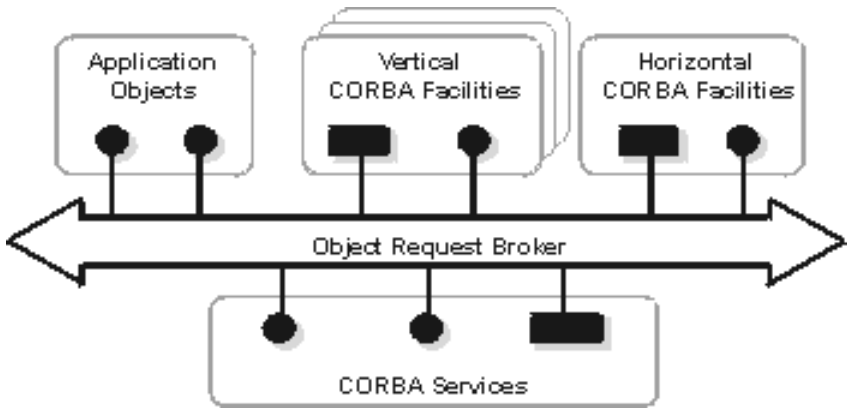
CORBA History

- CORBA 1 was all about object request brokers and IDL is its hallmark contribution.
- CORBA 2 focuses on interoperation (and interworking) and IIOP is its hallmark.
- CORBA 3 focuses on component and system integration and CCM is its hallmark.

Object Management Architecture

- Since CORBA 2, the OMG's overall effort is called the object management architecture (OMA).
- The OMA adds several new areas of standardization:
  - CORBA Services: object service specifications
  - CORBA Facilities: facility specifications
  - A set of application object specifications
  - CCM: the CORBA Component Model

Object Management Architecture (2)



The OMA Reference Model shown above identifies and characterizes the components, interfaces, and protocols that compose the OMA.

ORB

- Central to the OMA model - its communications heart - is the *Object Request Broker* (ORB) component that enables clients and objects to communicate in a distributed environment.
- The ORB provides an infrastructure allowing objects to communicate independent of the specific platforms and techniques used to implement the addressed objects.
- The ORB guarantees portability and interoperability of objects over a network of heterogeneous systems.
- It supports embedded and real-time systems as well as those requiring fault tolerance and other Quality of Service requirements.

CORBA Services

- The *CORBA Services* component standardizes the life cycle management of objects.
- Functions are provided to create objects, to control access to objects, to keep track of relocated objects and to consistently maintain the relationship between groups of objects.
- The CORBA Services components provide the generic environment in which single objects can perform their tasks.
- Standardization of CORBA Services leads to consistency over different applications and improved productivity for the developer.

CORBA Facilities

- Vertical CORBA Facilities represent components providing computing solutions for business problems within a specific vertical market (e.g., healthcare, manufacturing, finance).
  - Lists of Vertical CORBA Facility specifications are provided on the OMG web site.
- Horizontal CORBA Facilities represent those components providing support across an enterprise and across businesses.
  - A Digital Asset Management component serves as an example of such a component.

Application Objects

- The *Application Objects* part of the architecture represents those objects performing specific tasks for users.
- Application objects can invoke methods on remote objects either statically or dynamically in a distributed environment through the ORB.
- Application objects standardized by the OMG represent domain frameworks.
- An application is typically built from a large number of basic object classes. New classes of application objects can be built by modification of existing classes through generalization or specialization of existing classes (inheritance) as provided by CORBA Services.

Naming and Trader Services

- The naming service allows arbitrary names to be associated with an object. Names are unique within a naming context and naming contexts form a hierarchy.
- The resulting naming tree is quite similar to directory structures in file systems.
- The trader service allows providers to announce their services by registering offers. Clients can use a trader to locate services by description.
- A trader organizes services into trading contexts. Clients can search for services, based on parts of descriptions and keywords, within selected contexts.
- The naming service can be compared to White Pages and the trader service to Yellow Pages.

Event and Notification Services

- The event service allows event objects that can be sent from event suppliers to event consumers to be defined.
- Event objects are immutable in that information flows strictly in one direction, from supplier to consumer.
- Events travel through event channels that decouple supplier from consumer.
- Events can be typed (described using OMG IDL) and channels can be used to filter events according to their type.

