CORBA Component Model상에서의 재귀적 컴포넌트 결합

윤동찬°, 백경원
아주대학교 정보통신전문 대학원

The recursive component composition based on the CORBA Component Model

Yoon, Dong Chan , Beak Kyung Won
Graduate School of Information & Communication, Ajou University
E-mail : dcyoon@webgateinc.com, beeback@magang.ajou.ac.kr

요 약

CBSD의 핵심 연구 과제 중 하나인 컴포넌트 결합에 의한 응용 프로그램 개발에 대한 많은 연구가 진행되고 있지만 구체적 개발 환경에 대하여 아직 미비한 실정이다. 이에 본 논문에서는 CCM 컴포넌트들의 재귀적인 결합과 이에 기반을 둔 응용 프로그램 개발 방안을 제안하고자 한다. 이를 위해 컴포넌트 서비스의 기능적 결합을 기술하는 결합 명세서를 작성하고, 결합 명세서에 기반하여 컴포넌트를 결합하고자 한다. 또한 결합 컴포넌트의 결합성과 적합성을 검사하여 컴포넌트를 재구성하여 새로운 어플리케이션이나 컴포넌트를 자동으로 생성해 주는 응용 프로그램 개발 프레임웍을 제안하고자 한다.

1. 서론

최근 활발히 연구가 진행되는 CBSD (Component-Based Software Development)는 소프트웨어 업계의 직면한 문제를 해결하는 패러다임으로 받아들여지고 있다. CBSD의 핵심 연구과제 중 하나는 컴포넌트 결합에 의한 응용 프로그램의 개발에 있으며 그러한 개발 환경을 지원할 수 있는 컴포넌트 인프라스트럭처의 구체적 모습을 제시하는 것이 이 분야의 현재 주된 목표이다. 현재 컴포넌트 결합과 CBSD의 구체적 개발 환경에 대한 많은 연구가 국내외적으로 진행되고 있다.

컴포넌트 결합에 대한 연구로 우선 Piccola[1], Rapide, Darwin, Aseop, Unicon, Wright, ACME등과 같은 ADL(Architdecture Description Languages)을 들 수 있는데 이들은 주로 시스템을 구성하는 소프트웨어 요소들을 components, connectors, configurations, ports, roles, 등으로 구분하고 이들의 연관관계를 정형적으로 표현해 줄 수 있는 언어를 의미한다.

그러나 ADL의 컴포넌트는 일반적인 프로그램 모듈을 의미하고 어떤 정형화된 컴포넌트 모델을 가지고 있지 않으며 시스템을 전체적인 아키텍처 레벨에서 각 요소들 간의 조합과 연관관계를 설계 검증한다. 이에 반해서 CBSD에서 추구하는 컴포넌트는 정형화된 컴포넌트 모델을 가지고 있으며 특정 응용프로그램을 생성하는 방법을 제공한다는 점에서 차별화 된다. 현재 산업계의 컴포넌트 프레임웍으로는 EJB, CCM(CORBA Component Model), COM+등을 들 수 있는데 이들 모델들은 자체적인 컴포넌트 모델을 가지고 있으며 분산 환경을 지원하는 컴포넌트 프레임웍이다. 이들 모델에서는 분산 환경에서의 개개의 컴포넌트를 쉽게 개발할 수 있는 방법을 제공하고 있지만 CBSD에서 추구하고 있는 컴포넌트간의 결합 및 재구성, 재배치 등을 허용하는 컴포넌트 지향적 응용 프레임웍으로서는 다소 미흡하다. 이외에 컴포넌트 결합과 컴포넌트 프레임웍에 대한 연구로는 XML기반의 결합언어를 제안하고 이를 이용해서 자바빈즈(JavaBeans) 컴포넌트의 재귀적을 연구한 BML(Bean Markup Language)[2], 이질적인 컴포넌트 모델을 통합하려는 Gaia[3], 웹 서비스를 결합하려는 WSFL(Web Service Flow Language)[4], ADL언어를 기반으로 소프트웨어를 재귀적으로 결합하고 동적으로 재구성하려는 Fractal Framework[5]등을 들 수 있으며 국내에서는 ETRI의 C2 아키텍처를 기반으로 EJB 컴포넌트를 재귀적으로 결합하려는 연구[6]가 진행되고 있다. 그러나 이들 연구들도 재귀적 컴포넌트 결합에 대한 일반적인 모델이 미흡하고 구체적인 재구성 가능한 컴포넌트 응용 프레임웍의 구현 방안을 제시하지 못하고 있다.

