2001년 한국정보처리학회 추계 학술발표논문집 제8권 제2호

컴포넌트간의 의존관계 모델 분류

°채정화*, 유철중*, 장옥배*
*전북대학교 컴퓨터과학과
jhchae@cs.chonbuk.ac.kr
{cjyoo, okjang}@moak.chonbuk.ac.kr

On Component Dependency Model

°Jung-Hwa Chae*, Cheol-Jung Yoo*, Ok-Bae Chang*
*Dept. of Computer Science, Chonbuk National University

요 약

컴포넌트는 다수의 구현물들이 통합되어 요구되는 기능을 수행한다. 이렇게 다른 컴포넌트들과 공존하며 주어진 기능을 수행하기 위해서 컴포넌트간의 통신은 필수적이며 그들간의 의존관계는 반드시 존재하게 된다. 본 논문에서는 효율적인 컴포넌트의 통합 및 관리를 위하여 비즈니스 컴포넌트의 의존관계 모델을 분류한다. 비즈니스 컴포넌트의 의존관계를 범주화하기 위하여 비즈니스 컴포넌트 인터페이스의 유형을 살펴보고, 컴포넌트의 개념도 및 시스템 개발 라이프사이클의 핵심 단계에서 비즈니스 컴포넌트 관점을 도출하여 이것을 기반으로 비즈니스 컴포넌트 의존관계 모델을 제안한다. 본 논문에서 제시한 의존 관계 모델은 컴포넌트 기반 시스템이 대형화되고 복잡도가 증가함에 따라 이러한 시스템을 개발하는데 있어서 컴포넌트간의 의존 관계를 명확히 파악하고 그에 대한 적절한 관리를 할 수 있도록 하는데서 그 의의를 찾을 수 있다.

1. 서론

컴포넌트는 단일의 구현물로서가 아니라 다수의 구현물들이 통합되어 필요한 기능을 수행한다. 이렇게 다른 컴포넌트들과 공존하며 요구되는 기능을 수행하기 위해서 컴포넌트간의 통신은 필수적이며 그들간의 의존관계가 반드시 존재하게 된다. 컴포넌트 기반 소프트웨어 시스템이 대형화되고 복잡도가 증가함에 따라 시스템 내에서 상호 작용하는 다양한 컴포넌트의 의존관계를 식별하고 그에 대한 적절한 관리를 하는 것은 매우 중요하다[1].

오늘날의 주요 관심은 상황이 계속 변하고 있으며, 특히 소프트웨어 개발에서 중요한 것은 어떻게 변화에 대응하고 그것을 관리할 것인가 하는 문제이다.[2] 이것은 아키텍처 단계와 설계 단계 동안 주요한 관심사를 컴포넌트간의 의존성과 그 의존성을 관리하는 것이 됨으로써 변화에 잘 대응할 수 있는 시스템을 구축한다는 의미이다.

본 논문에서는 효율적인 컴포넌트의 통합 및 관리를 위하여 비즈니스 컴포넌트의 의존관계 모델을 제안한다. 비즈니스 컴포넌트의 의존관계를 범주화하기 위하여 먼저 비즈니스 컴포넌트 인터페이스의 유형을 살펴보고 컴포넌트의 개념도 및 시스템 개발 라이프사이클의 단계에서 핵심적인 비즈니스 컴포넌트 관점을 도출하여 이것들을 기반으로 비즈니스 컴포넌트 의존관계 모델을 제시한다.

본 논문의 구성은 2장에서 관련연구를 기술하고, 3장에서는 비즈니스 컴포넌트 정의 및 개발 라이프사이클 단계에서의 비즈니스 컴포넌트 관점을 도출한다. 4장에서는 비즈니스 컴포넌트 의존관계 모델을 제시하며, 5장에서는 결론을 기술한다.

2. 관련연구

구현 단계에서의 소프트웨어 시스템은 하나 이상

의 프로그래밍 언어로 구현된 모듈들의 집합이다. 대부분의 프로그래밍 언어는 모듈간의 상호작용 메커니즘을 직접적으로 지원한다. 그러나 그와 같은 메커니즘은 오늘날의 소프트웨어 컴포넌트 시스템에서 일반적으로 발생하는 복잡한 의존관계를 관리하기에는 충분하지 않다.

기존의 설계 및 프로그래밍 툴들은 컴포넌트의 의존 문제를 컴포넌트의 명세나 구현과 분리하여 설계 문제로서 인식하지 못하고 있는 실정이다[1]. 컴포넌트간의 의존관계의 특성 및 중요성은 시스템의 구조적 서술과 같은 상위 수준에서는 비교적 잘 나타나고 있지만 그러한 의존관계에 대한 서술이 일반적으로 비정형적인 수준이거나 구현 단계의 개념으로 변환하기에는 불명확하다. 또한 설계 단계에서 구현 단계로 갈수록 컴포넌트간의 상호작용 및 시스템의 독립되고 분산된 컴포넌트를 관리하는 프로토콜의 구현에 대한 언급은 하지 않은 채 컴포넌트 그 자체에만 초점을 둔다.

