□ 기술해설 □

TINA-C DPE 플랫폼

고려대학교 박성원* · 선경섭* · 안순신**

목 차

1. 서 론
2. 분산처리 플랫폼
3. TINA-C DPE 구조
3.1 TINA-C DPE 구조의 개념 및 정의
3.2 TINA-C DPE 구조의 목적
3.3 TINA-C DPE 플랫폼에 대한 요구사항
3.4 TINA-C DPE의 기본구조
3.5 TINA-C DPE 구조의 관점에서 본 OMG와의 비교
4. TINA-C DPE 모델
4.1 개요
4.2 배치 개념
5. 결 론

1. 서 론

최근 정보통신 분야와 컴퓨터 기술 분야에서의 급속한 발전으로 인해 이를 이용한 서비스의 질적, 양적인 향상에 대한 사용자의 요구가 날로 증대되고 있으며, 이러한 사용자의 요구를 충족시켜 주기 위해서는 정보의 위치, 형태, 양, 시간적 제한에 관계없이 다양한 서비스를 제공할 수 있는 정보망 서비스가 제공되어야 한다.

사용자에게 제공되는 고도의 정보망 서비스는 분산 애플리케이션을 통해서 실현되는데 분산 애플리케이션의 공통적인 요구사항으로는 성능, 안전성, 자원의 공유, 확장성, 효율성, 이질성의 극복 등을 들 수 있다. 이와 같은 분산 애플리케이션의 요구사항을 충족시켜 주는 한편 다양한 하드웨어, 소프트웨어, 통신망, 프로토콜을 포함하는 이질적인 형태의 분산 환경에서 분산 애플리케이션의 개발 및 사용을 용이케 하기 위해서는 정보망 서비스를 위한 하부구조로 표준 분산 플랫폼이 개발되어야 하며 이러한 분산처리 플랫폼은 애플리케이션의 개발, 사용 및 관리를 용이하게 하기 위한 유연한 망 구조 개념과 이질적인 연산 환경의 표준화된 관점을 제공하는 개방형 분산처리 개념을 통합적으로 지원해야 한다.

이러한 중요성에 몇몇 회사가 인식을 같이하여 1990년 TINA(Telecommunications Information Networking Architecture) 워크샵이 최초로 개최되었고 이에 힘입어 1992년 봄에 TINA-C(Consortium)가 미국의 Bellcore, 영국의 BT, 일본의 NTT의 제창 아래 1992년 10월에 정식으로 발족하게 되었으며 현재는 Bellcore, BT, NTT 외에도 한국통신, AT&T, NDD, 프랑스 텔레콤, 네델란드 텔레콤, 독일 텔레콤, 노르웨이 텔레콤, DEC, NEC, OKI, 알카텔, 노키아 등 세계 굴지의 통신 업체들이 회원으로 참여하고 있다. 최근에는 국내의 민간 통신 업체들도 TINA-C에 참여하기 시작했으며 TINA 워크샵은 1990년 개최 이래 매년 개최되어 왔고 참고로 1996년에는 독일에서 개최될 예정이다.

TINA-C에서는 매년 12월경 그 해의 연구결과를 출간하여 회원에게 배포하고 있으며 TINA-C에 참여한 기관들에 의해서 프로토타입의 구현도 지속적으로 이루어지고 있다.

*학생회원
**종신회원

TINA-C와 호환인 애플리케이션들은 상호 연동이 가능한데, TINA-C에서는 일부 사양도 제정되기는 하지만 TINA-C의 궁극적인 목표는 특정한 서비스나 동작의 사양을 정하는 것이 아니라 모든 소프트웨어 애플리케이션이 명시되고 설계될 수 있는 틀(framework)을 정의하는 것이다.

앞서서 언급한 분산처리 플랫폼 구축 시 애플리케이션 간과 애플리케이션 내의 상호작용을 위한 하부구조는 TINA-C에서 DPE(Distributed Processing Environment)라 불리는 구조에 의해 제공된다.

DPE는 TINA-C의 고유한 개념만은 아니며 분산처리 환경과 관련되어 추진되고 있는 유사 연구활동을 살펴보면, 분산 객체 관리 환경에 관한 연구사례로 OMG/CORBA, Object-Broker, Orbix 등이 있고 또한 RM-ODP, OSF/DCE, OSF/DME, ANSAware, Bellcore의 INA 등을 생각할 수 있다.

TINA-C의 DPE 구조는 Bellcore의 INA에 있는 DPE 개념에서 출발 하였으나 연구가 진행되면서 그 세부적인 내용은 상당 부분 달라졌으며 현재 TINA-C 내에서는 주로 OMG와 RM-ODP의 모델을 참조하여 그 연구가 진행되고 있는 것으로 보인다.

본 고에서는 TINA-C의 전체 구조 중 DPE 플랫폼에 관한 사항을 TINA-C에서 1994년 12월에 발간된 자료를 기준으로 하여 주로 공학 모델링의 개념에서 기술 하였다.

2. 분산처리 플랫폼

분산 애플리케이션이란 여러 부분으로 구성된 일종의 소프트웨어로서 각 부분은 같은 망 내의 다른 시스템에서 배치, 실행이 가능하며 또한 서로 통신하고 협력할 수 있다. 분산 애플리케이션은 그 기능을 제공함에 있어 다른 시스템에 있는 애플리케이션과 상호 작용하며 그 기능들을 이용한다. 분산 애플리케이션을 상호 작용하는 연산단위의 분산 객체로 모델링하고 애플리케이션 객체간의 이식성과 상호 운용성을 보장하기 위해서는 애플리케이션 및 이를 지원하는 플랫폼 전반에 걸쳐 분산 연산 기술과 객체 지향 기술의 적용이 필요하다.

분산 환경을 구성하는 세 가지 요소, 즉 애플리케이션, 분산처리 플랫폼, 원시환경 중에 분산처리 플랫폼은 분산 애플리케이션에서 필수적으로 요구되는 분산 투명성(접근 투명성, 위치 투명성, 장애 투명성, 이동 투명성, 재배치 투명성, 복제 투명성, 트랜잭션 투명성, 영속 투명성 등)을 제공함과 동시에 실제적인 연산 환경과 통신 환경 그리고 특정 실현 기술에 대해 독립적이고 표준화된 처리 환경을 제공하는 하부구조를 지칭하며 애플리케이션과 하부 원시환경과의 인터페이스 역할을 수행한다.

서론에서도 언급한 바와 같이 분산처리 플랫폼 구축 시 애플리케이션 간과 애플리케이션 내의 상호작용을 위한 하부구조는 DPE라 불리는 구조에 의해 제공된다. 분산 애플리케이션은 망연결의 설정과 해제를 포함하는 애플리케이션 내의 접속과 해제, 동작의 요구와 응답을 위한 메세지 구성, 데이터 표현 방법에 있어서의 상이성과 같은 분산처리의 복잡성이 따르기 마련인데 DPE는 애플리케이션이 이러한 복잡성을 극복하도록 해 준다. 분산 애플리케이션의 실행 환경으로 간주되는 DPE는 정보망에서 분산 처리 플랫폼 구축 시 분산 애플리케이션 설계의 복잡성을 제거해 주는 메커니즘 및 도구를 제공함으로서 분산처리 환경 및 분산 관리에 있어 중추적 역할을 담당하고 있다.

