분산객체관리 표준으로 'CORBA' 부상

OMG는 객체지향 환경에 대한 프레임웍의 제시와 이를 업계 표준으로 정착시키기 위해 업체간의 토론, 교육, 그리고 객체 기술의 전파 등을 목적으로 설립된 단체이다. 이번호에는 OMG에서 제시하는 객체지원환경 표준에 대해 살펴봤다.

이동진/ 한국CIM 대표이사

연 · 재 · 순 · 서

1. 분산객체관리 표준

1.1 개 요

OMG(Object Management Group)는 현재 약 500여개의 정보산업 관련업체들과 대학, 정부기관, 정보시스템 사용자들의 컨소시엄으로서 분산 컴퓨팅을 위한 객체 기술의 표준화를 위한 모임이다. OMG는 크게 다음과 같은 목적을 가지고 구성되었다.

먼저 상용 객체지향 환경에 대한 새로운 프레임웍을 제시하고 이에 대한 자세한 명세를 제시함으로써 다양한 소프트웨어들 사이의 이식성(Portability), 재사용성(Reusability), 상호운용성(Interoperability)의 극대화를 목표로 한다.

또 OMG는 객체기술에 대한 참조모형(Reference Model)과 용어, 정의 등을 제시하고, 이를 업계 표준으로 정착시키고자 하며 업체간의 토론, 교육, 그리고 여러 가지 객체 기술의 전파 등을 그 목적으로 하고 있다.

OMG 표준안의 특징은 하드웨어, 운영체제, 응용 프로그램 등의 모든 수준에서의 동질성을 추구하지 않는다는데 있다. 대신 기존의 컴퓨터들의 이질성을 인정하고 기존의 인터페이스 위에 하나의 접속 표준을 제시한다.

OMG의 표준화 작업은 구체적으로 아래와 같은 세부적인 기술적 목표를 가지고 이루어졌다.

● 표준안은 OMG의 객체 모형에 부합해야 하며, 특히 다음과 같은 기능들이 제공되어야 한다.

- 인터페이스의 상속(Inheritance of interface)
- 구현(Implementation)의 상속
- 프로그램이나 프로세스가 하나 이상의 메소드(Method)를 정의하고 실행할 수 있

어야 한다.

- 객체의 메소드들은 하나 이상의 프로그램이나 프로세스에서 정의되고 실행된다.

● 객체의 분산은 투명성(Transparency)이 보장되어야 한다. 즉 다른 객체를 이용하고자 하는 클라이언트 객체는 아래와 같은 서버 객체의 특성에 대해 무관하게 서비스를 요청할 수 있어야 한다.

- 위치(Location)
- 접근 경로(Access path)
- 객체 재배치(Relocation)의 독립성
- 서버 객체의 자료와 코드의 표현(Representation)에 무관하게 서비스를 받음
- 통신 매커니즘에 무관한 서버 객체의 사용
- 실행 매커니즘에 무관한 서버 객체의 사용
- 저장 매커니즘에 무관한 서버 객체의 사용
- 프로그래밍 언어
- 기종(Machine type)
- 운영 체제
- 보안 매커니즘에 무관한 서버 객체의 사용

● 원격 객체 요청(Remote object request)에 대한 성능 보장이 됨

● 객체의 확장성(Extensibility)이 고려됨

- 새로운 클래스의 추가가 기존의 객체에 영향을 미치지 않고 이루어짐
- 객체의 구현을 변경할 때, 이 작업이 객체의 인터페이스를 수정하지 않는 한 이 객체를 사용하는 다른 클라이언트 객체는 변경되지 않아야 함
- 서버 객체의 코드와 자료의 위치가 바뀌더라도 다른 클라이언트 객체는 이에 무관하게 그 객체를 접근할 수 있어야 함

● 다수 개의 이름 문맥(Naming Context)을 제공하는 이름 시스템을 제공

- 객체가 다수 개의 이름을 가지고 있더라도 그 객체를 유일하게 표시하는 객체 핸들의 개념을 포함해야 함
- 객체를 효율적으로 참조할 수 있는 방법의 제시
- 참조 아키텍쳐는 각 개체를 모호하지 않게 지정할 수 있는 이름 공간(Naming space)을 제공

