CBD 상에서 컴포넌트 설계를 위한 통합 컴포넌트 메타 모델에 관한 연구

조은숙*

A Study for Integrated Component Meta-model for Component Design in CBD

Eun Sook Cho

Abstract

Lately CBD methodologies like as CBD96, Catalysis, Fusion, and SCIPIO have been introduced. These methodoloties has their own proprietary component reference model. Using proprietary reference model falls interoperability among methodologies. Furthermore it can cause confusion and difficulty for component developers. In this paper, we propose a integrated component meta-model for support consistency and interoperability among component designs. Also, we compare our proposed meta model to existing component reference model by using component's characteristics. We expect that it is easy to add new meta model elements and extends meta-model by using integrated component meta-model.

Key Words: 컴포넌트기반 개발, 컴포넌트 참조 모델, 컴포넌트 메타 모델

1. 서론

컴포넌트 기반 소프트웨어 개발에 대한 관심이 고조되면서 RUP, Catalysis, CBD96, UNIFACE, Fusion[2,3,4,5,6,7,9] 등과 같은 CBD 방법론들이 소개되고 있을 뿐만 아니라 여러 CASE 도구와 EJB, CCM, .NET 등과 같은 컴포넌트 기술 플랫폼들이 제시되고 있다[1,8,10,11]. 또한 이러한 방법론들이나 문헌 및 기타 연구들에서 컴포넌트 참조 모델들을 다양하게 제시하고 있다. 그러나 이러한 컴포넌트 참조 모델들을 보면 컴포넌트에 대한 동일 개념에 대해서 서로 다른 표기법과 장치명을 제시할 뿐만 아니라 일부 참조 모델들은 컴포넌트의 특성들을 일부는 반영하지 못하는 경우도 있다. 이로 인해 서로 다른 컴포넌트 참조모델을 기반으로 설계 또는 개발 된 컴포넌트들 간의 불일치성과 호환성 저하의 문제가 발생되고 있다 그리고 컴포넌트 플랫폼을 고려하지 않은 개략 설계에서 정의하고 있는 컴포넌트 참조 모델의 요소들과 컴포넌트 기술 플랫폼에서 정의하고 있는 컴포넌트 참조 모델의 요소 및 장치들 간의 매핑 관계가 정의되어 있지 않음으로 인해서 컴포넌트 설계자와 개발자들 간의 많은 의사소통의 어려움 또한 증가시키고 있다. 본 논문에서는 이러한 문제점을 해결하기 위해 현존하는 여러 참조 모델들을 단일화 한 통합 참

* 동덕여자대학교

조 모델을 UML[8] 클래스 다이어그램 표기법을 기반으로 한 메타 모델 을 제시하고자 한다.

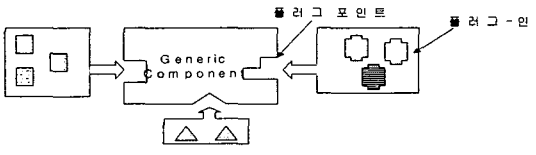
본 논문의 구성은 다음과 같다. 2장에서는 관련연구로서 현존하는 컴포넌트 참조 모델들을 제시하고, 3장에서는 2장에서 제시된 기존 컴포넌트 참조 모델들을 통합한 통합 컴포넌트 참조 모델과 메타모델을 명세 수준과 구현 수준으로 나누어 제시한다. 다음으로 4장에서는 제시된 통합 컴포넌트 참조 모델이 기존의 방법론 등에서 제시하는 메타모델들을 비교한다 마지막으로 5장에서 결론 및 향후 연구과제를 제시한다.

