인터페이스 명세기반 컴포넌트 저장소 모델

김태웅 김경민 김태공
인제대학교 전자계산학과
(twkim, kmkim, ktg)@cs.inje.ac.kr

A Model of Component Repository Based on Interface Specification

Tae-Woong Kim Kyung-Min Kim Tae-Gong Kim
Department of Computer Science, Inje University

요 약

객체기술의 확장, 분산처리기술의 발전과 더불어 주목받는 컴포넌트 기반 소프트웨어 구축기술은 기존의 프로그래밍 기술을 대체하는 효율적인 기법으로 평가받고 있다. 소프트웨어의 재사용 측면에서 이러한 컴포넌트들은 프로그램 개발의 생산성 증대에 기여하는 것은 사실이지만 이러한 효과가 현실화되기 위해서는 기본적인 제반요소 -검색 및 저장소, 효과적인 컴포넌트 서술- 가 해결되어야 한다.

이에 본 논문에서는 컴포넌트의 명세를 기반으로 한 컴포넌트 저장소 모델을 제안 한다. 이러한 컴포넌트의 명세는 인터페이스 기술에 대한 명세, 상속등과 같은 재사용에 대한 정보, 인터페이스들을 포함하는 컴포넌트 명세들 간의 상호작용에 관한 정보를 포함한다.

1. 서 론

최근 들어 소프트웨어의 대형화, 통합화가 요구되어지면서 소프트웨어 개발방법의 변화가 대두되고 있으며, 그 해결책의 하나로 컴포넌트 소프트웨어가 제시되고 있다. 경험이 많은 개발자에 의해 개발된 컴포넌트는 소프트웨어 재사용이 뛰어나고, 이미 많은 곳에서 사용 중이므로 안정성 및 신뢰성이 인정된다.

소프트웨어 컴포넌트는 꼭 실행 가능한 바이너리 코드만을 말하는 것이 아니라 소프트웨어 개발 시 발생하는 산출물-분석 및 설계, 클래스, 문서, 도움말등-들을 재사용 가능한 형태로 만들어 놓은 모든 것들을 포함한다.

컴포넌트는 재사용을 위해 다른 소프트웨어나 컴포넌트와 연결할 수 있는 방법을 정의하고 있는 인터페이스(Interface)를 가진다[1]. 컴포넌트가 수행 가능한 기능을 컴포넌트 외부에서 사용가능 하도록 하는 것이다.

소프트웨어 컴포넌트는 다른 것들과 명확히 구별(identifiable) 되어야 하며, 다른 컴포넌트나 어플리케이션에 포함된 후에 그 존재를 확인(traceable)할 수 있어야 하고, 어플리케이션 사용에 영향을 주지 않고 다른 버전의 컴포넌트나, 동일한 서비스나 기능을 제공하는 다른 컴포넌트로 교체(replaceable)할 수 있어야 한다[2]. 또한 문서로 정확히 기록(Accurately Documented Service)되어 재사용을 위해서 인터페이스를 통해 제공되는 서비스에 대해 정확한 기록을 남겨야 하는데, 이를 통해 서비스를 다루는 방식 뿐 아니라 어떠한 서비스가 제공되는지에 대해서도 이해할 수 있도록 해야 한다. 이러한 일련의 과정이 이루어지기 위해서 선행되어야 할 과제가 컴포넌트 저장소이다. 그러나 현재 연구, 개발되고 있는 컴포넌트 저장소는 인터페이스에 대한 명세를 간과한 채 기능중심의 컴포넌트 저장소가 대부분[3,4]이며 인터페이스에 대한 명세를 포함하더라도 컴포넌트들 간의 상호작용에 관한 조건은 잘 표현하지 못하고 있다[5]. 다양한 컴포넌트의 결합을 제공하기 위해서는 컴포넌트 명세부분만을 이용하여 컴포넌트들을 결합할 수 있어야 한다[5].

이에 본 논문에서는 인터페이스 명세를 중심으로 한 컴포넌트 저장소 모델을 제안한다. 이 저장소는 명세의 상속등과 같은 재사용에 대한 정보, 인터페이스들을 포함하는 컴포넌트들 간의 결합관계를 나타내는 상호작용에 관한 정보를 포함한다.

2장에서는 본 연구의 컴포넌트 저장소 모델에 대해 제안하고 3장에는 컴포넌트 명세 적용 사례를 들고 4장에서는 결론을 맺는다.

2. 컴포넌트 저장소

개발자들이 개발한 컴포넌트를 사용자가 쉽게 검색하고 사용 가능하도록 컴포넌트를 명확히 명세화[6] 하여 저장한곳이 컴포넌트 저장소이다.

현재 CORBA에서는 객체를 표현하고 메쏘드를 노출하기 위한 방법으로 IDL(Interface Definition Language)을 사용한 인터페이스 저장소를 제공한다. 이는 객체에 대한 구문적인 표현을 정형적으로 나타낸 명세서라고 볼 수 있다. 그러나 IDL은 구문적인 정합성만을 가지기 때문에 컴포넌트가 가져야 하는 컴포넌트 결합인 상호작용에 대한 정보가 포함되어 있지 않다.