● 질의(Query)는 객체의 이름이나 속성 그리고 다른 객체와의 관계에 기반해야 함

● 서버 객체에 대한 이벤트 공지(Event notofication)기능이 지원되어야 함

● 분산 객체에 대한 오퍼레이션의 원자성(Atomicity)의 보장을 위하여 트랜잭션 관리 기능이 포함되어야 함

● 분산 객체에 대한 병행성 제어(Concurrency control) 매커니즘이 제공되어야 함

- 클라이언트 측에게 병행성 제어에 대한 책임을 넘겨서는 안됨
- 제어 매커니즘은 서버 객체 사용의 병렬성을 최대화 시켜야 함
- 비정상적인 상황에서의 복구(Recovery) 기능이 마련되어야 함

● 객체에 대한 접근 제어를 통해 보안 기능이 마련되어야 함

- 객체의 자료는 객체 클래스에서 정의된 메소드를 통해서만 접근할 수 있어야 함
- 구현 또는 인터페이스 정보와 같은 메타 객체 자료에 대해서도 접근 제어가 이루어져야 함

● 높은 가용성(Availability)과 실행의 일관성(Consistency)을 위해 복구 기능이 제공되어야 함

● 객체의 버전 관리 기능이 포함되어야 함

● X/Open, POSIX, ANSI C, ISO 표준 등 기존 표준 중 널리 받아들여지고 있는 표준에 부합되어야 함

1.2 핵심 객체 모형(Core Object Model)

OMG가 객체기술에 대한 참조 모형으로 제시한 것이 핵심 객체모형과 CORBA(Common Object Request Broker Architecture)이다

핵심 객체모형은 OMG(Object Management Architecture) 모형에 부합하는 객체 시스템이 지원해야 할 개념들을 설명하는 형식적인 모형으로서 객체, 오퍼레이션, 비 객체타입, 인터페이스, 대체 가능성, 타입, CORBA와 핵심 객체모형과의 관계를 살펴보기로 한다.

● 객체

객체는 사람, 비행기, 부서, 상품 주문, 파일, 윈도우 매니저 등의 임의의 개체나 개념을 모형화 하는데 사용할 수 있다. 객체의 기본적인 특성은 자신만의 아이덴티티(Distinct identity)를 가지는데 있다. 아이덴티티는 객체가 가지는 속성 값이 변하더라도 항상 동일하다.

● 오퍼레이션

각 객체는 자신만의 오퍼레이션을 제공하는데 객체에 오퍼레이션을 적용하거나 또는 요청을 보내 객체의 내용을 조작하거나 객체의 정보를 얻어낸다.

각 오퍼레이션은 서명(Signature)을 가지며, 서명은 연산의 이름, 매개변수들의 집합, 그리고 반환될 결과 값들의 집합으로 이루어 지며 이때 매개변수는 해당 오퍼레이션의 인터페이스에서 선언된 매개변수를 이야기한다. 실제 오퍼레이션을 수행할 때 넘겨주는 자료는 전달인자(Argument)라고 부른다. 객체에 대한 오퍼레이션을 실제 수행시키는 작업을 요청(Request)이라고 하며, 요청의 결과로 클라이언트가 얻을 수 있는 결과는 반환 값(Return value), 부작용(Side effect), 예외상황 코드(Exception)이다.

● 비 객체타입

C++이나 CORBA에 부합하는 시스템과 같은 많은 객체 시스템에서는 객체와 객체가 아닌 것들을 명확히 구분짓고 있다. 비 객체타입의 예로는 CORBA 명세에서 정의된 Short, UShort, Ulong, Float등의 기본 값과 같은 것이다. 객체들의 집합과 비 객체들의 집합은 함께 명명 가능한 값(Denotable value)들의 집합을 나타낸다.

● 인터페이스와 대체가능성 관계(Substitutability relationship)

오퍼레이션의 서명들의 집합을 인터페이스라고 한다. 인터페이스 내에서의 오퍼레이션은 연산의 이름으로 구분되며 하나의 인터페이스는 동일한 이름을 가진 두개의 오퍼레이션을 가질 수 없다. 두개의 인터페이스는 대체가능성 관계에 의해 서로 연관되어 있을 수 있다. 만약 인터페이스 B가 나타내는 객체를 사용하고자 하는 클라이언트가, 인터페이스 A가 나타내는 객체에 대한 참조를 사용할 수 있다면 인터페이스 A는 인터페이스 B를 대체할 수 있다고 정의한다.