이에 본 논문에서는 결합 명세서를 기반으로 컴포넌트를 재귀적으로 결합하고 결합된 컴포넌트를 재구성 할 수 있는 응용 프레임웍을 제안하고자 한다.

2. 컴포넌트 결합

2.1 결합 명세서

컴포넌트간 상호동작 시에 각 컴포넌트의 인터페이스는 제공하는 서비스와 요구하는 서비스가 기술된다. 컴포넌트들은 결합되어서 복합 컴포넌트를 생성하게 되는데, 생성되는 복합 컴포넌트 자체가 재귀적으로 다른 컴포넌트와 결합하기 위해서는 자신이 또한 자신의 인터페이스를 외부로 내보여야 한다. 그리고, 컴포넌트 결합 시 결합될 컴포넌트의 인터페이스 중 결합에 참여하는 것은 결합하는 상대에 따라 달라질 수 있고, 컴포넌트가 결합하기 위해 필요한 상호동작 조건 또한 달라질 수 있다. 따라서, 이러한 내용들을 결합을 위한 조건으로 기술하는 것이 필요하고, 이런 내용들을 기술한 것이 결합 명세서이다[7].

결합 컴포넌트가 제공하는 서비스는 결합에 참여하는 하부 컴포넌트들이 제공하는 서비스들의 부분 집합의 개념을 가지며, 이것은 결합 명세서에 기술된 내용에 따라 바뀌게 된다. 즉, 하부 컴포넌트가 동일하더라고 결합 명세어의 작성에 따라 결합 컴포넌트의 서비스는 바뀔 수 있다. 즉, 결합 컴포넌트의 서비스와 하부 컴포넌트의 서비스와의 관계는 단순 일대일 연결 관계뿐 아니라 결합 컴포넌트의 하나의 서비스에 대하여 결합에 참여하는 컴포넌트들의 서비스들의 순서나 조합에 관계되는 것이다. 또한, 이렇게 생성된 결합 컴포넌트는 그 자체로 하나의 실행 프로그램이 될 수도 있고, 다른 결합 컴포넌트의 하위 컴포넌트로 재귀적으로 결합될 수 있다.

다음 그림 1은 결합 컴포넌트의 서비스 호출에 대하여 하부 컴포넌트 서비스로의 연결을 보여주고 있는 것으로, 하위 컴포넌트 A 와 B를 결합하여 F 와 E 라는 서비스를 제공하는 결합 컴포넌트의 한가지 형태를 보여주고 있다. 그림 2는 기본적인 결합 명세서의 문법을 보여주고 있으며, 사용자는 결합 명세서를 작성하는 것만으로도 자동으로 결합 컴포넌트를 생성할 수 있다.