따라서 본 연구에서는 컴포넌트를 기술하기 위한 방법으로 컴포넌트의 인터페이스에 대한 명세를 기존의 방법인 CORBA IDL형식[7]에 따라 기술하고 이를 확장하여 컴포넌트의 상호작용 및 재사용에 필요한 정보를 추가하고, 이러한 정보는 상호작용 중심의 인터페이스를 표현하기 위한 UML[8]을 확장하여 기술 하였다.

하나의 컴포넌트를 명세 하는데 필요한 항목은 컴포넌트의 분류, 기능 및 부가설명을 포함하는 컴포넌트 정보,컴포넌트의 서비스를 기술할 인터페이스 정보, 컴포넌트 명세 상속과 상호작용에 관한 정보로 이루어진다.

2.1 컴포넌트 정보

명세의 분류는 다양한 컴포넌트 분류법[9]을 지원하도록 되어 있으며 하나의 명세는 다양한 분류법을 동시에 가질 수 있다. 컴포넌트는 유일한 명칭을 가지며 해당 컴포넌트의 기능을 설명할 항목은 사용자 정의 속성을 정의하도록 하여 컴포넌트를 설명하는데 유동성을 확보하도록 하였다.

Classification

ID	ClassificationName	Content	TableName
cl0001	NAICS	산업분류 시.	T_NAICS
cl0002	UNSPSC	표준 서비스.	T_UNSPSC

FlexibleClassification

ID	SpecID	Classification ID	CategoryID
fl0001	sp0001	cl0003	ca0001
fl0002	sp0002	cl0001	ca0002

Category

ID	CategoryName	Parent	Degree	Difference	Content
ca0001	전자계산기	0	0	0	전자계산기.
ca0002	운영체제	0	0	0	운영체제는 .

그림 1 동시 분류법을 지원하기 위한 데이터베이스 스키마

2.2 인터페이스 정보

각 인터페이스의 모든 정보를 가진다. 인터페이스 명칭, 간단한 기능설명을 포함하며, 각각의 인터페이스는 메소드, 속성, pre와 post 조건을 가진다. 그림 2에서와 같이 CORBA IDL형식을 확장하여 표현함으로써 가독성을 높여준다.

2.3 명세 상속 및 상호작용

컴포넌트는 다른 컴포넌트들과 결합되어 사용되어 지는 경우가 대부분이며, 명세도 다른 명세와의 상속 및 상호작용을 가진다. 그림 3와 같이 본 연구에서 제안한 확장된 IDL을 사용하여 명세의 상속과 상호작용을 표현하였으며 그림 4와 같은 UML로도 이와 같은 관계를 표현하여 컴포넌트 명세에 포함하였다.

```
Component Spec unique-component-name {
provide intreface interface-name {
    method-name((in,out,inout) type parameters);
    (* pre and post conditions *)
}
provide intreface interface-name {
        method-name((in,out,inout) type parameters);
    (* pre and post Conditions *)
}
```

그림 2 IDL 형식의 인터페이스 명세

```
Component Spec component-name {
  privide interface interface-name {
    method-name((in,out,inout) type parameters);
     (* pre and post conditions *)
    method-name((in,out,inout) type parameters);
     (*pre and post conditions *)
  }
  required interface interface-name :: Component Spec
           related component-name {
    method-name((in,out,inout) type parameters);
  }
}

Component Spec component-name {
  provide interface interface-name {
    method-name((in,out,inout) type parameters);
     (*pre and post conditions *)
  }
}

Component Spec component-name : Component Spec
inherited component-name {
  provide interface interface-name {
    method-name((in,out,inout) type parameters);
     (* pre and post Conditions *)
  }
}
```