2. 컴포넌트 참조 모델

2.1. Catalysis의 컴포넌트 참조모델

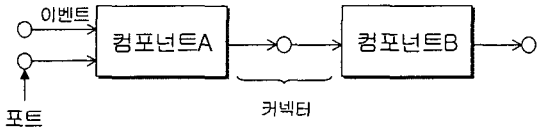
Catalysis 방법론에서는 컴포넌트는 개별 단위로 독립적으로 개발되어서 배포될 수 있는 응집력 있는 소프트웨어 패키지로서 다른 컴포넌트들과 조립할 수 있도록 서비스를 제공하는 인터페이스와 사용하는 인터페이스가 정의되어 있어야 한다고 제시하고 있다[4,5]. 또한 여기서는 하나의 컴포넌트 패키지에는 제공하는 인터페이스 목록, 사용하는 인터페이스 목록, 컴포넌트 명세서(스펙), 실행 코드, 확인 코드, 설계 등이 포함되어 있어야 한다고 제시한다.

한 컴포넌트가 재사용성이 높아지기 위해서는 여러 유사 응용 프로그램들 간에 공통적인 부분들이 구현되어 있어야 할 뿐만 아니라 각각의 응용 프로그램마다 필요에 따라 특화될 수 있도록 하는 메커니즘을 제공해야 한다고 제시한다. 이 메커니즘을 그림 1에 표현된 것처럼 "플러그 포인트(Plug-Point)로" 정의하고 있다.



〈그림 1〉 플러그-인 포인트

Catalysis 방법론에서는 컴포넌트 모델을 컴포넌트, 커넥터(Connector), 포트(Mapping schema), 포트 유형(Mapping schema Category), 그리고 컴포넌트 아키텍쳐로 정의하고 있다.(그림 2 참조)



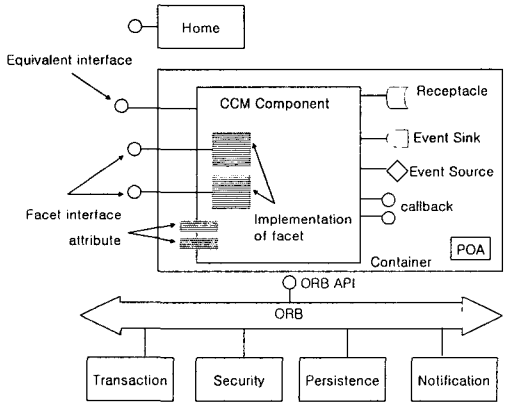
〈그림 2〉 컴포넌트 모델

컴포넌트는 하나의 유용한 작업을 수행하는 능동적인 요소로서, 개별적인 객체, 객체들의 집합, 혹은 서브시스템을 지칭하고 있다. 그리고 객체와 달리 독립적으로 배포될 수 있는 패키지 단위가 된다고 제시하고 있다. 커넥터는 컴포넌트들 간의 통신 수단으로서, 하나의 커넥터는 단순하게는 함수 호출 혹은 그룹 함수 호출이 될 수도 있고, 또는 CORBA나 COM을 통해 전달되는 메시지가 될 수도 있다고 표현하고 있다. 포트는 한 컴포넌트가 다른 컴포넌트들과 연결하는데 필요한 구별된 수단들을 의미한다. 커넥터는 서로 다른 컴포넌트들의 포트들을 연결한다. 따라서 각각의 포트는 각각의 이름을 가지게 된다. 일반 객체에서 여러 오퍼레이션들 가운데 각각의 오퍼레이션들은 메시지 이름을 가짐으로써 다른 오퍼레이션들과 구별되는 원리와 같은 개념이다.

2.2 CORBA Component 참조모델

CCM은 두 단계의 컴포넌트로 분리된다[6]. Basic 컴포넌트는 CORBA object를 컴포넌트화하는 메커니즘을 제공하며 EJB와 매핑되거나 통합될 수 있다. Extended 컴포넌트는 Basic 컴포넌트에 비해 많은 기능을 제공한다. Basic 컴포넌트와Extended 컴포넌트 모두 component

home에 의해서 관리된다.