하부구조로서의 DPE가 분산처리의 복잡성과 세부사항들을 애플리케이션으로부터 보이지 않도록 하기 위해서는 애플리케이션에서 요구되는 여러 종류의 분산 투명성을 제공해야 한다. DPE는 분산 투명성을 제공하기 위하여 시스템을 포함한 망의 자원을 관리할 수 있는 능력이 있어야 하고 연산 구조의 관점에서 본다면 분산처리 플랫폼을 위한 시스템 관리 기술은 앞서 언급한 바대로 분산 연산 기술과 객체 지향 기술에 바탕을 두어야 하며, 정보망에서 분산처리 플랫폼 구축 시 DPE 자원들을 관리할 수 있는 분산객체 모형의 정립과 객체 지향 프로토타이핑 기술의 확보는 필수적이라 할 수 있다.

통상적인 분산 환경은 크게 애플리케이션과 애플리케이션의 분산 실행 환경인 DPE로 구

성되어 있다고 간주되며, 분산 애플리케이션의 실행에 있어 DPE는 애플리케이션의 배치, 실행 및 상호작용을 지원하는 서비스를 제공한다. 이때 애플리케이션은 상호작용하는 연산 객체 단위로 설계되고 구현되며 DPE는 연산 객체의 공학적(Engineering) 구성을 담당하는 플랫폼으로 사용된다. 분산처리 플랫폼에 대한 애플리케이션의 요구사항은 DPE 구조로 표현되며, 이 구조의 구현이 DPE 플랫폼이라 할 수 있다.

3. TINA-C DPE 구조

본 절에서는 공학 모델링 개념의 목적과 DPE 구조를 서술한다. 또한 TINA-C DPE 구조와 사양의 개념을 정의하며 제시된 구조의 목적과 요구사항을 개괄적으로 설명한다. 본 절에서 사용된 그래픽 표현의 용례는 그림 1과 같다.

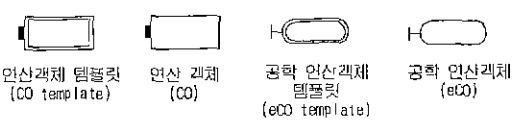


그림 1 그래픽 표현의 용례

3.1 TINA-C DPE 구조의 개념 및 정의

TINA-C 공학 모델링 개념은 TINA-C 연산 객체의 실행을 가능케 하는 추상적 하부구조의 구성을 기술하는 틀(framework)이다. 연산 객체로 구성된 TINA-C 애플리케이션은 그것의 배치, 실행, 상호작용을 지원하는 추상적 하부구조에 의존하며 이러한 기능을 제공하는 하부구조를 TINA-C DPE라 한다(그림 2 참조).

TINA-C 공학 모델링 개념의 목적은 TINA-C 애플리케이션을 지원하기 위한 하부구조 요구사항의 기술에 있으며 이러한 요구사항은 DPE 구조와 사양에서 공통적으로 표현된다.

TINA-C DPE 구조는 DPE 모델로 이용된다. DPE 구조는 애플리케이션을 지원하기 위한 개념, 모델, 메커니즘을 다루고 있으며, 구조의 서술이 공학 모델링 개념의 주 목적이다.

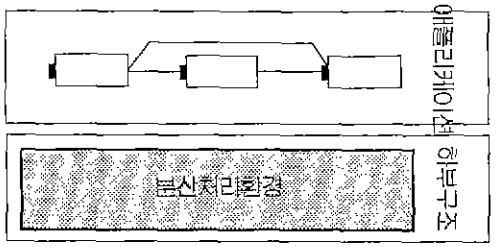


그림 2 추상적 하부구조

에플리케이션의 실행 환경에서 예상되는 이질성을 감추기 위해 DPE 구조는 하부 기술에 독립적이어야 하며 애플리케이션 요소를 지원하는 DPE의 실현은 이 구조를 따른다.

DPE 구조에 설명되어 있는 하부구조에 대한 요구사항의 일부는 서비스, 인터페이스 및 동작 등의 항목으로 표시된다. 그러한 요구사항들은 TINA-C 연산 구조의 개념(정보 모델링 및 연산 모델링)이 차례대로 적용되며 그에 따라 연산 하부구조의 기능적 요구사항이 정보 및 연산 모델로 표현된다.

TINA-C DPE 사양은 DPE에서 제공되는 인터페이스와 모델을 기술하는 구조의 정형적 부분으로 정보 모델링 개념과 연산 모델링 개념을 사용한다. 기타 TINA-C 사양과의 일관성을 위해 인터페이스는 하부 모델에 대한 명세와 더불어 TINA-C ODL(Object Description Language) 또는 Q-GDMO로 서술된다.

DPE 노드는 DPE 구조를 지원하는 자원의 운용 단위를 의미하며, 노드에서 구조를 지원하는 부분이 플랫폼이다. 플랫폼을 지원하는 연산 자원을 원시 연산 통신 환경(NCCE : Native Computing and Communication Environment)이라 부르며, NCCE 자체가 지역적

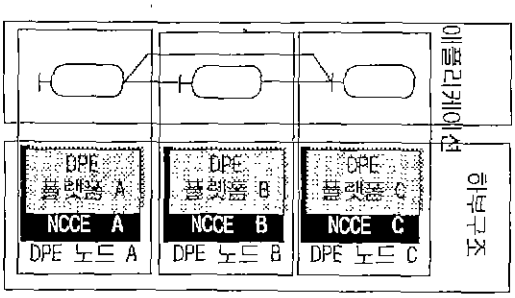


그림 3 물리적 하부구조

으로 분산되었더라도 하나의 DPE 노드에 관련된 플랫폼은 유일하게 하나만 존재한다.

DPE 플랫폼은 객체가 실행되는 NCCE의 이질성을 감추어 주는 기능을 제공한다(그림 3 참조).

3.2 TINA-C DPE 구조의 목적

애플리케이션의 실행에 필요한 추상적 하부구조를 기술하기 위해 TINA-C DPE 구조는 다음과 같은 목적을 갖는다.

- 하부구조에서 필요한 기능을 판별하고 서술한다(DPE에 대한 기능적 요구사항). 요구되는 기능들은 애플리케이션이 사용하는 개념, 모델, 메커니즘 및 인터페이스로 표현된다.
- 하부구조에서 필요한 속성을 판별하고 서술한다(비기능적 요구사항). 속성 서술을 위한 형식적 정의, 모델과 속성을 감시하는 메커니즘을 통해 속성이 지원된다.
- 가능한 경우 최대한으로 기존 표준과 결과를 활용한다
- 하부 기술에 무관한 방식으로 연산 하부구조의 요구사항을 서술한다

DPE 구조의 주요 사용자는 애플리케이션 설계자와 개발자 그룹과 DPE 제공자 그룹으로 양분된다. 애플리케이션 설계자와 개발자에게 DPE 구조는 연산 하부구조에서 제공되는 기능과 속성을 의미한다. 따라서 하부구조상에서 애플리케이션을 실행하기 위한 구성 및 배치 방법을 구조에서 서술한다. DPE 제공자 입장에서 DPE 구조는 애플리케이션에 제공되어야 하는 기능과 속성을 의미한다. 즉, 구조는 플랫폼 개발을 위한 외형적 요구사항으로 사용된다.

3.3 TINA-C DPE 플랫폼에 대한 요구사항

TINA-C DPE 구조는 유일하게 하나만 존재하지만 구조를 지원하는 다수의 플랫폼이 가능하며 서로 다른 NCCE를 사용할 수 있다. TINA-C DPE 플랫폼에 대한 요구사항은 다음과 같다.

- 같은 노드 또는 이질적인 NCCE상의 서로 다른 노드에 관계없이 객체의 위치에 무관하게 객체간 상호작용이 가능해야 한다. 장애허용성, 가용성, 신뢰성 등의 지원 기능이 이 부분에 포함된다.
- 애플리케이션에게 서로 다른 NCCE상에서 실행된다는 사실을 감추어야 한다.