컴포넌트 상호 작용 등과 관련된 기존의 유사 연구들을 살펴보면 다음과 같다.

[4]에서는 컴포넌트의 핵심 기능과 컴포넌트간의 통신 메커니즘을 분리하기 위하여 포트(port)와 링크(link) 개념을 이용하였고, 컴포넌트를 추상화한 소프트웨어 아키텍처를 네 가지의 기본 동작 유형으로 분류하였다. [1]은 컴포넌트간의 상호의존관계를 크게 Flow 의존관계, Sharing 의존관계, Timing 의존관계로 분류하고 이들 유형별로 상호작용 프로토콜을 제시하였다. [1]에서 제시한 의존관계 분류 수준은 일반적인 소프트웨어 상호작용 관계를 정의한 것으로서 소프트웨어의 기본적인 의존 유형을 나타내고 있다. 본 연구는 컴포넌트 기반 소프트웨어의 특성을 기반으로 하는 의존관계 모델을 제시한다.

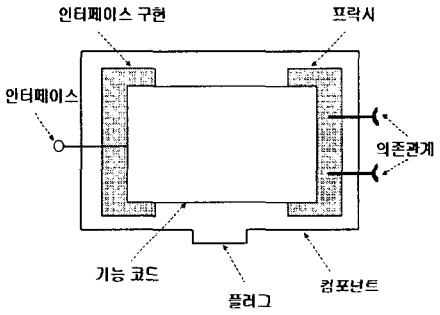
3. 컴포넌트의 개발 단계별 관점

3.1 비즈니스 컴포넌트의 정의

컴포넌트는 물리적, 논리적 장치에서 작동하는 소프트웨어 구현물로서 컴포넌트의 설계, 구현, 환경설정 관리 및 대체 등이 동시에 이루어지고, 하나 이상의 인터페이스를 구현하며 준수하는 하나의 구분되는 단위이다[4].

본 논문에서 대상으로 하는 컴포넌트는 비즈니스 컴포넌트로서 이것은 정보시스템에 비즈니스 개념을 적절히 표현하는 엔터프라이즈 컴포넌트(enterprise component)이다. 비즈니스 컴포넌트는 전체 개발 라이프사이클 동안 인터페이스 개념을 통하여 진정한 의미의 컴포넌트 역할을 한다. 확장된 인터페이스 개념을 통해서 컴포넌트의 인터페이스가 개발 단계나 실행 단계에서 고려될 수 있다. 전통적인 의미의 인터페이스는 오직 행위와 서비스만을 포함한다. 그러나 확장된 인터페이스 개념은 오퍼레이션 뿐만 아니라 모든 소프트웨어 구현물의 집합을 포함한다.

비즈니스 컴포넌트를 블랙박스로 생각하면 명세서, 모델, 소스코드, 설정파일 및 설계, 구현, 테스트, 배치 시에 필요한 모든 것을 포함할 수 있다. 이러한 것들은 오직 비즈니스 컴포넌트의 소유자에게만 보여진다. 개발 라이프사이클 동안 누군가 그것을 사용하고자 한다면 사용할 수 있는 허가권을 얻어야 하는데, 그것이 바로 접근 포인트이며 그와 같은 허가는 극히 제한되어 있다.



[그림 1] 비즈니스 컴포넌트

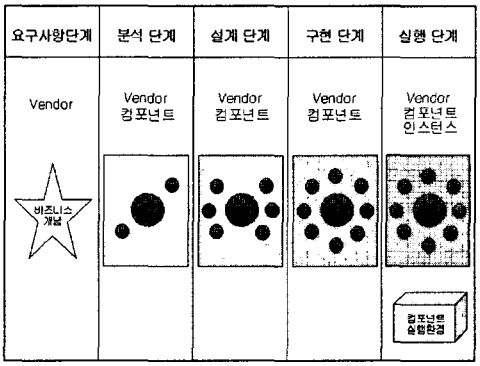
컴포넌트의 확장된 인터페이스는 컴포넌트에 대한 접근 포인트의 집합을 서술한다. 컴포넌트에 대한 접근 포인트는 아키텍처 관점에서의 컴포넌트에 대한 개발 라이프사이클의 모든 단계에서 컴포넌트에 포함되는 소프트웨어 구현물이다. 즉, 비즈니스 컴포넌트의 확장된 인터페이스는 그것을 구성하는 분산 컴포넌트 및 비즈니스 컴포넌트 자체의 모든 시스템 내부와 시스템 외부 인터페이스 집합을 말한다.