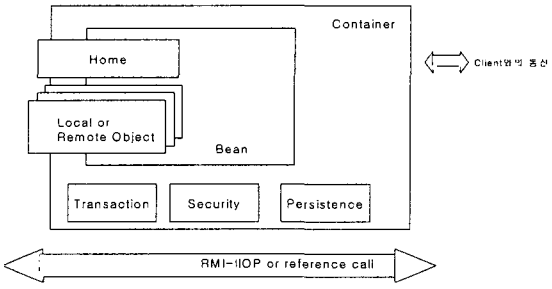
〈그림 3〉 CCM 참조 모델

Equivalent interface는 컴포넌트를 식별하는 역할을 한다. Client는 equivalent interface를 통해 facet interface의 reference를 얻을 수 있다. Facet을 통해 컴포넌트는 client에게 필요한 기능을 제공한다. 컴포넌트는 외부의 객체 reference를 참조하여 객체의 오퍼레이션을 호출할 수 있다. 이 경우 컴포넌트와 객체와의 연결을 receptacle로 정의한다. Attribute는 Get, Set 메소드를 통해 컴포넌트에서 얻거나 수정할 수 있는 값이다. Basic component는 facet, receptacle, event source, event sink를 갖지 않으며 단지 attribute만을 제공한다.

Event source는 컴포넌트가 외부에 event는 보내는 부분이다. Event sink는 컴포넌트가 event를받는 부분이다. Callback은container가 컴포넌트를 호출할 수 있도록 컴포넌트가 cotainer에게 제공하는 API이다. Container는 Mapping schemaable Object Adapter(POA)를 통해 사용자의 컴포넌트 호출을 컴포넌트에게 전달한다. 사용자와 컴포넌트 사이의 통신이나 컴포넌트와 컴포넌트 사이의 통신은 Object Request Broker (ORB)를 통해 이루어진다. 컴포넌트는 ORB를 통해 Transaction, 보안, 영구성, Notification에 대한 서비스를 제공받는다. Home 인터페이스는 컴포넌트의 생성과 삭제 등 컴포넌트를 관리하는 역할을 한다.

2.3 EJB 컴포넌트 참조 모델

EJB 컴포넌트 모델은 CCM 컴포넌트 모델과 유사하다[7]. 컴포넌트는 컨테이너 안에서 실행되며 home 객체를 통해 관리된다.



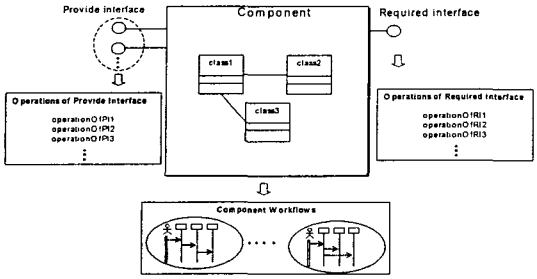
〈그림 4〉 EJB 참조 모델

그러나 CCM 컴포넌트와 달리 EJB 컴포넌트는 하나의 component interface를 통해서만 참조된다. 따라서 Bean은 여러 개의 인터페이스를 갖지 않는다. EJB 컴포넌트 모델은 Local or Remote Interface와 내부 구현 로직으로 분리된다. CCM이 ORB안에transaction, 보안, 영구성에 대한 서비스를 포함하는 것과는 달리, EJB는 container에서 서비스를 제공한다.

3. 범용 컴포넌트 참조 모델과 메타모델

2장의 컴포넌트 참조 모델들을 기반으로 새로운 컴포넌트 참조 모델을 제안한다. 그림 5는 범용(Generic) 컴포넌트 참조 모델이다.

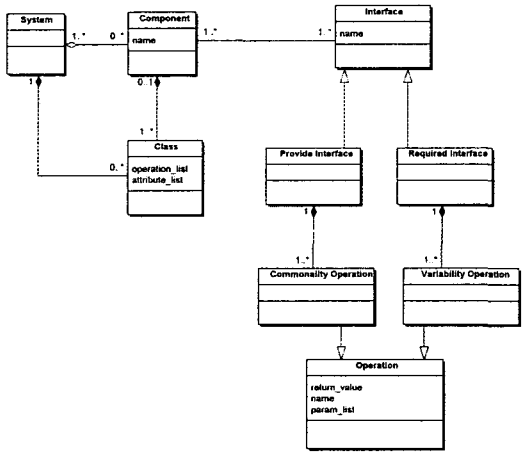
그림 5는 컴포넌트의 구조적인 요소와 동적 요소를 함께 보여준다. 컴포넌트는 구조적으로는 Provide Interface와 Required interface, 컴포넌트 내부 클래스들로 이루어져 있다. 동적으로는 provide interface의 한 오퍼레이션을 호출함으로써 컴포넌트 워크플로우가 발생한다.