● 상속과 서브 타입

핵심 객체모형에서는 모든 새로운 타입은 상속성을 이용하여 기존의 타입을 기반으로 정의된다. 특히 이 모형에서는 하나의 타입에서 구성된 새로운 타입은 자동으로 원래 타입의 서브타입이 되도록 상속성이 정의된다.

● 핵심 객체모형에서의 상속규칙

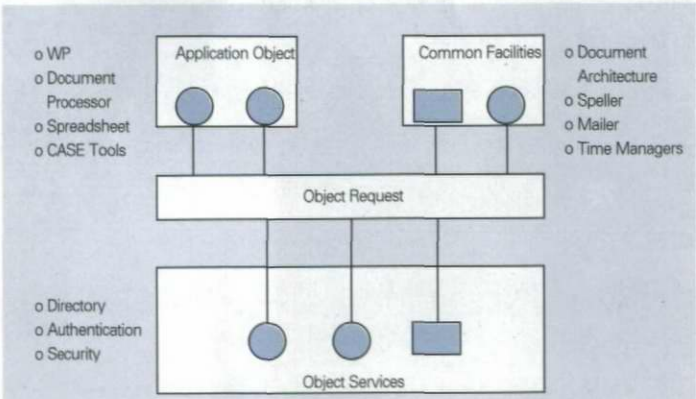
핵심 객체모형의 모든 객체는 하나의 타입에 있어서 직접적인 인스턴스가 된다. 그리고 이 객체는 자신의 타입이나 슈퍼타입의 모든 인스턴스들에 의해 대체 가능하다.

1.3 OMA(Object Management Architecture) 참조모형

OMG는 분산 객체 환경을 위해 OMA라는 참조 모형을 제시했으며 이것은 객체 중개자(Object Request Broker, ORB), 객체 서비스(Object Service), 공용 부대 시설(Common Facility), 그리고 응용 객체(Application Objects)의 네가지 구성 요소로 이루어진다. 〈그림 1〉은 OMA 참조모형을 나타낸다.

하나의 응용 프로그램이 OMA 참조모형에 부합한다는 것은 응용 프로그램내의 클래스나 그 인스턴스들이 OMA 모형에 따르는 중개자를 통해 서비스를 요청하고 이에 대한 응답을 받게 된다는 것을 의미한다. OMA 참조모형중 이질적인 환경에서 구현된 응용프로그램들 간의 상호운용성을 보장하는 등 전체 관리구조에서 핵심적인 역할을 담당하는 것이 객체중개자이다. 객체중개자는 네트웍상에 분산되어 있는 객체들 사이의 요청과 응답을 중재하는 역할을 하게된다.

객체가 위치한 컴퓨터나 서비스를 제공하는 객체가 위치한 컴퓨터에서 OMG가 제시한 표준에 맞는 객체중개자를 가지고 있다면, 접근하고자 하는 객체가 구현된 환경에 상관없이 서비스의 요청과 제공이 가능하다. 또 클라이언트 객체의 요청이 투명하게 처리된다. 클라이언트는 접근하고자 하는 객체가 네트웍상의 어디에 있는지, 어떻게 구현되어 있는지, 또



〈그림 1〉 OMA 참조모형

는 어떤 방식으로 저장되어 있는지 알 필요가 없다. OMG에서는 객체중개자의 표준으로서 제시한 것이 CORBA이다.

객체 서비스는 객체 중개자가 객체간의 통신을 중개하는데 필요한 기본적인 서비스를 의미한다. 예를 들어, 객체 A가 다른 컴퓨터상에 존재하는 객체 B를 접근하고자 할 때, 객체 중개자는 A의 요청에 대해 B가 네트웍상에 실제로 존재하는 지 그리고 어디에 존재하는지를 알아내야 한다. 이름 서비스에서는 객체의 이름 'B'에 B의 실제 위치를 매핑시켜 주는 서비스를 제공한다.

그 외에 객체 서비스의 예로서는 다음과 같은 것들이 있다.