Event and Notification Services (2)

- The event channel supports both the “push” and the “pull” model of event notification.
  - In the “push” model, the event supplier calls a push method on the event channel, which reacts by calling the push method of all registered consumers.
  - In the “pull” model, the consumer calls the pull method of the event channel, effectively polling the channel for events. The channel then calls a pull method on the registered suppliers and returns an event object if it found a supplier that returned an event object.

Event and Notification Services (3)

- The notification service includes support for:
  - Quality of service (QoS) specification and administration.
  - Standards for typed and structured events.
  - Dynamic event filtering based on type and QoS.
  - Filtering at source, channel, consumer group, and individual consumer level.
  - Event discovery among source, channel, and client.
- Technically, the notification “service” is not a CORBA service but a CORBA horizontal facility.

Object Transaction Service (OTS)

- The OTS automatically maintains a current transaction context that is propagated along with all ORB-mediated requests.
- The context is passed to any CORBA object that implements interface TransactionalObject.
- The transaction operations begin, commit, and rollback are defined on the current context.
- Instead of providing transaction control as a separate service, as promoted by the OTS design, it is now more common to integrate transaction services into a container abstraction provided by an application server (as in the CORBA Component Model).

Security Service

- The CORBA Security specification defines a number of services for tasks such as authentication, secure communication, delegation of credentials (also known as impersonation), and non-repudiation.
- Most products don’t actually support the full spectrum of security services in this specification.
  - Some simply rely on secure sockets layer (SSL).
  - Using SSL is fine to establish simple authentication and secure communication properties. However, it does not support more advanced concepts such as delegation or non-repudiation.

Other Services

- Concurrency control
- Licensing
- Lifecycle
- Relationship
- Persistent state
- Externalization
- Properties
- Object query
- Object collections
- Time

CORBA Component Model

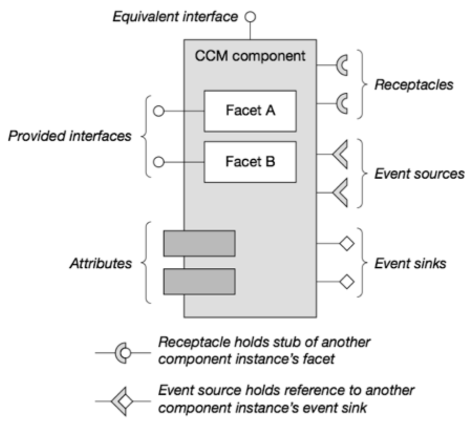
- CCM describes a standard application framework for CORBA components. It is similar to EJB, but more general, providing four component types instead of the two that EJB defines.
  - service, session, entity, and process components
- Enterprise JavaBeans components and CCM components can be combined in a single application.
- It provides an abstraction of entities that can provide and accept services through well defined named interfaces called *ports*.

Features of CCM Components

- A CCM component is programmatically characterized by a number of features:
- Ports that are classified into facets, receptacles, event sources, and event sinks.
  - A facet is a provided interface and a receptacle is a required interface. A component instance's receptacles are connected to other instances' facets. A special facet of a CCM component is the *equivalent interface*, which enables navigation between the different facets of a CCM component.
  - Event sources and event sinks are similar, but instead of being connected to each other, they are both connected to event channels.

Features of CCM Components (2)

- Primary keys, which are values that instances of entity components provide to allow client identification of the instances.
- Attributes, which are named values exposed via accessors and mutators.
- Home interfaces, which provide factory functionality to create new instances.


Equivalent interface
CCM component
Receptacles
Facet A
Provided interfaces
Facet B
Event sources
Attributes
Event sinks
Receptacle holds stub of another component instance's facet
Event source holds reference to another component instance's event sink

CCM Containers

- Every CCM component instance is placed inside a CCM container.
- Containers provide transactions, security, persistence, and notification services to components via interfaces on the container.
- A number of options are available for each of the four services that CCM packages.
  - Transaction control can be container-managed or self-managed. In the container-managed case, the container will begin and end transactions to meet requests.
  - Similarly, persistence can be declared as container-managed or self-managed.
  - For security, required access permissions can be declared on operations and will be checked by the container.

Model Driven Architecture

- The idea of MDA is for specifications to be written at two levels, namely platform-independent models (PIMs) and corresponding platform-specific models (PSMs.
- The hope is that business processes and entities can be modeled at the PIM level to a degree of precision that then enables the automatic generation of large parts of implementations for a variety of platforms, driven by PIM-level models and PIM-to-PSM mappings for the target platform.
- An MDA goal is to "embrace CORBA, J2EE, XML, .NET and other technologies".
  - It remains to be seen to what extent this goal can be met.