〈그림 5〉 범용 컴포넌트 참조 모델

컴포넌트의 구조적인 요소를 상세히 설명하면 다음과 같다. 그림 5를 보면 컴포넌트는 여러 개의 provide interface를 가질 수 있으며 한 provide interface는 여러 개의 오퍼레이션으로 이루어진다.

Required Interface는 컴포넌트의 가변성을 설정한다. 가변성이란 사용자의 설정에 의해서 컴포넌트 내부 구성이나 워크플로우가 변하는 것을 의미한다. 따라서 컴포넌트의 가변성은 컴포넌트 내부 클래스에 표현될 것이다. Required Interface의 호출에 의해 가변성이 설정되면 컴포넌트 내부 클래스 중 실행 가능한 클래스가 일부분으로 제한될 것이다. 이런 클래스들의 집합을 configuration이라고 부른다. 그림 6은 컴포넌트의 구조적인 요소를 UML 클래스 다이어그램을 이용한 메타 모델로 표현한다[8].

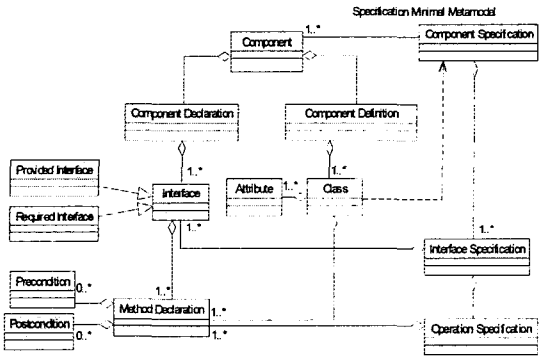


〈그림 6〉 컴포넌트의 구조적 구성 요소

그림 6을 보면 컴포넌트는 클래스와 인터페이스로 이루어져 있다. 인터페이스는 다시 컴포넌트 사용자에게 서비스를 제공하는 Provide Interface와 사용자 스스로 컴포넌트의 기능을 수정하거나 추가하기 위해 사용되는 Required Interface로 나뉜다.

3.1 명세 수준의 메타 모델

명세 수준 최소 메타 모델은 현존하는 여러 CBD 방법론들에 공통적으로 포함되어 있는 컴포넌트 참조 정보들을 표현한 모델이다. 그림 7에 나타난 것처럼, 컴포넌트는 컴포넌트 선언부와 컴포넌트 정의부로 구성된다. 컴포넌트 선언부는 하나 이상의 인터페이스들을 포함하고 있으며, 각각의 인터페이스는 하나 이상의 오퍼레이션들을 내포하고 있다.

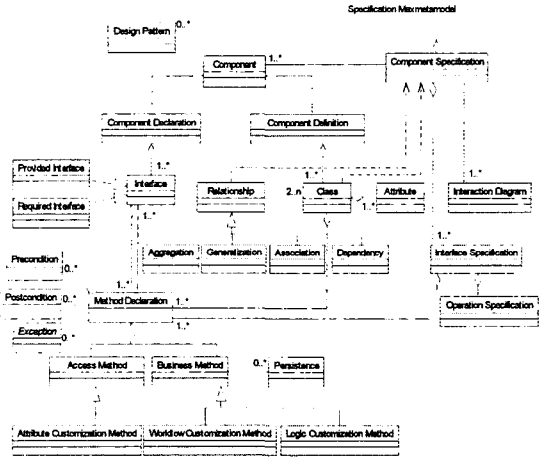


〈그림 7〉 명세수준 최소 메타모델

인터페이스는 제공 인터페이스와 요구 인터페이스로 구분된다. 인터페이스에 선언된 함수 선언은 함수 시그네쳐(Signature), 사전 조건, 그리고 사후 조건으로 구성된다. 컴포넌트 정의부는 컴포넌트의 몸체를 의미한다. 컴포넌트 정의부에는 하나 이상의 클래스들이 포함되어 있다. 이 클래스들은 컴포넌트 선언부에 선언된 인터페이스들을 반드시 구현해야 한다. 이 클래스와 속성에 대한 정의는 UML 정의를 따른다. 또한 컴포넌트 명세부는 하나 이상의 인터페이스 명세와

컴포넌트들을 포함한다.