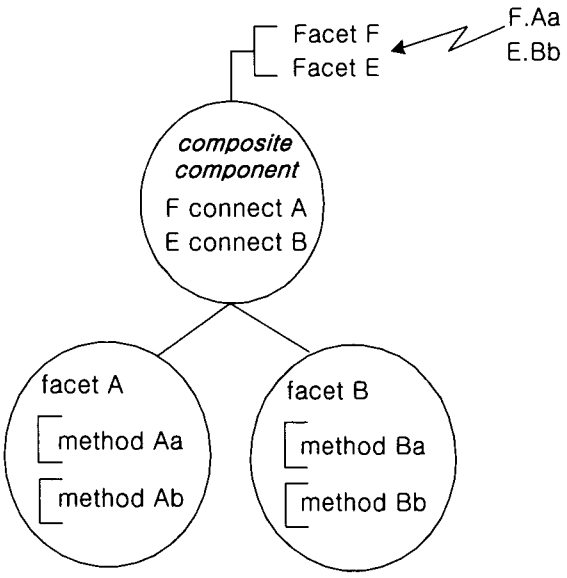

그림 1 결합 컴포넌트 서비스 호출과 연결관계

```
COMPOSITION_SPEC spec_name
{
  COMPOSITE_COMPONENT
    component_name
  {
    PROVIDE_SERVICE
        [ interface_name ]+
    REQUIRE_SERVICE
        [ interface_name ]+
  }

  IMPORT_COMPONENT
    component_name
  {
    PROVIDE_SERVICE
        [ interface_name ]+
    REQUIRE_SERVICE
        [ interface_name ]+
  } +

  CONNECTION_DESCRIPTION
    [ interface_name con
        interface_name]+
    [ interface_name con
        interface_sequence] +
}
```

그림 2 결합 명세서

▶ <composition_definition>

생성하고자 하는 결합 컴포넌트를 정의하는 것으로 결합 컴포넌트의 이름, 제공하고자 하는 서비스(<provide_service>) 목록과 요구되는 서비스(<require_service>) 목록으로 구성되며, 이 내용에 의해 필요한 <import_component> 들을 정의하게 된다.

▶ <import_component>

결합 컴포넌트를 생성하기 위하여 필요한 컴포넌트들의 목록과 각각 컴포넌트들의 제공 받을 수 있는 서비스(<require_service>) 목록과 제공해야 하는 서비스(<provide_service>) 목록으로 구성된다.

▶ <connection_description>

상호 연결되어야 하는 서비스들(<connection>)과 실행의 유기적인 순서를 정의한 서비스들의 연관관계(<sequence_list>)로 구성되며, 여기서 정의하는 서비스들은 <composition_definition> 과 <import_component>의 서비스 목록에 적절히 정의되어있어야 한다. 이 정보를 바탕으로 CCM에서 컴포넌트들의 결합에 사용되는 assembly description에 필요한 정보를 생성한다.

2.2 결합 컴포넌트 생성기(Generator)

본 논문에서는 최하위의 컴포넌트로서 CCM기반의 컴포넌트를 사용하고 있으며 이는 CCM의 컴포넌트가 타 컴포넌트 모델과 비교하여 비교적 CBSD의 컴포넌트 프로세스를 잘 따르고 있다고 판단하였기 때문이다[8].

작성된 결합 명세서를 기반으로 Generator는 결합 컴포넌트의 CCM CIDL과 구현 코드, 그리고 Composition 과 Deployment를 위한 Description을 자동으로 생성하며 결합 컴포넌트의 CIDL은 Component Category별로 CCM의 컨테이너의 전략에 따라 여러 가지 상황에 대하여 패턴화 시킬 수 있다. Generator의 구현 방법으로는 컴파일러 방식

과 인터프리터 방식을 들 수 있는데 본 논문에서는 보다 안정된 시스템을 위하여 컴파일러 방식을 제안하고자 한다. 생성된 결합 컴포넌트는 재귀적으로 다른 컴포넌트 결합에 참여 할 수 있어야 하며 결합 컴포넌트를 패키지화 했을 때 CCM의 패키지 정보와 상호 동작하여야 한다.

이를 위하여 Composition Description은 CCM의 Assembly Descriptor의 정보를 충분히 지원하며 결합 컴포넌트의 정보를 추가로 가지게 된다. 또한 입력된 CCM 컴포넌트들의 category 와 type에 따라 Component Deployment에 필요한 정보를 위하여 Deployment Description이 입력된 CCM component들의 category 와 type의 특성을 고려하여 생성되게 되며, 이 정보를 토대로 CCM의 Component Loader 기가 컴포넌트들을 적절히 배치해 줄 수 있게 된다. Composition Description 과 Deployment Description으로 결합 컴포넌트를 패키지화 할 수 있으며, 이것으로 기존의 CCM 패키지와 상호동작이 가능한 형태를 구성할 수 있게 된다.