- 관계(Relationship) 서비스
- 저장(Persistence) 서비스
- 생명주기(Lifecycle) 서비스
- 사건(Event) 서비스
- 보안(Security) 서비스
- 자료 교환(Data Exchange) 서비스
- 변경 관리(Change Management) 서비스

2. 객체중개자

2.1 개 요

가. 객체 중개자

객체 중개자는 OMG가 제안한 OMA 참조모형에서 응용프로그램의 객체간 서비스를 중개하는 역할을 하는 부분으로서 객체관리 구조 중 핵심적인 기능을 수행한다. 분산 객체 환경에서는 각 객체가 네트웍상에 분산되어 있으며 어떤 객체에 대한 오퍼레이션을 하고자 하는 경우에는 특정한 전달 매체를 통해 그 객체에게 서비스를 요청하고 원하는 오퍼레이션 결과를 돌려 받게 된다.

이때 서비스를 받고자하는 객체를 클라이언트라 하고 서비스를 하게 되는 객체의 실제 구현된 코드와 데이터를 객체구현(Object Implementation)이라고 한다. 객체 중개자는 서비스가 요청된 해당 객체를 네트웍 상에서 찾거나 객체구현에게 서비스 요청을 전달하거나 요청과 관련된 자료를 전달하는 등의 원격 객체에 대한 수행에 관련된 일체의 작업을 수행하게 된다.

클라이언트의 경우에는 객체의 어디에 위치해 있는지 또는 그 객체가 어떤 프로그래밍 언어로 구현되어 있는지와 같은 사항에 상관없이 구현이 가능해 분산 응용프로그램 개발에 많은 이점을 얻을 수 있다.

나. OMG CORBA

CORBA는 OMG에 의하여 객체관리 구조모형의 구성요소인 객체 중개자의 표준으로 제시되었다. 객체 중개자에 의해 제공되는 기본적인 서비스는 클라이언트의 객체 요청을 서버에게 전달하고 그 결과를 다시 클라이언트에게 되돌려 준다. 이론적으로 객체 분산환경의 모든 객체는 서비스를 요청하는 클라이언트가 될 수 있고 또 동시에 서비스를 제공하는 서버가 될 수 있다. 하나의 전달 매체를 통해 네트웍상의 객체들이 상호간에 의사소통을 하기 위해서 원격 프로시듀어 호출(Remote Procedure Call) 매커니즘을 사용한다.

호출자와 피호출자간에는 CORBA의 IDL(Interface Definition Language, 접속정의언어)과 CORBA가 정의한 언어 매핑(Language mapping)을 통해 해결할 수 있다. 서비스를 요청한 클라이언트와 서비스를 제공하는 서버는 모두 IDL로 자신과 다른 객체간의 가능한 인터페이스를 정의해 놓음으로써 동일한 프로토콜로 상호간에 정보를 전달할 수 있다. 이는 IDL이 객체지향 개념을 포함하고 있어서 상속과 정보은닉 등의 개념을 사용할 수 있다는 점에 기인한다.

응용프로그램에서 CORBA 표준을 따르는 객체 중개자를 통한 분산 서비스를 사용하는 작업은 동적 방식과 정적 방식의 두가지로 이루어진다. 동적 방식에서는 클라이언트와 서버와의 연결이 실행 시간에 이루어지는 반면에 정적 방식에서는 서버와의 인터페이스를 컴파일시에 결정하게 된다.

CORBA는 객체 중개자를 객체 중개자 코어(ORB Core), 객체 중개자 인터페이스(ORB Interface), 객체 어댑터(Object Adapter), 동적 수행 인터페이스(Dynamic Invocation Interface), 정적 수행 인터페이스(Static Invocation Interface), 클라이언트, 객체 구현 등의 요소로 구성하고 있다.

객체 중개자는 하나의 독립적인 구조라기 보다는 중개자가 제공하는 인터페이스에 의해 정의된다고 할 수 있다. 객체 중개자에서 가장 핵심이 되는 부분은 객체 중개자 코어로서 이는 객체의 기본적인 표현과 요구의 전달을 담당한다.

CORBA는 다양한 형태의 객체 시스템을 지원하도록 설계되었으며 이는 객체 중개자 코어의 상단에 위치한 구성요소들을 적절히 구성함으로써 다른 형태의 객체들 사이의 차이점을 극복하게 한다.

다. 클라이언트

객체의 클라이언트는 자신이 원하는 객체의 객체참조(Object reference)를 가지고 그 객체에 대해 오퍼레이션 수행을 요청하게 된다. 클라이언트는 그 객체의 인터페이스를 통한 논리적인 구조와 오퍼레이션의 수행을 통한 객체의 행위만을 경험적으로 알 수 있다.