공학적 레벨에서 애플리케이션 지원을 위한 두 가지 기본적 요구사항은 설계 이식성과 상호운용성이다.

애플리케이션의 설계 이식성이란 하부구조에 대한 기본적 가정이 모든 플랫폼에서 동일하게 지원된다는 보장하에 애플리케이션을 설계한다는 것이다.

하부구조에 관한 기본적 가정은 구조 내에서 다음에 열거된 내용을 서술하는 개념, 모델, 메커니즘 및 인터페이스의 형태로 표현된다.

- TINA-C 연산 모델링 개념의 지원
- 연산 애플리케이션 요소 또는 객체를 하부구조에서 실행되는 단위로 배치하는 방법
- 연산 객체간 상호작용을 위한 연산 바인딩을 공학 레벨로 대응하는 방법
- 연산 객체간 상호작용에서 분산 투명성을 제공하는 방법
- TINA-C 객체의 라이프 사이클 관리
- 연산 하부구조와 망 하부구조의 속성과 QOS 파라메터

TINA-C 구조에 따라 애플리케이션을 설계할 때 애플리케이션 개발자는 객체가 실행되는 NCCE에 무관하게 하부구조의 기본적 가정을 이용한다. 이러한 독립성은 동일한 DPE가 지원되는 임의의 NCCE상에서 TINA-C 객체 설계의 재사용을 가능케 한다.

서로 다른 DPE 플랫폼에서 운용되는 애플리케이션 간의 정적 호환성과 실행 호환성이 상호운용성으로 정의된다. 애플리케이션이 운용되는 시스템이 동일한 NCCE를 지원하는지의 여부에 관계없이 애플리케이션 간 상호작용이 일어나려면 상호운용성의 관점에서 추가적인 제약조건이 발생한다. 상호운용성에 관련된 객체의 요구사항은 다음과 같다.

- 원격 객체의 인터페이스에 대한 동작 호출이 가능해야 한다. 이 조건은 상호작용에 필요한 여러 가지 분산 투명성이 하부

구조에서 제공됨을 의미하며 관련된 여러 플랫폼간의 협조를 뜻한다. 이 동작을 위해서는 다른 노드에 존재하는 객체의 지역화를 가능케 하는 서비스 지원 기능이 필요하다.

- 원격 객체의 라이프 사이클(배치, 생성, 활성화 등)을 관리할 수 있어야 한다. 그러므로 해당되는 DPE 노드들에서 객체 라이프 사이클을 위한 공통적 모델과 개념이 지원되어야 하고 각각의 플랫폼이 그러한 개념의 조작이 가능한 인터페이스를 상대 플랫폼에게 제공 하여야 한다.
- 원격 객체에 대한 트랜잭션 형태의 통신을 관리할 수 있어야 한다.
- 다른 DPE 플랫폼에서 제공되는 원격 서비스에 대한 접근이 가능해야 한다.

이러한 요구사항들은 기술적 선택에도 관련되므로 상호운용성에 관한 모든 기술적 사항이 해결되지는 않을 것이다. DPE 구조에서는 상호운용성을 제공하기 위해 각 플랫폼에서 요구되는 지원 기능만을 기술한다.

3.4 TINA-C DPE의 기본구조

TINA-C DPE 구조의 목적과 요구사항에 따라 연산 하부 구조에 대한 세 종류의 요구사항이 도출된다. 하부 구조는 1) 연산 모델링 개념, 2) 공통 서비스, 3) 비기능적 요구사항에 대한 기능을 지원해야 한다.

1) 연산 모델링 개념에 대한 지원

연산 모델링 개념은 애플리케이션의 연산 사양에 대한 기본구조를 제공한다. 그러한 기본구조에서 필요한 개념, 메커니즘, 투명성 등이 하부구조에서 지원되어야 하며 DPE 구조에서 설명된다.

- 객체 상호작용 : RM-ODP에서 정의된 여러 가지의 투명성이 애플리케이션의 연산 사양에서 사용된다(접근, 위치, 이주, 연합, 장애, 복제 투명성 등). 따라서 이러한 투명성에 대한 지원이 하부구조에서 제공되며 DPE 구조의 분산 투명성으로 설명된다. 애플리케이션 개발자에게 균일한 상호작용 모델을 제공하려면 이밖에도 객체 상호작용 지원을 위한 개념과 메커니즘이 정의되어야 한다. 통신 모델은 채널, 스터브, 바인더, 프로토콜 어댑터 등의 개념을 정의한다.
- 객체 라이프 사이클 : 객체 라이프 사이클(객체 배치 및 철거, 객체 생성 및 제거, 객체 활성화 및 비활성화 등)에 관련된 기능을 어느 플랫폼에서나 공통적으로 지원하려면 객체 라이프 사이클 관리에 대한 개념, 모델, 메커니즘에 대한 정의가 필요하다.

2) 공통 서비스에 대한 지원

하부구조에서 제공되어야 하는 일련의 서비스를 공통 서비스(DPE 서비스라고도 함)라고 한다. 공통 서비스는 플랫폼을 지원하는 NCCE에 무관하게 기술되는 총괄적 서비스로서 아래에 열거된 서비스들이 대표적이다.

- 트레이딩 서비스 : 트레이딩 서비스는 객체간 실행 중 바인딩을 지원한다. 트레이딩 서비스는 객체가 동적으로 변하는 분산 환경에서 특정 서비스를 이용하고자 하는 객체가 그 서비스를 제공하는 객체에 관한 사전지식이 없더라도 가장 적절한 서비스 제공 객체를 찾아 이용할 수 있도록 한다. 트레이딩 인터페이스를 제공하는 트레이더는 특정 서비스에 관한 정보를 서비스 오퍼 단위로 관리한다. 서비스 오퍼에는 서비스의 종류 및 그 서비스를 제공하는 인터페이스 레퍼런스, 그 서비스의 QoS 속성 등이 포함된다. 트레이더에게 서비스 오퍼를 제공하는 객체를 엑스포터라 하며 트레이더에게 요구하여 원하는 서비스 오퍼를 제공받는 객체를 임포터라고 한다.
- 객체 라이프 사이클 서비스 : 객체 라이프 사이클 서비스는 객체의 생성(creation), 제거(deletion), 활성화(activation), 비활성화(deactivation), 이동(move), 복사(copy)를 제공한다. 연산의 관점에서 객체의 생성은 객체 템플릿을 인스탄스화함으로써 이룰 수 있다. 객체의 생성을 위하여 추가로 필요한 정보는 관련된 공학 템플릿에 보관된다. 한 노드의 객체 라이프 사이클 서비스는 다른 노드에서도 접

근기능 하여야 한다.