명세 수준 최대 메타 모델은 그림 8에 제시되고 있다. 명세 수준 최대 메타모델은 CBD 방법론들에서 한정적으로 정의하고 있는 정보들까지 표현한 메타 모델이다. 명세 수준 최대 메타모델에서는 메소드 선언이 접근 메소드와 비즈니스 메소드 부분으로 분류된다. 속성 특화 메소드는 접근 메소드의 서브 클래스이다. 그 이유는 일반적인 get 또는 set 함수는 속성 값만 특화시켜주는 메소드인 반면에 속성 특화 메소드는 가변적인 속성 값 뿐만 아니라 가변적인 속성 타입까지도 지원하기 때문에 접근 메소드의 서브 클래스로 표현한다.



〈그림 8〉 명세 수준 최대 메타모델

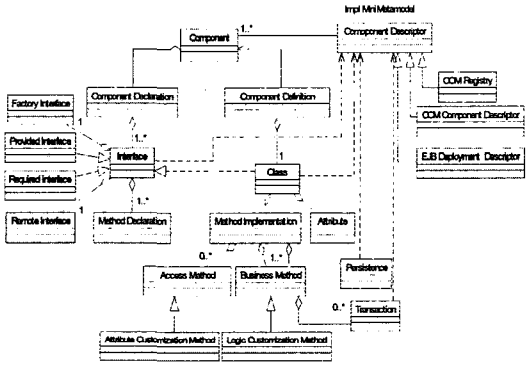
비즈니스 메소드는 하나 이상의 영구적인 정보들을 포함할 수도 있다. CBD 방법론들 가운데 일부는 한 컴포넌트 정의 내부에 클래스들간의 관계 정보를 정의한다. 이러한 관계 정의는 UML의 정의를 따른다.

3.2 구현 수준의 메타 모델

구현 수준 메타 모델은 EJB, CCM, 그리고 COM과 같은 컴포넌트 기술 플랫폼을 이용해서 컴포넌트를 개발할 때 사용한다. 구현 수준 메타 모델은 다시 최소 메타 모델(그림 9)과 최대 메타모델(그림 10)로 분류된다.

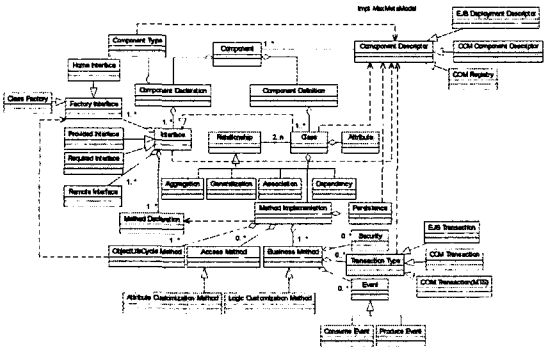
구현 수준 최소 메타 모델은 컴포넌트 구현과 관련된 추가적인 정보들을 표현한다. 예를 들어 factory 인터페이스와 원격 인터페이스는 컴포넌트 구현 시 필요하다. 또한 트랜잭션과 관련된 정보도 비즈니스 메소드에 반영되어야 한다.

명세 수준 메타 모델에서는 컴포넌트 정보가 컴포넌트 명세에 기술되지만, 구현 수준 메타 모델에서는 컴포넌트 정보는 컴포넌트 디스크립터(Descriptor)에 기술된다.