라. 객체구현

객체구현은 객체 인스턴스에 대한 자료와 객체의 메소드들에 대한 코드를 정의함으로써 객체의 의미(Semantics)를 제공한다. 객체구현은 다른 객체를 사용할 수 있으며 객체의 행위를 구현하기 위해 다른 형태의 소프트웨어를 사용할 수 있다.

마. OMG IDL

OMG IDL은 각 객체의 인터페이스를 기술하는데 사용되는 언어로서 객체의 타입을 정의할 수 있는 방법을 제공한다. 인터페이스는 명명된 오퍼레이션과 그 오퍼레이션에 대한 매개변수들에 대한 집합으로 이루어져 있다. IDL은 어느 특정한 객체구현이 자신의 클라이언트에게 어떤 오퍼레이션이 가능하고, 또 이 연산을 수행시키기 위해서는 어떻게 수행해야 하는지를 알려주는 수단이 된다.

바. 클라이언트 스텁

비 객체지향 언어의 매핑을 위해서는 각 인터페이스 타입에 대하여 스텁에 대한 프로그래밍 인터페이스가 존재한다. 일반적으로 스텁은 객체에 대하여 OMG IDL에 의해 정의된 오퍼레이션을 수행할 수 있는 방법을 제공한다.

사. 동적 수행 인터페이스

동적 수행 인터페이스는 객체수행(Object Invocation)의 동적인 생성을 도와준다. 클라이언트는 동적으로 수행하고자 하는 객체를 지정하고 어떤 오퍼레이션을 수행할 것인지를 결정한 후, 이 오퍼레이션에 대한 매개변수 값을 결정한다. 이 작업은 동적 수행 인터페이스가 제공하는 일련의 오퍼레이션을 실행함으로써 가능하다.

아. 객체 어댑터

객체 어댑터는 객체구현이 객체 중개자의 서비스를 받는 주요한 통로 역할을 하게 된다. 객체 중개자가 기본 어댑터를 통해 제공하는 서비스는 다음과 같다.

- 객체 참조의 생성과 해석
- 메소드 수행
- 보안 매커니즘 제공
- 객체와 객체구현의 실행(Activation)과 실행중지(Deactivation)
- 객체 참조와 구현과의 매핑
- 객체 구현의 등록

자. 객체 중개자 인터페이스

객체 중개자 인터페이스는 객체의 인터페이스와 객체 어댑터를 통한 인터페이스와는 달리 객체 중개자가 직접 제공하는 오퍼레이션들의 집합을 의미한다. 주로 객체 어댑터나 스텁, 스켈레톤, 또는 동적 수행 인터페이스를 통해 원하는 서비스를 제공하기 때문에 직접 클라이언트나 객체구현에게 제공하는 오퍼레이션은 많지 않다.

차. 인터페이스 저장소(Interface repository)

인터페이스 저장소는 IDL에 의한 객체 정보를 제공함으로써 클라이언트가 실행시간에 원하는 객체의 인터페이스를 알아내고 적당한 서비스 요청을 할 수 있도록 도와준다.

카. 구현 저장소(Implementation repository)

구현 저장소는 객체 중개자가 네트웍 상에서의 객체의 위치를 알아내고 객체의 구현을 실행시키는데 필요한 정보를 가지고 있다. 일반적으로 구현을 객체 중개자에 등록한다든지 등록된 구현을 찾거나 실행시키는 작업은 구현 저장소에 대한 오퍼레이션으로 이루어진다.

타. 객체 중개자의 구현

CORBA에 부합하는 객체 중개자는 실제로는 다음과 같은

여러 형태로 구현될 수 있다.

첫째는, 클라이언트와 서버에 상주하는 객체 중개자의 형태이다. 이 경우 적당한 통신 매커니즘이 설정되어 있다면 객체 중개자는 클라이언트와 서버의 코드내에 포함시킬 수 있다.

둘째는, 객체 중개자를 서버 형태로 구성하는 것이다. 이 때 클라이언트와 서버는 객체 중개자 서버에게 요청을 함으로써 클라이언트와 서버간의 의사 소통을 수행하게 된다.

셋째는, 객체 중개자는 운영체제 수준에서 구현하는 방법이 있을 수 있고 마지막으로 객체 중개자를 라이브러리 형태로 구현할 수 있다.