- 통지(notification) 서비스 : 통지 서비스는 사전 지식이 없는 객체에 이벤트를 보내거나 그러한 객체로부터 이벤트의 발생을 보고 받을 수 있도록 하는 DPE 서비스이다. 일반적으로 이벤트를 발생시키기 위해서는 우선 통지 서비스와 상호 작용하여 어떠한 종류의 통지를 할 것인지 알려 주어야 한다. 통지 서비스를 제공하는 객체는 이러한 요구에 대한 응답으로 통지를 담당할 인터페이스에 대한 객체 레퍼런스를 리턴한다. 객체는 이벤트를 발생시킬 때 리턴받은 객체 레퍼런스를 사용한다.
- 트랜잭션 서비스 : 트랜잭션 서비스는 흔히 ACID 속성이라고 칭해지는 원자성(atomicity), 일관성(consistency), 격리성(isolation), 지속성(durability)을 만족시키는 객체간의 트랜잭션 통신을 지원한다. 트랜잭션 서비스에서 지원되는 기능은 트랜잭션 관리 기능, 동시 접근 관리 기능, 교착 관리 기능, 로그 관리 기능 등이 있다.
- 레포지토리 서비스 : 레포지토리 서비스에는 사양 레포지토리 서비스와 구현 레포지토리 서비스가 있다. 사양 레포지토리 서비스에서는 객체 및 인터페이스의 템플릿 타입과 그 관계에 관한 정보를 저장하고 이용할 수 있는 서비스를 제공한다. 또한 사양 레포지토리에서는 트레이딩이나 바인딩을 위하여 인터페이스 타입 매칭 서비스를 지원한다. 구현 레포지토리에서는 객체 구현에 관한 정보를 저장하고 이용할 수 있는 서비스를 제공한다. 객체 구현은 객체의 실행 코드와 구현에 관한 정보, 즉 운영체제, 요구되는 메모리 및 저장소의 크기, 위치 등에 관한 정보로 구성된다. 레포지토리 서비스를 통해 서로 다른 레포지토리 영역간에도 레포지토리 내용의 공유가 가능하다. 레포지토리 서비스는 인터페이스 템플릿, 객체 템플릿, 관계, 객체 구현 등을 추가, 갱신, 삭제, 복원하는 기능을 제공하며 또한 레포지토리의 내용을 검색하는 기능도 제공한다.
- 구성 서비스 : 구성 서비스는 객체 구성에 관한 정보, 즉 객체의 위치, 활성화 상태, 결합 상태 등을 제공한다. 구성 서비스를 제공하는 구성 관리자는 DPE 내의 여러 가지 시스템 관리 동작을 조정한다.
- 설치 서비스 : 설치 서비스는 한 노드에서 객체의 설치와 삭제에 관한 서비스를 제공한다. 설치 서비스를 제공하는 설치 서버는 구현된 객체를 활성화 시키는 기능뿐만 아니라 객체의 상태를 감시하거나 객체를 없애는 기능도 포함한다.
- 보안 서비스 : 보안 서비스는 TINA-C 에 플리케이션의 보안 요구사항을 지원한다.
- 성능 감시 서비스 : 성능 감시 서비스는 성능을 측정하기 위해 특정 망 자원 활동의 성능을 접근하고 제어하는 기능을 제공한다.

TINA-C 연산 구조는 모델링 개념(정보, 연산, 공학)을 애플리케이션의 설계와 배치에 적용한다. 그 중에서도 연산 모델은 애플리케이션을 서로간에 상호작용하는 연산객체로 표현한다. 애플리케이션과 마찬가지로 공통 서비스도 연산 객체 형태로 기술될 수 있으며 공통 서비스 요청은 보통의 애플리케이션처럼 연산 인터페이스를 통해 호출된다(그림 4 참조).

애플리케이션 개발자에게 공통 서비스를 적절하게 사용하기 위한 정보를 주려면 공통 서비스의 정보 사양 또한 제시되어야 한다. DPE 사양은 공통 서비스의 연산 사양과 정보 사양을 제공한다.

다수의 공통 서비스가 사양에 기술되어 있지만 어느 플랫폼에서나 모든 서비스를 항상 제

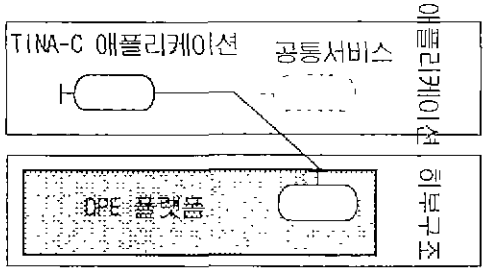


그림 4 공통 서비스 객체 표현

공할 필요는 없다. DPE 프로파일 개념은 기본적 서비스와 선택적 서비스를 구분한다.

3) 비기능적 요구사항에 대한 지원

연산 모델 및 공통 서비스에 대한 하부구조의 요구사항 외에도 별개의 요구사항이 존재한다. 앞의 사항은 기능성(하부구조에서 필요한 일련의 기능과 메커니즘)으로 표현되지만 후자는 비기능적 요구사항으로 정리된다.

비기능적 요구사항은 하부구조에서 아래와 같은 비기능적 지원을 필요로 한다.

- (실시간) 성능 : 애플리케이션이 갖는 서로 다른 형태의 타이밍 제약조건과 성능은 관련이 있다.
- 장애 허용도 : 하부구조에 장애가 발생해도 합리적 방법으로 수행을 지속하는 능력을 말한다.
- 가용도 : 서비스 사용자가 서비스를 사용코자 할 때의 사용 가능성을 최대화한다.
- 축척도(scalability) : 프로세서 구성, 망 규모, 트레픽과 서비스 패턴의 변화에 대응하는 구조의 적응도를 말한다.
- 보안 : 자원과 정보에 대한 손상 위험을 최소화한다.

비기능적 요구사항을 실현 기술에 독립적으로 기술하려면 정형적 정의와 모델이 필요하다. 정형적 접근은 개발자에게 애플리케이션의 특징 요구사항(QoS로 표현)을 표시하고 감시토록 한다.

모든 애플리케이션이 하부구조에서 동일한 지원을 요구하는 것은 아니다. 따라서 애플리케이션의 기본적 항목으로 표시된 일반적 기능 요구사항 중에는 특정 애플리케이션에서 원하지 않는 것도 있다. 예를 들어 모든 애플리케이션에서 트레이딩이 사용되지는 않는다. 특정 애플리케이션에서 사용되는 하부구조의 실제적인 요구사항만을 DPE 서브세트로 규정한 것을 애플리케이션 프로파일이라 한다. DPE QoS 및 속성을 위해 정의된 개념과 모델을 이용하여 하부구조에 대한 비기능적 속성도 프로파일에 포함된다.

3.5 TINA-C DPE 구조의 관점에서 본 OMG와의 비교

TINA-C DPE의 관점에서 OMG와 비교하여 그 유사점과 차이점을 기술하면 다음과 같다.

1) 유사점

OMG 객체 서비스 중 다음의 구조적인 요구사항과 목적은 TINA-C DPE 서비스 요구 사항에 적용되었다.

- 객체 서비스 인터페이스는 객체 지향이며 IDL(Interface Description Language)로 기술된다. (OMG-IDL과 TINA-C ODL은 유사함)
- 객체 서비스 사양은 구현에 관련된 사항을 포함하지 않는다.
- 각각의 객체 서비스를 분리해서 명시하고 구현하는 것이 가능하다.
- 객체 서비스는 확장 가능하다.
- 객체 서비스는 다른 환경을 위해 최적화된 많은 구현이 있을 수 있다는 것을 목적으로 명시되어야 한다.
- 객체 서비스는 다양한 플랫폼을 통하여 구현의 이식성을 수용하도록 설계되어야 한다.

표 1의 다섯 가지 서비스는 OMG의 서비스와 유사함을 찾을 수 있다.

표 1 TINA-C DPE 서비스와 OMG 서비스의 비교

TINA-CDPE 서비스	OMG 공통 객체 서비스
트레이딩 서비스	이름 주기(naming) 서비스
구현/사양 레포지토리	구현/인터페이스 레포지토리
라이프 사이클 서비스	라이프 사이클 서비스
통지 서비스	사건(event) 서비스
트랜잭션 관리	트랜잭션/동시성 제어 서비스

2) 차이점

다음의 사항은 TINA-C DPE 서비스와 OMG 공통 객체 서비스간의 차이점 이다.