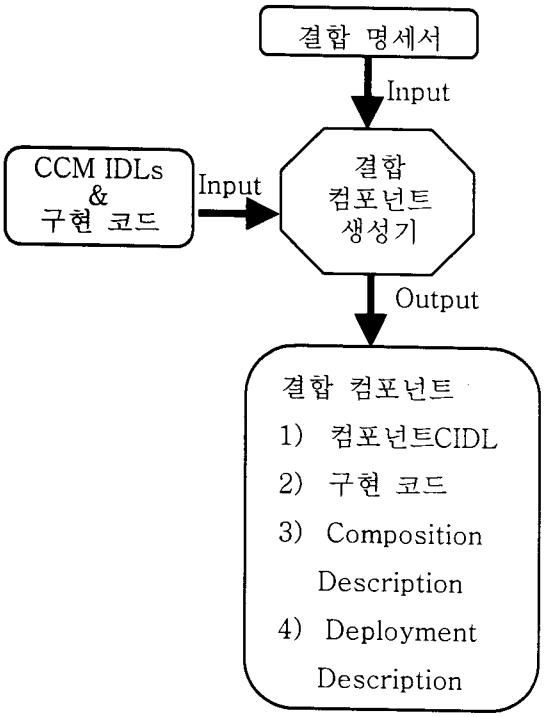

그림 4 결합 컴포넌트 구성도

결합 컴포넌트 생성기는 3부분으로 구성된다.

▶ IDL & CIDL 분석 및 생성

CCM에서는 컴포넌트 타입과 인터페이스에 대한 정의를 기술하기 위해서 CORBA 3.0에서 제시한 IDL을 사용하고 컴포넌트의 구현 환경에 대한 내용들을 기술하기 위해서 CIDL을 사용한다[8][9].

따라서, 결합 컴포넌트 생성기는 결합 하고자 하는 컴포넌트들의 IDL 과 CIDL을 분석하여, 결합 명세서에 작성된 결합 컴포넌트의 서비스의 생성이 가능하지를 검사하고, 결합 컴포넌트에 적합한 IDL 과 CIDL을 생성해 준다. 이때, 고려되어야 할 요소 중 가장 중요한 것은 결합 하고자 하는 컴포넌트들의 Category 와 Type 이다. 이것은, CCM 컴포넌트의 HOME의 구조와 Lifecycle 정책을 결정하는 요소로 Deployment description을 구성하는 요소이기도 하다.

결합 되는 컴포넌트의 조합의 경우는 다음과 같이 크게 3가지 유형으로 구분된다.

Session 컴포넌트와 Session 컴포넌트 결합

Session 컴포넌트와 Entity 컴포넌트 결합

Entity 컴포넌트와 Entity 컴포넌트 결합

결합되는 컴포넌트들 중에서 결합 컴포넌트에 정의된 서비스를 위하여 연관되는 컴포넌트들이 어떤 분류에 속하는지에 의해 결합 컴포넌트의 Home 과 Container Type이 결정되며, CIDL을 생성할 수 있다.

▶ 구현 코드 생성

결정된 IDL 과 CIDL에 의하여 결합 컴포넌트의 구현 코드가 결정되며 이것은 크게 두가지 형태로 구분된다.

1) 서비스 중계 역할을 위한 코드

하위의 결합된 컴포넌트가 제공하는 서비스의 일부가 상위의 결합 컴포넌트의 제공하는 서비스일 때 는 상/하위의 서비스의 중계역할을 하는 코드를 생성해주게 된다.

2) 서비스 결합 역할을 위한 코드

상위의 결합 컴포넌트의 요구하는 서비스가 하나 이상의 하위 컴포넌트에서 요구하는 서비

스일 경우 서비스 전달의 순서나 요구하는 하위 컴포넌트가 stateless 인지 durable 인지에 따라 서비스 상태정보의 저장을 결정하는 코드가 필요하다. 또한 결합 컴포넌트의 제공하는 서비스가 하나이상의 하위 컴포넌트의 서비스들의 조합으로 이루어진 경우 서비스 상태의 초기화 시점과 서비스 수행의 시작 조건과 만족되는 시점의 조건, 그리고 수행 순서를 제어할 수 있는 코드를 생성해준다.