비즈니스 컴포넌트는 비즈니스에 의해 필요한 기능을 수행하기 위하여 컴포넌트간에 최소한의 의존관계를 갖으며 상호 협력하게 된다. 하나의 컴포넌트는 다른 컴포넌트에게 인터페이스를 제공하기도 하고, 또 다른 컴포넌트와 의존성을 갖기도 한다[그림 1]. 이러한 의존관계는 임포트(import)/엑스포트(export) 메커니즘을 통해서 가능하다.

3.2 개발 라이프사이클 단계별 비즈니스 컴포넌트 관점

비즈니스 컴포넌트의 가장 강력한 개념 중 하나는

요구사항 분석 단계에서부터 배치 단계의 전체 개발 라이프사이클 동안 정제과정을 거치지만 최소한의 변형으로 개발이 이루어진다는 것이다[1]. [그림 2]는 분석 및 설계 단계의 비즈니스 컴포넌트가 그 초기 개념을 유지하면서 다음 단계에서 정제되며 세분화되는 것을 나타내고 있다.



[그림 2] 컴포넌트 추적성(traceability)

비즈니스 컴포넌트의 접근 방법은 독립적이며 요구 분석 단계에서부터 시스템 배치, 실행 시간 및 진화 단계에 이르기까지 최소한의 의존관계를 갖으며 상호 협력함을 가정한다. 즉, 의존관계는 실행시간 뿐만 아니라 전체 소프트웨어 개발 라이프사이클 및 배포 후까지 계속된다. [1]은 3.1절의 확장된 인터페이스 개념을 적용하여 비즈니스 컴포넌트 인터페이스를 다음과 같이 분류하였다.

- 디자인타임 인터페이스
- 빌드타임 인터페이스
- 런타임 인터페이스

본 논문에서는 이러한 특성을 바탕으로 설계, 개발, 실행 단계에서 각각 설계 관점, 구현 관점 및 배치 관점으로 분류한다. [표 1]은 설계 관점, 구현 관점, 배치 관점의 각 단계에서 필요한 요소를 나타낸 것이다.

● 설계 관점

설계단계에서는 엑스포트되고 임포트되는 비즈니스 컴포넌트의 모든 가공물에 대한 상세 정의 및 비즈니스 컴포넌트 자체에 대한 상세 명세서를 생성한다. 이러한 명세서에는 사용자 인터페이스 프로토타입과 명세서, 비즈니스 컴포넌트의 각 계층에 대한 기본 설계, 다른 비즈니스 컴포넌트와의 의존관계 등이 포함된다. 설계 관점은 컴포넌트의 인터페이스와 인터페이스의 상호작용 관계 및 다른 컴포넌트와의 관계 등이 주요 쟁점이다. 이것은 시스템에 대한 전통적인 객체 지향 관점으로써 독립된 단위 요소들이 어떻게 컴포넌트로 구성되며 컴포넌트들이 내부의 어떤 계층에 분류되는가 등을 주로 다루게 된다.

관 점	요 소
설계관점	• 인터페이스 • 컴포넌트 간의 상호작용 • 컴포넌트 간의 관계 • 기타 문서
구현관점	• 파일 • 소스 코드 • 링크 라이브러리 • 디렉토리 • 컴파일 및 링크 관계 • 클래스 • 클래스 및 인터페이스 간의 구현 • DB 테이블 및 파일 레코드 • 공유 기억장치 구조 • 기타 문서
배치관점	• 목적 코드 • 실행 코드 • DDL • 바이트 코드 파일 • 런타임 환경설정/파라미터 파일 • 기타 문서

[표 1] 각 관점별 필요한 요소

● 구현 관점

구현 단계에서는 비즈니스 컴포넌트 내부의 실제 코딩 작업이 이루어진다. 구현시간 배포물에서 가장 중요한 것은 하위 컴포넌트 인터페이스 및 랭귀지 클래스에 대한 프록시(proxy)를 포함하는 것이다. 또한 하나의 컴포넌트에 포함되면서 내부 구현을 위해 재사용 되어 의미적으로 다른 컴포넌트에 내포되는 각종 소프트웨어 가공물을 포함한다.