- OMG에서의 현재 객체 서비스는 연합의 개념이 없다.
- 트레이딩 서비스와 정확하게 일치하는 것을 OMG에서는 찾아볼 수 없다.
- 보안 서비스를 TINA-C에서는 정의하였으나 OMG에서는 유사한 서비스의 사양을 연구 중이다.
- 트랜잭션 관리 서비스가 TINA-C에서 정의 되었으나 OMG에서는 트랜잭션 서비스를 위한 사양은 아직 발표되지 않았다.
- TINA-C의 구성 서비스와 정확하게 일치하는 것을 OMG 공통 객체 서비스에서는 찾아볼 수 없다.
- OMG에서 정의된 관계(relationship) 서비스, 영속(persistence) 서비스, 구체화(externalization)는 TINA-C에서는 고려되지 않고 있다.

이상의 차이점이 발생하는 가장 큰 이유 중의 하나는 TINA-C는 전기통신 서비스에 그 초점을 둔 소프트웨어 구조이고 따라서 TINA-C DPE는 전기통신 애플리케이션을 지원하기 때문이라 할 수 있다. 상대적으로 OMG는 좀 더 광의의 목적을 가지고 있다.

4 TINA-C DPE 모델

4.1 개요

TINA-C DPE 모델(단순히 DPE 모델이라고도 함)은 DPE에서 요구되는 기능을 정의하는 일련의 개념이며 연산 객체를 위한 추상적 실행 환경을 제공한다. DPE 모델은 연산 객체 레벨의 애플리케이션 요소, 하부구조에 속한 다른 추상 시스템, 종단간 정보 전달 기능을 제공하는 커널 전달망(kTN : kernel Transport Network)과 DPE와의 관계를 설명한다.

따라서 세 종류의 모델이 고려될 수 있는 데 연산 모델은 애플리케이션 프로그래머 입장에서 필요한 DPE 요구사항을 결정한다. 배치 모델은 DPE 구조를 정의하며 연결 모델은 연결 메커니즘의 요구사항을 정의한다. 그림 5는 애플리케이션과 커널 전달망 사이에 삽입된 하부구조를 보여 준다.

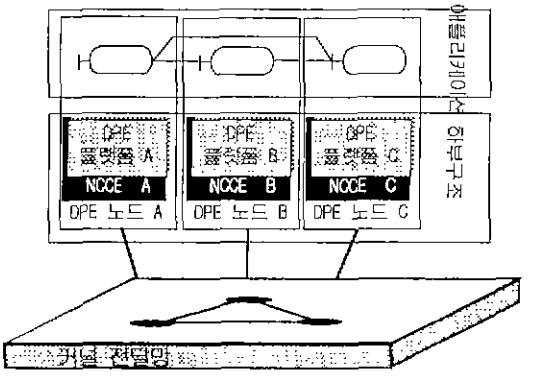


그림 5 하부구조와 환경 모델

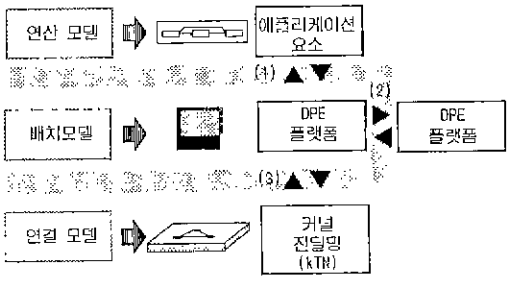
NCCE와 추상 머신 과의 관계는 여기에서 고려되지 않지만 DPE는 NCCE 일부에 대한 추상화로 간주된다. NCCE와 애플리케이션 및 커널 전달망 간의 상호작용이 가능하지만 플랫폼 차원에서 고려 대상은 아니다.

배치 모델에 의하면 하부구조는 다수의 DPE 노드로 구성된다. 각 DPE 노드는 애플리케이션 요소에 DPE 기능을 제공하는 실체인 DPE 플랫폼을 포함한다. 각각의 DPE 노드에 존재하는 NCCE는 DPE 플랫폼을 지원하며 애플리케이션 요소에 추가적인 연산 지원을 제공할 수도 있다.

배치 단위는 배치 개념 측면에서 DPE 모델을 설명한다. 배치 개념은 애플리케이션 요소의 배치와 관리를 조정하며 자원 단위와 분산 단위의 정의를 이용한다. DPE 노드는 그 자체가 기본적인 자원 단위이다.

커널 전달망(kTN)은 DPE 노드들의 종단간 상호접속 기능을 제공하며 연결 모델의 개요는 객체 통신모델에 설명되어 있다. kTN을 제어하고 관리하는 요소는 애플리케이션에 해당되며 따라서 DPE에 속하지 않는다. 그러나 DPE 노드는 kTN을 초기화 하는데 필요한 최소 기능을 제공하는 제어 요소를 갖고 있다. 이 제어 요소의 커널 전달망에 대한 인터페이스는 제어 요소의 부분이 아니라 DPE 구조에 해당된다.

DPE 플랫폼을 기술하는데 사용되는 관련 모델간의 인터페이스 집합이 참조점으로 정의된다. 애플리케이션 요소, DPE 플랫폼, 커널 전달망 상호간에 이루어지는 정보 교환을 위해 세 가지 참조점이 정의되며 각기 해당되는 인터페이스 사양과 관련 모델을 갖는다. 그림 6에 표시된 바와 같이 설계 이식성을 지원하려면 DPE 플랫폼과 애플리케이션 요소간에 정의된 인터페이스가(참조점 (1)) 요구되지만 상호운용성은 플랫폼에서 다른 플랫폼에 제공되는 인터페이스가(참조점 (2)) 중요하다. 이밖에도 플랫폼에서 커널 전달망에 대한 기본적 제어 및 관리 기능을 수행하기 위한 인터페이스가(참조점 (3)) 정의된다.



(1) DPE 플랫폼과 TINA-C 애플리케이션간의 인터페이스
(2) DPE 플랫폼간의 인터페이스
(3) DPE 플랫폼과 커널 전달망과의 인터페이스

그림 6 상호 통신을 위한 모델 및 참조점

4.2 배치 개념

4.2.1 연산 객체의 배치

이 절은 연산 객체의 배치에 사용되는 개념을 TINA-C DPE에서 지원되는 배치 단위를 이용하여 기술한다.

연산 사양은 다수의 상호작용하는 연산 객체로 이루어진다. 연산 객체는 본질적으로 분산의 대상이다. 보통 연산 사양은 공학적 관점에서 여러 종류의 분산 구성이 가능하다. 이때 가능한 분산 구성(공학적 구조) 세트는 QoS 조건 내지는 특정 분산 투명성 등의 요구사항을 만족해야 한다(그림 7 참조).

배치 개념은 연산 객체의 그룹화를 허용하며 그렇게 구성된 객체 그룹에 대해 공통적 특성을 규정한다. 따라서 배치의 개념은 RM-ODP의 합성(composition) 개념과 비교될 수 있다.

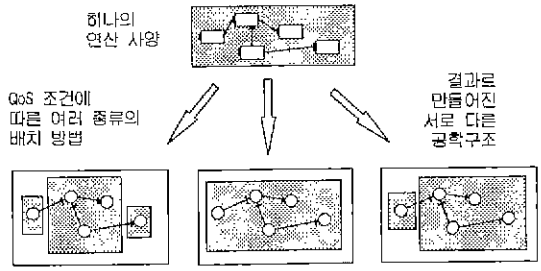


그림 7 공학 단위로 배치되는 연산 객체

연산 객체의 공학 표현 시에는 다음과 같이 연산 관점과 공학 관점으로 나누어서 생각할 수 있다.