▶ Composition & Deployment Description 생성

CCM에서 제안한 XML 구조의 Assembly description 과 Deployment description의 구조를 따르면서 재귀적 결합에 의해 발생하는 하위 컴포넌트의 Deployment 위치와 시점을 고려한 정보가 추가된다. 즉, 결합하고자 하는 하위 컴포넌트가 결합 컴포넌트라도 결합 컴포넌트의 명세서를 작성할 때는 일반적인 컴포넌트와 동일한 방법으로 작성된다. 하지만, 하위 컴포넌트와의 연관성을 고려하여, 결합 컴포넌트의 Deployment 정보는 하위컴포넌트의 Deployment 정보를 알고 있어야 하고, Component Loader에 의해 결합 컴포넌트가 Loading될 때 하위 컴포넌트들의 Deployment 시점과 위치가 결정될 수 있어야 한다. 따라서, 결합 컴포넌트는 일반 컴포넌트가 가져야 하는 Deployment 정보 외에 추가적으로 하위 컴포넌트들의 Deployment 상태를 제어할 수 있는 요소가 필요하게 되며, 생성기는 결합되는 하위컴포넌트들의 Description 정보에 의해 결합 컴포넌트의 Description 정보를 생성해주게 된다.

3. 컴포넌트 결합을 위한 응용 프레임웍

본 논문에서 제시하는 시스템은 크게 RCCF (Recursive Component Composition Framework)와 응용 프레임웍으로 구분된다. 그림 5는 시스템 구성요소와 상호 관계를 보여주고 있다.

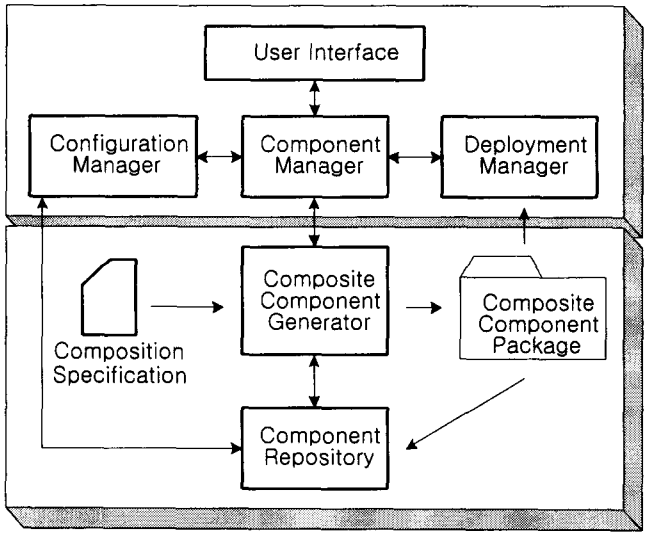

그림 5 RCCF와 응용 프레임웍

3.1 RCCF의 구성요소

▶ Composition Specification

결합 컴포넌트를 정의하는 명세서로, 결합 컴포넌트의 기능과 결합에 참여하는 컴포넌트들의 기능과 상호간의 결합 관계를 정의한 것이다.

▶ Composite Component Generator

결합 명세서에 기반하여 결합 컴포넌트를 생성하고 결합 컴포넌트와 결합에 참여한 컴포넌트를 패키지화 한다. 패키지 구성에 대한 정보와 배치에 대한 정보를 기술하는 XML 기반의 Composite Component Package Descriptor의 생성을 담당한다. 생성된 패키지 정보는 차후 재구성 시 사용하기 위해 Component Repository에 저장한다.

▶ Component Repository

결합 컴포넌트의 생성 및 결합, 배치에 필요한 모든 정보를 저장 관리하며 CCM의 단순 패키지와 어셈블리 패키지 정보를 저장 관리한다. Generator에 의해 생성된 결합 컴포넌트의 정보를 저장 관리한다. 저장된 정보를 재귀적 컴포넌트 결합 및 컴포넌트 재구성의 정보로서 사용한다.