2.2 상호 운용성

객체 중개자간의 상호운용성을 보장하기 위하여 OMG에서는 세가지의 표준을 제시했다. 객체 중개자 구조에서의 표준, 객체 중개자간 브리지(inter ORB bridge)에서의 표준, 일반/인터넷상의 객체 중개자간 프로토콜(GIOP/IIOP) 표준이다.

가. 객체 중개자 상호운용 구조

서로 다른 업체에서 생산된 객체 중개자들간의 상호운용성을 보장하기 위한 기본적인 개념을 제공한다. 이 구조는 객체 중개자 도메인간의 지급(immediate)브리징과 지연(mediate)브리징의 개념을 소개한다.

나. 객체 중개자간 브리지

객체 중개자 상호운용 구조에서는 객체 중개자의 정보를 몇가지 도메인으로 구분한다. 예를 들어, 객체 참조 도메인, 타입 도메인, 보안 도메인, 트랜잭션 도메인 등이 그것이다. 동일한 도메인에 있는 객체 중개자들끼리는 직접적으로 통신이 가능하다. 그러나 만약 다른 도메인에 있는 객체 중개자를 접근하기 위해서는 브리지를 거쳐야만 한다. 브리지의 역할은 객체 중개자간의 서로 다른 표현 양식을 서로에 맞게 매핑시켜 주는데 있다.

다. 일반 중개자간 프로토콜(GIOP)

GIOP는 하위 수준에서의 자료 표현 방법과 객체 중개자간에 통신을 위한 메시지 양식을 명시한다. GIOP는 연결 중심 프로토콜하에서 객체 중개자 사이의 작업을 위해 만들어졌다.

라. 인터넷 객체 중개자간 프로토콜(IIOP)

IIOP는 GIOP 메시지들이 TCP/IP연결을 통해 교환되는 방식을 명시하는 프로토콜로서 IP도메인에 있는 임의의 객체 중개자 사이에 원하는 통신이 이루어지도록 도와준다.

마. 환경 종속 객체 중개자간 프로토콜(ESIOP)

ESIOP(Environment-Specific Inter ORB Protocol)은 상호운용 구조의 확장을 위한 프로토콜들을 의미한다. 특정 환경을 위해 최적화할 필요가 생길 경우 이에 해당하는 ESIOP를 개발함으로써 기능을 확장해 나가게 된다.

3. 객체 서비스

1994년 OMG는 객체관리구조를 지원하기 위한 객체 서비스 표준을 채택하였다. 공동 객체 서비스 명세(Common Object Service Specification, COSS)라 불리는 이 표준안은 객체를 실현하고 관리하기 위한 기본적인 함수를 제공하는 서비스의 집합이다.

COSS에는 현재 이름서비스, 사건서비스, 생명주기서비스가 표준으로 명시되어 있다. 객체 서비스는 다음과 같은 서비스를 제공한다.

- 객체의 생성에서부터 소멸에 이르기까지의 생명주기를 표준화한다.
- 객체를 생성하고, 객체를 통제하고, 객체의 이동을 관장하고, 객체사이의 관계를 관리하기 위한 인터페이스를 제공한다.
- 응용의 일관성과 생산성 향상을 위한 서비스를 제공한다.
- 클래스 관리, 개체 관리, 저장, 순결성, 보안, 질의, 버전 등을 표준화한다.
- 객체에 관한 기본적인 함수를 제공함으로써 OMA의 공용 부대시설과 응용객체를 구축하기 위한 건축재료를 제공한다.

3.1 이름 서비스 명세

이름 서비스 명세(Naming Service Specification)는 객체에 이름을 부여하는 것에 대한 COSS에 규정된 OMG 표준이다. 객체와 이름을 연관시키는 '이름결합'들을 포함하는 집합

자체도 객체로 취급된다. 이 객체는 응용프로그램에 따라 달라질 수 있으며, 기존의 이름 시스템을 사용할 수도 있다.

이름 표준은 이름에 관한 문법과 표현의 독자성을 보장하며, 이름의 의미와 해석은 규정하고 있지 않다. 이는 상위 응용소프트웨어가 결정해야 할 사항으로 간주한다. 표준에 부응하는 이름 시스템은 다목적용 이름 서비스를 사용할 수도 있고 상이한 목적 - 예를 들면 전자 우편, 파일 등 - 을 위하여 상이한 이름 서비스를 사용할 수도 있다.