그림 3 컴포넌트들간의 상호작용에 대한 명세

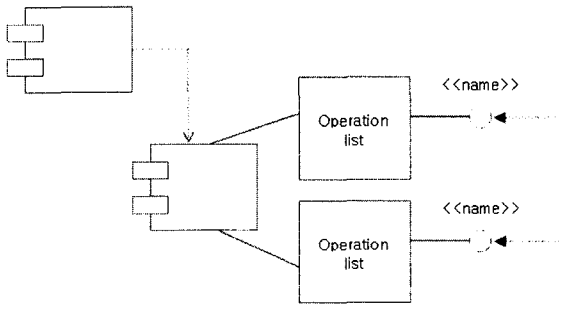

그림 4 확장된 UML을 이용한 명세 상속 및 상호작용

3. 컴포넌트 명세 적용 사례

본 연구에서는 본 연구에서 제안한 시스템을 개발하기 위한 컴포넌트의 명세중 XML과 데이터베이스의 상호작용을 예제로 작성하였다.

```
Component Spec XML_Data_Manager : Component Spec
XML_Manager{
   provide interface I_TransXML {
          void setXMLQuery(in string query);
          getRecordset(in string XMLquery);
          getXMLResult(in recordset result);
          void setDBQuery(in string query);
   }
   provide interface I_ProsessXML {
          DBConnect(in string procName);
          makeXMLDoc(string XMLDoc);
   }
}
Component Spec Data_Manager {
   required interface I_TransXML :: Component Spec
XML_Data_Manager {
          void setXMLQuery(in string query);
          getRecordset(in string XMLquery);
          getXMLResult(in recordset result);
          void setDBQuery(in string query);
   }
   provide interface I_ProsessData {
          makeXMLQuery(in string Query);
          getXMLResult(in string XMLDoc);
   }
}
Component Spec DataBase_Manager {
   required interface I_ProsessXML :: Component Spec
XML_Data_Manager {
          DBConnect(in string procName);
          makeXMLDoc(string XMLDoc);
   }
}
Component Spec XML_Manager {
   provide interface I_Manager {
          makeXMLDoc(string XMLDoc);
                  :
   }
}
```

그림 5 컴포넌트들간의 상호작용에 대한 명세

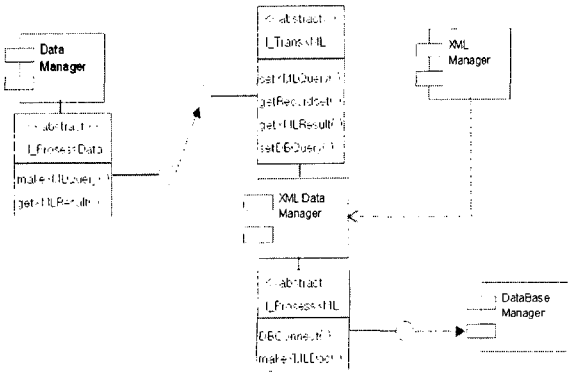

그림 6 확장된 UML을 이용한 명세 상속 및 상호작용

4. 결론 및 향후연구

소프트웨어의 대형화, 통합화 추세에 따라 개발비용 및 유지보수비용의 증가 문제를 해결하고자 컴포넌트 소프트웨어를 이용하는 추세이다. 이에 맞추어 본 논문에서는 인터페이스 명세를 중심으로 한 컴포넌트 저장소를 제안한다.

본 연구에서 제안한 컴포넌트 저장소의 특징은 컴포넌트의 인터페이스 명세에 초점을 맞추어 이를 재사용하고 명세 관점에서 컴포넌트의 상호작용을 쉽게 파악할 수 있다. 이와 같이 명세의 등록과 재사용은 사용자의 요구분석 사항이 하나의 명세로 등록이 가능하며, 이는 개발자와 사용자간의 공동 작업영역을 의미하기도 한다. 개발자는 유저인터페이스를 통해 요구사항을 등록하면 개발자는 자동생성 되어진 확장된 가독성 높은 IDL 형식을 사용하여 컴포넌트를 개발한다. 본 연구에선 제안한 컴포넌트 명세 저장소의 구성요소는 컴포넌트 정보에 대한 명세와 하나 이상의 인터페이스에 대한 정보, 컴포넌트 명세들간의 상속과 같은 재사용 및 상호작용에 대한 정보가 하나의 컴포넌트에 대한 명세로 이루어져 있다.

이러한 컴포넌트에 대한 명세 정보는 향후 컴포넌트 명세에 기반한 다양한 플랫폼과 아키텍처를 지원하는 실행 가능한 컴포넌트를 등록하여 하나의 명세에 따르는 여러 가지 인스턴스 형태의 컴포넌트를 등록하고 검색하여 사용자가 쉽게 컴포넌트에 접하고 실행할 수 있도록 하는 연구의 기반이 된다.

[참고문헌]

[1] Dedmond F.D'Souza, Alan C. Wills, "Object, Component and Frameworks With UML", ADDISON-WESLEY, 1998

[2] Butlerforums, "Application Delivery and Integration Using Componentised Software", http://www.butlerforums.com/cbdindex.htm

[3] J. Han, "An Approach to Software Component Specification", Proceedings of 1999 International Workshop on CBSE, Los Angeles, 1999

[4] Sherif Yacoub Ammar, "A Model for Classifying Component Interface", Proceedings of 1999 International Workshop on CBSE, Los Angeles, 1999

[5] 백경원, "컴포넌트 결합 명세서에 기반한 컴포넌트 결합 모델", 정보처리학회논문지 제8-D권 제6호, 2001.12

[6] Jun Han, " An Approach to Software Component Specfication", IWCBSE, p97-p102, May 1999

[7] Jon Siegel, "CORBA Fundamentals and Programming", Wiley, 1996

[8] 박성호, "상호작용 중심의 컴포넌트 인터페이스를 표현하기 위한 UML확장", 정보처리학회논문지 제9-D권 제 1호, 2002.2

[9]Componentsource,"http://www.componentsource.com/BuyComponents/ProductCatalog/default.asp"