1) 연산 관점의 연산 객체

TINA-C 연산 모델링 개념에 규정된 대로 연산 객체(CO : Computational Object)는 데이터(상태)와 행위를 캡슐화 한다. 연산 객체는 다른 객체가 사용할 수 있는 능력(기능)을 제공하며 이러한 능력의 집합은 서비스라고 하는 여러 개의 서브세트로 구성되고 각각의 서비스는 인터페이스를 통해 접근된다.

연산 객체는 여러 종류의 인터페이스를 제공하며 같은 인터페이스 또는 다른 인터페이스를 통해 동시에 접근될 수 있다.

2) 공학 관점의 연산 객체

DPE 구조는 연산 객체를 실행하는 추상적 하부구조를 기술하므로 연산 객체의 개념은 공학 개념으로 명백하게 대응되어야 한다. 따라서 연산 객체에 해당하는 공학 관점의 표현이 정의되며 이를 공학 연산 객체(eCO : engineering Computational Object)라고 한다.

연산 객체와 그에 해당되는 공학 연산 객체 간의 차이는 추상화에 있다. 상호작용에 있어 연산 객체는 다른 연산 객체만이 가능하지만 공학 연산 객체는 공학 연산 객체 이외의 공학 객체와도 가능하다. 즉 공학 연산 객체는 다른 공학 연산 객체와 통신을 설정하는 공학 객체와 상호작용을 하며 특정 분산 투명성을 제공하는 공학 객체와도 상호작용 한다.

연산 객체 템플릿과 eCO 템플릿 간에는 일대일 대응 관계가 성립하므로, 특정 eCO는 여러 연산 객체의 합성이 아니며 하나의 연산 객체가 다수의 eCO에 관련되지도 않는다(그림

8 참조).

eCO 인터페이스는 해당되는 연산 객체의 인터페이스를 표현한다. 그러나 동작 파라메터의 타입 변환이나 공학 객체와 eCO의 상호작용에 필요한 동작의 추가 등을 통해 인터페이스가 수정될 수 있으며 관리 인터페이스가 추가되기도 한다. 이러한 수정과 추가 작업은 연산 객체 템플릿을 eCO 템플릿으로 변환하는 컴파일 도구에 의해서 제공된다. 변환 과정 시 연산 객체에서 정의된 객체의 행위는 변경되지 않는다.

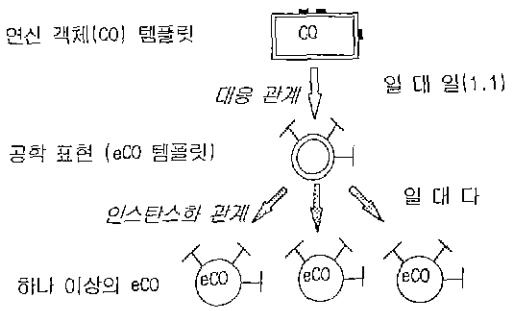


그림 8 연산 객체, eCO 템플릿, eCO

공학 관점의 eCO는 연산 관점에서 정의된 연산 객체의 특성을 상속 받는다. 따라서 eCO 템플릿과 eCO는 다음의 특성을 갖는다.

- eCO 템플릿은 DPE 플랫폼에서 실행되는 eCO를 임의의 수 만큼 인스탄스화 한다 (그림 8 참조).
- eCO는 상태/데이터와 프로세싱을 캡슐화 한다.
- eCO는 캡슐화 된 상태/데이터를 접근할 수 있는 인터페이스 집합을 제공한다.
- eCO에 대한 동작은 같은 템플릿에서 생성된 다른 인스탄스와 동시에 실행 가능하다.
- eCO에 대한 동작은 다른 인터페이스를 통해 동시에 호출 가능하다.
- eCO에 대한 동작은 하나의 인터페이스를 통해 동시에 호출 가능하다.

마지막의 두 가지 특성은 한정된 eCO에만 적용되며, 일반적인 공학 개념에서는 동일 인스탄스의 동시 호출을 처리할 수 없다.

4.2.2 자원 단위

자원 단위는 연산 자원의 집합과 특정 속성을 가진 연산 자원 할당 정책에 사용되는 모델이다.

분산 처리 시스템에 관련된 용어를 정의하고 구조를 설명키 위하여 자원 단위가 사용되며, 자원 단위로 그룹화 된 객체는 객체의 행위에 많은 영향을 미친다.

현재까지 규정된 자원 단위로는 노드와 캡슐이 있다.

노드는 추가적으로 망 노드, 연산 노드, DPE 노드로 세분되며 캡슐은 노드에 포함된다.

1) 노드

망 노드는 망에 존재하는 임의의 노드를 지칭하는 일반적 용어이며 망의 연결과 관리의 단위가 된다.

망 상호접속과 관리의 책임은 망 노드를 경계로 한다. 망 노드 내부의 연결은 망 노드의 관할이며 노드간 통신은 망에서 제공된다. 따라서 망 노드는 OSI의 관점에서 개방 시스템에 비교될 수 있다. 망 노드에는 처리 및 교환 능력이 없는 노드(예, 단말기, 전화기), 약간의 처리 능력을 갖고 있는 노드, DPE 플랫폼을 갖고 있는 노드들이 있다.

망 노드 중에서 약간의 처리 능력을 제공하는 노드를 연산 노드라고 하며, 연산 노드에 반드시 플랫폼이 있는 것은 아니다.

연산 노드 중에서 DPE 플랫폼의 형태로 DPE 구조를 지원하는 노드를 DPE 노드라고 하며, 노드는 연산 자원의 집합을 표현하는 모델이다. 하나의 연산 노드에 배치된 객체는 다른 연산 노드에 있는 객체와는 독립적으로 노드의 연산 자원을 사용한다.

DPE 노드(이하에서는 단지 노드라고 함)는 여러 가지의 연산 자원으로 구성된다. 노드는 여러 종류의 연산 자원을 자율적으로 다른 노드에 무관하게 관리한다. 노드에서 관리되는 연산 자원의 종류로는 처리 자원(예, 프로세스, 태스크, 쓰레드, 스케쥴러), 메모리 자원(예, 휘발성 메모리, 지속성 메모리), 통신 자원(예, 노드 내부 통신 메커니즘, 망 접근 메커니즘) 등이 있다.

노드는 하나의 자원 관리 영역을 나타내며

노드의 모든 자원은 자율적으로 관리된다. 자율적이란 의미는 노드가 독립적인 동작을 수행할 수 있다는 것으로 예를 들어 노드 내 통신은 다른 노드에 영향을 받지 않는다.

노드 내 연산 자원의 할당은 DPE 커널에서 캡슐화 된다. 또한 커널은 eCO 실행과 통신을 지원하는 메커니즘을 제공한다. 같은 노드에 있는 객체는 같은 커널을 공유한다.

자원의 실제적 구성은 노드마다 서로 다르지만 노드에 포함된 자원은 DPE 플랫폼이 연산 객체를 적절히 지원할 만큼 충분해야 한다. 이 밖에도 개방 시스템의 일부로서 노드는 최소한 하나의 망 접근 메커니즘(통신 프로토콜)을 시스템 내의 다른 노드와 공통적으로 사용해야 한다.

DPE 노드의 예로는 독립적인 동작이 가능한 단일 프로세서 시스템, 다중 프로세서 시스템, LAN상에 분산된 시스템으로 분산 운영체제를 갖고 있는 경우를 꼽을 수 있다.