〈그림 9〉 구현 수준 최소 메타 모델

구현 수준 최대 메타 모델은 여러 컴포넌트 플랫폼들에 있어서 특정한 정보들까지도 포함한 모델이다. 따라서 컴포넌트 플랫폼에 따라 일부 다르게 표현되는 정보들까지도 이 모델에서는 표현하게 된다.



〈그림 10〉 구현 수준 최대 메타 모델

4. 컴포넌트 참조 모델들의 비교

이 장에서는 제시된 컴포넌트 참조 모델과 메타모델이 컴포넌트의 특성들을 얼마나 잘 지원하는지를 기존의 CBD 방법론과 플랫폼들에서 제시하는 메타모델들과 비교하고자 한다 비교 결과는 에 제시되어 있다. 비교 결과표에서 삼각형은 부분 지원을 의미한다.

〈표 1〉 메타모델들 간의 비교

Factor		CBD 96	Catal-ysis	Fusion	UNIFACE	SCI-PIO	UML	제시 모델
Compo-nent	Object	O	O	O	O	O	O	O
	Composite	O	O	O	O	O	O	O
	Specification	O	O	O	O	O	O	O
	Dependency	O	O	O	O	O	O	O
	Implementation	O	O	O	O	O	O	O
Comonent Specifi-cation	Specification	O	O	O	O	O	O	O
	Pre-condition	O	O	X	O	O	X	O
	Post-Condition	O	O	X	O	O	X	O
Class	One	O	O	O	O	O	O	O
	Multiple	O	O	O	O	O	O	O
Inter-face	One	O	O	O	O	O	O	O
	Multiple	O	O	O	O	O	O	O
	Provided	O	O	O	O	O	O	O
	Required	O	O	O	O	O	O	O
	Attribute	O	O	X	X	X	O	X
	Specification	O	O	X	X	X	O	O
Custom ization	Attribute	O	O	O	O	O	O	O
	Logic	△	△	△	△	△	△	O
	Workflow	X	X	X	X	X	X	O

5. 결론 및 향후 연구과제

지금까지 컴포넌트에 대한 메타 모델을 크게 범용적인 수준, 명세 수준, 그리고 구현 수준에서 정의하여 제시하였다. 또한 명세 수준과 구현 수준에서는 각각 최소 메타 모델과 최대 메타 모델로 구분하여 정의하였다. 또한 제시된 메타 모델이 기존의 방법론들에서 제시하고 있는 메타모델들에 비해서 컴포넌트의 개념과 특성에 대한 지원 정도를 검증하기 위하여 컴포넌트 특성들을 가지고 비교하였다.

본 논문에서 제시한 메타 모델을 통해 컴포넌트 방법론들과 기술 플랫폼에서 제시하고 있는 모델들을 통합할 수 있을 뿐만 아니라 새로운 모델을 이 모델에 추가적으로 추가 또는 확장이 용이할 것으로 기대한다.

참고문헌

[1] Heineman, G. T., Council, W T., Component-based Software Engineering, Addison Wesley, 2001.

[2] Sterling Software Inc., "The CBD Standard Version 2.1", Sterling Software, July 1998.

[3] Veryad, R., "SCIPIO: Determining the requirements for software components," SCIPIO Consortium, January 1999.

[4] Butler Group, "Catalysis: Enterprise Components with UML," at URL: http://www.catalysis.org, pp.2, 1999.

[5] Desmond F. D'Souza and Alan Cameron Wills, "Objects, Components, and Frameworks with UML," Addison Wesely, 1999.

[6] HP Company, "Engineering Process Summary: Fusion 2.0", Hewlett-Packard Company, January 1998.

[7] Compuware Corp., *UNIFACE Development Methodology V7.2*, COMPUWARE Corp., 1998.

[8] Roman, E., Mastering Enterprise JavaBeans Second Edition, John Wiley and Sons, Inc., 2002.

[9] UML Specification v1.4, OMG, Inc., September, 2001.

[10] Rogerson, D., *Inside COM*, Microsoft Press,

1997.

[11] Object Management Group, *CORBA Components Volume 1*, Joint Revised Submission, OMG TC Document, August 1999.