● 배치 관점

배치 관점에서의 비즈니스 컴포넌트는 다양한 클라이언트 머신에서 실행 가능한 집합, 하나 이상의 서버에서 실행 가능한 집합, 초기화된 데이터를 갖는 데이터베이스 테이블 집합 등으로 나타난다. 기타 배포물로는 환경설정 파일, 사용자 가이드, 도움말 파일, 설치 가이드, 비즈니스 컴포넌트의 사용, 실행 및 관리시 필요한 다른 기타 가공물 등이 있다. 이러한 소프트웨어 가공물이 아닌 것들 또한 의존관계를 형성한다. 실행 관점은 배치된 컴포넌트가 목적하는 기능을 적절히 수행하기 위해 어떻게 서로

협력하는가가 주요 관심 대상이다.

4. 컴포넌트 의존관계 모델

본 논문에서는 3장에서 설명한 각각의 단계별 관점에 따라 개발 라이프사이클의 각 단계에서의 다양한 유형의 인터페이스와 각 단계별 관점, 즉 설계 관점, 구현 관점, 배치 관점의 특징 및 가공물 등을 기반으로 컴포넌트간의 의존관계를 범주화하였다.

컴포넌트 기반 전체 개발 프로세스는 일반적으로 요구사항 정의, 컴포넌트 명세 작성, 컴포넌트 공급, 컴포넌트 조립, 테스트, 배포 과정을 거친다.

● 설계 레벨 의존관계

컴포넌트의 설계시 고려되는 사항으로써 디자인타임 인터페이스의 의존관계가 여기에 속한다고 볼 수 있다. 컴포넌트 B의 다양한 가공물은 컴포넌트 A의 내부 설계를 위해 재사용될 수 있는데, 실제로 이러한 가공물은 컴포넌트 B의 런타임 인터페이스 형태로 나타나지 않고 다른 컴포넌트 내에서만 사용된다. 즉, 클래스를 이용하려는 컴포넌트는 그 컴포넌트 내에 클래스들을 두고 재사용 함으로써 다른 컴포넌트와 상호작용을 용이하게 할 수 있는데 이러한 관계를 설계 레벨 의존관계로 분류한다.

● 인터페이스 명세 레벨 의존관계

인터페이스 레벨 의존관계는 구현 단계에서 나타나는 의존관계가 아니고 컴포넌트 A의 인터페이스를 명시하기 위해 나타나는 의존관계를 말한다. 컴포넌트 A의 인터페이스를 정의하기 위해서는 컴포넌트 B의 가공물을 재사용할 필요가 있다.

● 빌드 타임 의존관계

레파지토리에 정의된 명세서로부터 생성된 파일들간에 의존관계가 컴파일 시간에 형성되는 경우가 있는데 이것을 빌드 타임 의존관계로 분류한다.
다른 컴포넌트의 비즈니스 데이터 타입을 사용하기 위해서는 비즈니스 데이터 타입의 선언이 먼저 이루어져야 한다. 이렇게 함으로써 선언될 파일을 소유하고 있는 컴포넌트 및 선언된 파일 그 자체에도 의존관계를 이루게 된다. 즉, 빌드 타임 의존관계는 파일간에 형성되는 의존관계를 말한다.

● 실행 레벨 의존관계

3장에서 설명한 런타임 인터페이스의 의존관계가 실행 레벨 의존관계에 포함된다. 즉, 컴포넌트 A가 컴포넌트 B를 호출하고 이에 대해 접근하기 위해서 컴포넌트 B의 프락시, 비즈니스 데이터 타입, 에러 정의 등과 같은 소프트웨어 가공물을 알고 있어야 하는 관계를 말한다.

5. 결론

컴포넌트 기반 시스템이 대형화되고 복잡도가 증가함에 따라 컴포넌트간의 의존관계를 명확히 파악하고 그에 대한 적절한 관리를 하는 것은 매우 중요하다. 본 논문에서는 효율적인 컴포넌트의 통합 및 관리를 위하여 비즈니스 컴포넌트의 의존관계 모델을 설계 레벨 의존관계, 인터페이스 명세 레벨 의존관계, 빌드 타임 의존관계, 실행 레벨 의존관계로 분류하였다. 향후 연구로는 컴포넌트 명세단계의 컴포넌트 의존관계 분석시 본 논문에서 분류한 컴포넌트 의존관계 모델을 적용하여 보고, 나아가서 디자인 패턴을 이용하여 컴포넌트 의존관계를 분석하는 것이다.

참고문헌

[1] Chrysanthos Dellarocas, "Toward a Design Handbook for Integrating Software Components", International Symposium on Assessment of Software Tools and Technologies, June 1997.

[2] John Cheesman, John Daniels, "UML Components", Addison-Wesley, 2001.

[3] G. Wang, L. Ungar and D. Klawitter, "Component Assembly for OO Distributed Systems", *IEEE Computer*, Vol. 32, No. 7, Jul 1999, pp.71-78.

[4] Peter Herzum, Oliver Sims, "Business Component Factory", OMG Press, 2000.