TINA-C에서의 노드의 정의는 RM-ODP에서 사용되는 노드의 정의, 즉 노드는 자원 관리 영역으로서 자원 독립성을 가진 공학 단위라는 것과 매우 유사하다. 또한 노드의 정의는 OSI의 개방 시스템 정의에도 부합한다. Bellcore의 INA에서는 망 노드와 가입자 댁내 장치(CPE : Customer Premises Equipment)를 구분한다.

2) 캡슐

캡슐은 DPE 노드의 연산 자원을 할당하는 공학 단위이다. 캡슐에 배치된 객체들은 노드의 커널에서 자원을 할당받는 데 있어 같은 할당 정책을 공유하며, 이 할당 정책은 같은 노드에 있는 다른 캡슐의 할당 정책과는 서로 다르다.

캡슐을 구성하는 객체의 집합이 다른 캡슐의 객체로부터 보호되도록 자원 할당 정책이 정의되어야 하며 객체들의 실시간 특성도 마찬가지로 보장되어야 한다. 자원 관리에 대한 커널의 접근 방식에 따라 노드의 자원이 캡슐간에 공유되거나 분할되어 각 캡슐에서 독점적으로 사용된다. 후자의 경우 한 캡슐의 객체는 자신의 자원 집단(pool)을 갖게 되며 다른 캡슐의 객체와는 서로 무관하다. 전자의 경우에는 자원이 공유되며 지연 또는 자원의 부족 현상이 발생할 수 있다.

캡슐의 예로는 UNIX에서의 프로세스와 다중 프로세서 시스템에서 단일 프로세서를 들 수 있다. 전자의 경우는 UNIX용 ANSAware 버전에서 볼 수 있는 예로서 캡슐은 하나의 UNIX 프로세스에 대응된다. UNIX 시스템의 ANSAware 캡슐은 처리 자원인 프로세스에 특별한 자원 할당 정책을 구현한 쓰레드 패키지를 제공한다. 단일 프로세서가 캡슐에 해당되는 경우 처리 자원은 단지 하나의 캡슐에서만 독점적으로 사용된다. 메모리 자원의 공유는 시스템 내 공유 메모리 존재 여부에 따라 결정된다.

RM-ODP에서 캡슐은 자원 할당과 캡슐화의 공학 단위로 정의되어 TINA-C의 캡슐 개념과 잘 상응한다. TINA-C DPE에서는 자원 할당 단위 개념이 강조된 편이며 보안 속성은 추후에 다루어질 것이다.

CHORUS에서 가상 머신을 모델링하는 액터 개념은 캡슐과 유사하지만 TINA-C DPE 캡슐과는 달리 액터는 분산 단위이기도 하다. TINA-C에서는 분산 단위와 자원 단위의 개념을 분리 한다.

TINA-C에서 캡슐은 자원 할당의 단위이며 분산의 단위라 할 수는 없다. TINA-C에서 분산의 단위는 뒤에 설명될 클러스터이다.

4.2.3 분산 단위

분산 단위는 분산 특성에 있어 공통점을 가진 객체 인스탄스의 집합이다(특히 eCO의 집합). 따라서 분산 단위는 분산 요구사항에 따라 객체를 그룹화하는 데 사용된다.

1) 분산 단위의 특성

분산 단위는 분산 특성에 따라 분류되며 다음의 속성들을 갖는다.

- 인스탄스화의 단위 : 객체들이 항상 같이 인스탄스화 될 경우, 해당되는 eCO 의 집합은 인스탄스화 단위이다.
- 배치의 단위 : eCO들이 노드의 같은 캡슐 내에 연합적으로 배치될 때 eCO의 집합은 배치 단위이다.
- 활성화의 단위 : 활성화 기능(비활성화,

재활성화 포함)이 단위 내의 모든 eCO에 공통적으로 적용될 때 eCO의 집합은 활성화 단위이다.

- 이주의 단위 : eCO들이 한 위치에서 다른 위치로 집단적으로만 이주하면 그때의 eCO 집합을 이주 단위라 한다. 어떤 시점에서 객체 전부가 이주하거나 하나도 이주를 하지 않은 것처럼 간주되며 객체의 일부만이 이주한 것처럼 보이는 경우는 없다.

이러한 분산 단위의 속성은 다양한 객체 패키지의 특별한 경우라 할 수 있으며 객체 그룹의 연산 개념과 관련되어 있다.

2) 분산 단위의 상호 의존성

분산 단위에 관한 위의 속성들은 상호간에 어떤 관계를 성립시킨다. 즉 이주 단위는 하나 이상의 활성화 단위로 구성된다. 이주 과정에서는 비활성화와 재활성화가 주요 단계이므로 이주 단위 내부의 객체와 외부의 객체 모두가 포함된 활성화 단위를 생성하는 것은 비합리적이다. 또한 이주 단위는 하나 이상의 배치 단위로 구성된다. 배치 단위 내의 객체들은 같이 존재하므로 이주 시에도 함께 이주해야 한다. 배치 단위는 활성화 및 인스탄스화 단위와 상호 독립적이다. 인스탄스화 단위는 나머지 세 가지 분산 특성과 관련이 없다.

3) 클러스터

이주 과정은 객체를 비활성화시켜 전달 가능 상태로 만든 후 새로운 위치에서 객체를 인스탄스화하고 재활성화하는 작업이기 때문에 세 가지 분산 속성이 결합된 분산 단위를 정의하는 것이 편리하다. 클러스터는 이에 해당되는 개념으로 배치, 활성화 및 이주 속성을 동시에 갖는 eCO의 집합이다. 클러스터는 캡슐 내로 캡슐화 되며, 하나의 eCO가 서로 다른 클러스터에 속할 수는 없다.

클러스터 내부 eCO 사이의 관계는 항상 지역적이며 eCO 간 상호작용에는 다음의 규칙이 적용된다.

- 클러스터간 상호작용 : 서로 다른 클러스터에 존재하는 eCO 사이의 상호작용을 모델링하는 공학 개념은 채널이다. 채널 개념은 분산 투명성(접근, 위치 등) 메커니즘에 의해 이질적이고 분산된 환경에서의 상호 연동을 수행한다.
- 클러스터 내 상호작용 : 클러스터 내부의 eCO를 위한 상호작용 메커니즘은 공학 모델링 개념에서 취급되지 않는다.

인스탄스화 속성은 클러스터의 특성이 아니기 때문에 클러스터에 대한 객체의 추가와 제거는 동적으로 수행이 가능하다.

향후에는 객체 패키징의 특정한 요구사항을 지원하는 여타의 분산 단위도 정의되리라 예상된다. 보다 유연한 분산 단위에 관한 정의는 패키지 개념과 결합되어 연구될 것이다.

RM-ODP에서 사용된 클러스터의 개념과 TINA-C에서 정의된 개념은 비슷하다. RM-ODP에서 클러스터는 비활성화 및 재활성화의 공학 단위로서 기본 공학 객체의 집합으로 정의되며 이주 및 배치의 단위로 이해되기도 한다.

배치 개념에 관련된 노드, 캡슐, 클러스터 및 eCO간의 관계를 OMT(Object Modeling Technique) 다이어그램으로 표현하면 그림 9와 같다. OMT는 TINA-C에서 사양을 만들어내기 전에(TINA-C에서는 사양을 ODL등으로 기술함) 객체 타입과 관계 타입을 규정하는데 사용된 방법으로 그림 9를 간략히 설명하면 다음과 같다(OMT 표기의 상세한 사항은 참고문헌 [23] 참조).