▶ Composite component package descriptor

결합 컴포넌트 패키지 정보가 XML기반으로 기술되어 있으며 그 주된 내용은 결합 컴포넌트의 인터페이스 파일과 구현 파일, 결합에 참여하는 컴포넌트들에 대한 패키지 링크, 결합 컴포넌트와 하부 컴포넌트와 연결관계이다.

3.2 응용 프레임웍의 구성요소

▶ User Interface

결합에 참여할 컴포넌트들의 검색, 연결, 그리고 결합 명세서의 작성 등을 용의하게 할 수 있도록 도와주는 비주얼 툴을 의미한다.

▶ Component Manager

결합 컴포넌트 및 결합에 참여하는 컴포넌트의 생명 주기를 관리한다. 사용자 인터페이스로부터의 요구에 대한 작업을 수행한다. Configuration Manager 및 Deployment Manager와 연계하여 결합 컴포넌트의 관리 및 재구성을 수행한다.

▶ Configuration Manager

컴포넌트의 결합 시 Component Repository의 각 컴포넌트의 구성정보를 이용한다. 결합 컴포넌트 재구성 시 Component Repository의 정보를 이용하여 교체될 컴포넌트의 결합 성 및 적합성을 검사한다.

▶ Deployment Manager

Generator로부터 생성된 Composite Component Package Descriptor의 가상의 구성 정보를 기반으로 CCM의 deployment interface와 배치 전략을 이용하여 실제 시스템에 하나의 결합 컴포넌트의 모습으로 보이도록 배치한다. 재귀적 결합 컴포넌트를 분산환경에 배치 및 관리한다.

3.3 결합 컴포넌트 개발 및 재구성 프로세스

컴포넌트를 결합 하고자 하는 제 3자의 사용자는 결합에 참여하는 컴포넌트의 구현에 대하여 자세히 알지 못하고 잘 정의 되어 있는 인터페이스의 제공 및 요구 서비스를 이용하여 기능적 결합을 위한 결합 명세서를 기술하며 Generator를 이용하여 자동으로 결합 컴포넌트를 생성한다. Generator는 결합 컴포넌트에 대한 구현 및 인터페이스 파일 등을 생성하고 패키지화 한다. 결합 컴포넌트 패키지의 구성 요소들에 대하여 XML기반의 내부 구성 정보를 생성하고 이를 Component Repository가 저장 관리하게 된다. 생성된 결합 컴포넌트는 Deployment Manager에 의해 분산 배치 및 관리된다. 하나의 결합 컴포넌트는 외부에서 보이는 결합 컴포넌트와 내부의 결합에 참여하는 컴포넌트로 구성되어 있다. 이에 대하여 실제 시스템에 배치되었을 때 결합 컴포넌트가 내부의 구성에 관계없이 하나의 컴포넌트의 모습을 갖도록 관리 하는 것이 Deployment Manager의 주된 목적이다. 이를 위하여 Deployment Manager은 CCM의 deployment interface와 전략을 수용 및 이용하여 구현한다. 또한 Deployment Manager는 결합 컴포넌트를 재귀적으로 다른 컴포넌트와 결합하여 응용 프로그램을 작성할 때 응용 프로그램 수준에서 재귀적 결합 컴포넌트의 배치 및 관리를 수행한다. Component Manager는 사용자로부터 요구를 수용하여 결합 컴포넌트 작성에 필요한 정보를 Component Repository로부터 요구하고 이를 사용자에게 제공하게 된다. 결합 컴포넌트를 새롭게 재구성하고자 할 경우 Configuration Manager는 결합 컴포넌트의 결합 구성에 대한 정보를 Component Repository로부터 조회하고 검사하여 재구성을 관리한다. 재구성의 방법을 그 정도에 따라 정적, 사용자의 개입, 또는 동적으로 처리할 수 있다. 사용자 인터페이스를 비주얼 툴의 모습으로 구현하고 보다 용이하게 컴포넌트를 결합 및 재구성할 수 있도록 한다.