이름 서비스는 분산, 이기종간의 구현, 관리를 위한 이름공간 연합을 그래프 형태로 지원하며 이름 서비스 명세의 설계원칙은 다음과 같다.

- 이름 자체에는 아무런 의미도 부여하지 않고 해석하지도 않는다.
- 이름과 이름 문맥의 이기종간 분산 구현을 지원한다.
- 이름은 문자 스트링이 아니라 구조체(struct)이다.
- 이름 서비스의 사용자는 분산 환경하에서 이름 서버의 물리적 위치를 알 필요가 없으며, 어느 서버가 이름의 어떤 부분을 해석하는지 또는 어떤 식으로 서버가 구현되었는가를 알 필요가 없다.

3.2 사건 서비스 명세

사건 서비스는 객체에 대하여 공급자와 소비자라는 두가지 역할을 정의한다. 공급자는 사건 데이터를 생성하고 공급자는 사건 데이터를 처리한다. 사건 데이터는 공급자와 소비자간에 CORBA 표준 요구에 의해 전달된다. 공급자와 소비자간의 사건 전달 방법은 push모형과 pull down의 두가지가 존재한다. push모형은 사건의 공급자가 사건의 소비자에게 사건 데이터를 전달하도록 하고 pull모형은 사건의 소비자가 사건의 공급자에게 사건 데이터를 요구하도록 한다.

push모형은 사건의 공급자가 우선권을 갖고 pull모형은 사건의 소비자가 우선권을 갖는다. 사건 채널은 다중 공급자와 다중 소비자간에 비동기적으로 통신을 할 수 있도록 해준다.

사건 채널은 사건의 공급자이면서 소비자이자 CORBA의 표준 객체이며, 사건채널과의 통신은 CORBA 표준 요구를 통해 이루어진다.

다음은 사건 서비스 명세의 설계 원칙이다.

- 사건 서비스는 분산 환경에서 동작된다.
- 사건 서비스는 다중 공급자와 다중 소비자를 지원한다.
- 소비자는 시스템의 성능이나 여러 상황에 따라 사건을 요청할 수도 있고 사건을 통보 받을 수도 있다.
- 소비자와 공급자는 CORBA 표준의 IDL 인터페이스를 지원한다. 소비자와 공급자의 인터페이스를 위해 CORBA를 확장할 필요는 없다.
- 공급자는 모든 소비자와 사건 데이터를 통신하기 위하여 즉시 단 하나의 CORBA 표준 요청을 할 수 있다.
- 공급자는 사건을 생성할 때 사건의 소비자를 확인할 필요가 없다. 소비자도 역시 사건의 공급자를 확인하지 않고 사건을 전달받을 수 있다.
- 사건 서비스의 인터페이스는 상이한 수준의 신뢰성의 확보, 향후 인터페이스 확장 가능, 부가적인 기능 제공과 같은 서비스의 질을 다양화할 수 있다.
- 사건 서비스 인터페이스는 예를 들면, 스레드를 지원하거나 지원하지 않는 운영환경처럼 상이한 운영 환경에서도 구현할 수 있고 사용될 수 있다.

3.3 생명주기 서비스 명세

생명주기 서비스 명세는 객체의 생성과 소멸, 복사, 이동에 관한 서비스를 정의한다. CORBA는 분산 객체를 지원하는 환경이기 때문에 생명주기 서비스는 클라이언트가 서로 다른 위치에서 객체에 대한 생명주기 오퍼레이션을 수행하는 컨벤션을 정의한다.

생명주기 서비스 명세의 설계원칙은 다음과 같다.

- 생성에 대한 클라이언트의 모형은 'Factory'라는 객체로 정의되는데 Factory 검색기에 등록된 Factory 객체는 Factory 검색기가 관장하는 영역에서 객체 구현을 가지고 있다. 그러므로 Factory 검색기는 클라이언트가 객체 구현에 대한 위치 정보를 질의할 수 있도록 한다.
- 객체 구현은 위치정보에 관계된 Factory를 발견하는 지식을 포함할 수 있다. 객체 구현은 일반적으로 위치에 대한 정보는 포함하지 않는다.
- 복사나 이동과 같은 생명주기 오퍼레이션에 대한 원하는 결과는 목표 객체와 다른 객체들간의 관계성에 달려 있다.
- 생명주기 서비스는 어떤 특별한 저장 모형에 의존하지 않으며 이질형 분산 환경에 적합하도록 한다.