- 클러스터는 eCO들로 구성되며 하나의 eCO는 하나의 클러스터에만 속한다.
- 캡슐은 클러스터들로 구성되며 하나의 클러스터는 하나의 캡슐에만 속한다.
- DPE 노드는 최소한 하나 이상의 캡슐과

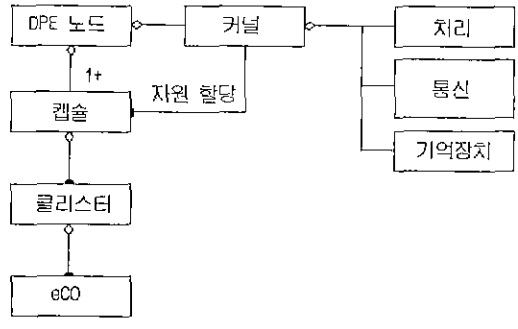


그림 9 배치 개념간의 관계

커널로 구성되며 하나의 캡슐 및 커널은 하나의 DPE 노드에만 속한다.
- 커널은 캡슐에게 자원을 할당한다.
- 커널은 처리, 통신, 기억장치 등의 기능으로 구성된다.

5. 결 론

이질적인 형태의 분산 환경에서 분산 애플리케이션의 개발 및 사용을 용이하게 하기 위해서는 정보망 서비스를 위한 하부구조로 표준 분산 플랫폼이 개발되어야 하며 이러한 분산처리 플랫폼은 유연한 망구조 개념과 개방형 분산처리 개념을 통합적으로 지원해야 한다.

본 고에서는 TINA-C의 전체 구조 중 분산처리 플랫폼에 관하여 기술하였다. 플랫폼이란 구조의 실현이라고 할 수 있는데, 분산처리 플랫폼을 위한 구조로서 TINA-C의 DPE 구조를 주로 공학 모델링의 개념으로 기술 하였다. TINA-C DPE의 기본구조를 고찰함으로써 DPE 공통 서비스에 관한 요구사항이 도출되었으며 이것을 OMG와도 비교하였다.

DPE 구조에서 제공되는 서비스를 정확히 정의하기 위해서는 DPE에서 요구되는 기능을 정의하는 일련의 개념이 필요한데 이것이 TINA-C DPE 모델이다. TINA-C DPE 모델에서는 배치의 개념이 도입되었으며 자원 단위 및 분산 단위로 캡슐과 클러스터의 개념을 이용하였다.

본 고에서는 TINA-C의 개념을 기술할 때 가능한 한 유사한 다른 모델(OMG, RM-ODP 등)과도 비교 하여 설명하였다.

참고문헌

[1] P. Graubmann, W. Hwang, M. Kudela, K. MacKinnon, N. Mercouroff, N. Watanabe, Engineering Modelling Concepts, TB-NS.005-2.0-94, TINA-C, December 1994.

[2] W.Hwang, N. Mercouroff, N. Watanabe, TINA DPE Service Specifications, TR-HW.001-1.0-94, TINA-C, December 1994.

[3] Dave Brown, Stefano Montesi, Requirements upon TINA-C architecture, TB-MH.002-2.0-94, TINA-C, February 1995.

[4] Martin Chapman, Stefano Montesi, Overall Concepts and Principles of TINA, TB-MDC.018-1.0-94, TINA-C, February 1995.

[5] L.A. de la Fuente, Tony Walles, Management Architecture, TB-GN.010-2.0-94, , TINA-C, December 1994.

[6] Harvey Rubin, An Overview of the TINA Consortium Work, TB-G1.HR.001-1.0-93, TINA-C, December 1993.

[7] F.Dupuy, P.Graubmann, K. Kanasugi, E. Kelly, M.Kudela, J.C.Moreno, N. Natarajan, N.Watanabe, G.Wheeler, DPE Phase 0.1 Specification Draft, TP-AD.NW.001-1.0-93, TINA-C, December 1993.

[8] H.Christensen, E.Colban, Information Modelling Concepts DRAFT, TB-EAC.001-1.1-93, TINA-C, November 1994.

[9] N.Natarajan, F.Dupuy, N.Singer, H. Christensen, Computational Modelling Concepts, TB-A2.HC.012-1.2-94, TINA-C, February 1995.

[10] Raimo Kantola, Mapping of TINA Components into Equipment, TP-RKA.002-1.0-94, TINA-C, February 1995.

[11] M.Kudela, K.MacKinnon, Engineering Modelling Concepts(DPE Kernel Specifications), TR-KMK.001-1.1-94, TINA-C, November, 1994.

[12] Jon Siegel, Ph.D.,Common Object Services Specification, Volume 1, OMG Document Number94-1-1, OMG, March 1994.

[13] DEC, HP, et al., The Common Object Request Broker : Architecture and Specification, Revision 1.2, December 1993.

[14] Digital Equipment Corporation, ObjectBroker 2.5 Release Notes, AA-QA9FA-TE, August 1994.

[15] ISO/IEC 10746-2.2 / ITU-T Draft Rec. X 901, Basic Reference Model of Open Distributed Processing - Part 1 : Overview and Guide to Use, International Organization for Standardization and International

Electrotechnical Committee, June 1993.

[16] ISO/IEC 10746-2.2 / ITU-T Draft Rec. X. 902, Basic Reference Model of Open Distributed Processing-Part 2 : Descriptive Model, International Organization for Standardization and International Electrotechnical Committee, June 1993.

[17] ISO/IEC 10746-3 / ITU-T Draft Rec. X. 903, Basic Reference Model of Open Distributed Processing-Part 3 : Prescriptive Model, International Organization for Standardization and International Electrotechnical Committee. April 1994.

[18] ISO/IEC 10746-5 / ITU-T Draft Rec. X. 905, Basic Reference Model of Open Distributed Processing - Part 5 : Architectural Semantics, International Organization for Standardization and International Electrotechnical Committee, December 1991.

[19] Document RM.099.02, An Overview of ANSAware 4.1, ANSA, February 1993.

[20] Document RM.102.02, ANSAware 4.1 Application Programming in ANSAware, ANSA, February 1993.

[21] M.P.Ferro, INA Cycle 1 Distributed Processing Environment(DPE) Specification (Issue 2),TM-NWT-021900, Bellcore, December 1992.

[22] N.Natarajan, INA Cycle 1 Framework Architecture(Issue 2), TM-NWT-021896, Bellcore, December 1992.

[23] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy and William Lorensen, Object-Oriented Modeling and Design, Prentice Hall : Englewood Cliffs, N.J., 1991.

박 성 원

1985 고려대학교 전자공학과 졸업(B.S)
1987 고려대학교 대학원 전자공학과 졸업(M.S)
1995 고려대학교 대학원 전자공학과 수료(박사과정)
관심분야 : 정보통신망, 컴퓨터 네트워크, 분산 시스템

선 경 섭

1985 고려대학교 전자공학과 졸업(B.S.)
1987 고려대학교 대학원 전자공학과 졸업(M.S.)
1995 고려대학교 대학원 전자공학과 박사과정
관심분야 : 정보통신망, 컴퓨터 네트워크, 분산 시스템

안 순 신

1973 서울대학교 공과대학 졸업(B.S.)
1975 한국과학기술원 전기 및 전자공학과 졸업(M.S.)
1979 프랑스 ENSEEIHT 공학박사(Ph.D.)
1993~1982 아주대학교 전자공학과 조교수
1982~현재 고려대학교 전자공학과 교수
1991~1992 미국 NIST 방문연구원
관심분야 : 정보통신망, 컴퓨터 네트워크, 분산 시스템