4. 결론 및 향후 계획

본 논문에서는 결합에 참여하는 컴포넌트 간 기

능적 결합에 기반을 둔 결합 명세서를 제안하였으며 결합 명세서에 기반하여 자동으로 결합 컴포넌트를 생성해주는 생성기를 구현하였다. 그리고 이에 의해 생성된 결합 컴포넌트를 다시 결합에 사용하는 방안에 대하여 제안 함으로서 기존의 컴포넌트 기반 프로그램 개발환경에서 부족하다고 지적되어온 컴포넌트 결합 방법을 보완해 줄 수 있을 것으로 기대된다. 또한 본 논문에서는 시스템의 성장에 따라 결합 컴포넌트를 재구성하고자 할 때 구성 컴포넌트의 적합성 및 결합 성을 검사하여 컴포넌트를 재구성할 수 있는 프레임웍을 제안 함으로서 CBSD 의 결합에 의한 일반적인 컴포넌트 개발 프로세스를 충분히 만족시키고자 하였다.

이러한 결합 명세서를 이용하여 개발한 응용 프로그램의 장점은 명세와 구현이 서로 독립적이어서 응용 프로그램 개발자 입장에서 컴포넌트의 결합만으로 쉽게 응용 프로그램 개발이 가능하다.

반면에 컴포넌트 개발자의 개발 비용이 커지는 단점이 있다. 자신이 개발하는 컴포넌트가 추후에 응용 프로그램에서 이용 될 때 사용될 제공 서비스를 추측하여 개발해야 하므로 디자인 시 고려할 사항이 많아지게 된다.

향후 본 논문에서 제안한 결합 명세서의 기술 구조를 더 개선하여 단지 CCM 에서뿐만 아니라 현존하는 대부분의 컴포넌트 기반 시스템에서 공통으로 사용할 수 있도록 할 필요가 있으며, 추가적으로 필요한 결합 정보에 대한 연구가 진행 중이다. 차후에는 결합되는 컴포넌트들간의 상호동작 조건을 기술하는 방법, 실행 시에 컴포넌트의 교체가 가능해질 수 있는 방법에 대하여 연구를 수행하여, 제안한 모델에 대하여 도메인을 설정하여 시스템을 구현 적용하여 검증하고자 한다.

[참고문헌]

[1] Franz Archermann, Markus Lumpe, Jean-Guy Schneider, and Oscar Nierstrasz. Piccola - a samll composition language. In Howard Bowman and John Derrick.(Eds.), editors, Formal Methods for Distributed Processing, an Object Oriented Approach. 2000.

[2] A. Beugnard, J-M Jezequel, and D.Watkins. Making Components Contract Aware. IEEE Computer, July 1999

[3] Francisco Curbera, Sanfiva Weerawarana, Matthew J. Duftler, On Component Composition Language , IBM T.J. Watson Research Center, 2000

[4] IBM Corporation, Web Services Flow Language (WSFL 1.0), available at http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf

[5] Bruneton, T. Coupaye, and J.-B. Stefani. Recursive and dynamic software composition with sharing. In Proceedings of the 7th ECOOP International Workshop on Component-Oriented Programming (WCOP'02), Malaga (Spain), June 2002.

[6] You-Hee Choi, Oh-Cheon Kwon, Gyu-Sang Shin. An Approach to Composition of EJB Components Using C2 style, Proceedings of the 28th Euromicro Conference(EUROMICRO'01), 2002

[7] 백경원, 박성은, 이정태, 류기열, "컴포넌트 결합 명세서에 기반 한 컴포넌트 결합 모델", 정보처리학회논문지 D 특집호, 컴포넌트 모델링 & 아키텍처 , 2001

[8] OMG, "CORBA Components", OMG TC Document orbos/02-06-65, June, 2002

[9] OMG, "CORBA 3.0 New Component Chapters", OMG TC Document orbos/2001-11-03, November, 2001