2000년 한국정보처리학회 추계 학술발표논문집 제 7 권 제 2 호

객체지향 프로그램에서의 컴포넌트 추출

윤 석진, 신 규상
한국전자통신연구원
e-mail : {sjyoon, gsshin}@etri.re.kr

Extraction Components from Object-Oriented Programs

Seok-Jin Yoon, Gyu-Sang Shin
Software Engineering Department
ETRI-Computer & Software Technology Laboratory

요 약

본 논문은 기존의 객체지향 방식으로 작성된 프로그램에서 컴포넌트를 추출하기 위한 제안한다. 객체지향 프로그램을 분석하여 클래스 정보를 추출하고, 클래스간의 관계를 조사하여 상호의존성이 낮고 재사용성이 높은 클래스를 선택하여 컴포넌트화 시킨다. 재사용성은 프로그램내에서 얼마나 많이 사용되는가로 정의하였다. 클래스가 사용된다는 것은 데이터 관점과 기능관점에서 각각 다르게 측정될 수 있는데, 본 논문에서는 기능적 관점에 맞추어서 클래스의 재사용성을 측정한다. 측정된 재사용성을 통해서 컴포넌트의 후보를 사용자에게 제시하고 사용자로부터 선택받아 컴포넌트화 시키는 도구를 설계하였다.

1. 서론

기존의 객체지향 방식으로 구축된 시스템을 컴포넌트 기반의 시스템으로 전환하기 위해서는 소스코드의 클래스로부터 컴포넌트를 추출하는 작업이 가장 중요하다. 이 때 재사용성이 가장 높은 클래스를 어떻게 선택하고 판별할 것인지가 문제점이 된다. 객체지향 시스템은 일반적으로 클래스간의 밀접한 연관관계를 가지고 있다. 반면 컴포넌트 기반 시스템은 개별 컴포넌트의 독립성이 중요하다. 기존의 객체지향 시스템에서 일부분을 분리하여 컴포넌트화 시키는 작업은 컴포넌트화가 용이한 클래스를 선택하여 선택한 클래스를 시스템내의 다른 클래스와의 연관관계를 제거하는 과정이 포함되어야 한다.

본 논문에서는 객체지향 방식으로 작성된 프로그램을 컴포넌트 방식으로 재사용하기 위한 컴포넌트 추출 방법과 추출 시스템을 제안하고자 한다.

본 논문의 2 장에서는 객체지향 시스템과 컴포넌트 기반 시스템에 대해서 설명하고, 3 장에서는 컴포넌트 추출을 위한 방법에 대해서 살펴본다. 4 장에서는 이 과정을 지원하기 위한 시스템의 설계에 대해서 소개한다. 그리고, 마지막으로 5 장에서는 결론과 향후 연구방향을 제시하고자 한다.

2. 객체지향과 컴포넌트

2.1 객체지향 시스템

객체 모델은 객체지향 개념의 등장과 함께 제안된 가장 일반화 된 모델로서 대상 시스템의 정적인 자료 구조를 표현한다. 객체 모델이 기존의 구조적 방법론의 개체관계도(Entity Relation Diagram)와의 가장 큰 차이점은 정보와 정보를 처리하는 기능을 결합시킨 캡슐화와 정보 은폐, 그리고 상속(Inheritance)을 통한 추상화 혹은 일반화 과정을 들 수 있다.

캡슐화(Encapsulation)는 필요한 자료와 함수를 하나의 단위로 묶어서 정의하는 것으로서 그 단위가 객체가 된다. 정보 은폐(Information Hiding)는 객체 내의 자료를 외부로부터 숨긴 채, 공개된 메소드들을 통해서만 허가된 정보를 접근하도록 하는 기법이다. 정보 은폐를 통하여 객체 내부의 자료에 대한 무결성을 확보할 수 있다.

추상화(Abstraction)란 관련 있는 정보들을 일반화

시켜서 수퍼클래스로 추출 함으로서 확장성과 적용성, 재사용성을 확보하는 기법이다. 하위 객체가 상위 객체에 대해서 " Is_A" 혹은 " Kinds_Of" 관계를 가지며 동일한 기능은 상위 객체에서 한번만 구현해주고, 서브 클래스들에서는 서로 상이한 부분들에 대해서만 작업해주도록 함으로서 재사용성과 생산성을 향상시키는 장점을 제공한다.

이와 같은 상속기능을 이용하여 일반적인 객체 지향 프로그램은 클래스 단위로 프레임워크를 확장해서 구성해나가는 특징이 있다. 이런 특징은 캡슐화에 대치되어 클래스간의 연관성을 더 높여주게 된다.

2.2 컴포넌트 기반 시스템

1990 년대에 국내외 IT 업계는 현재 두 종류의 중요한 패러다임의 진행을 경험하고 있다. 첫째, 생성되는 어플리케이션의 종류와 관계된 새로운 컴퓨팅 모델로서 네트워크 컴퓨팅(network computing)의 출현이며, 두번째, 이러한 어플리케이션이 생성되는 방법과 관련되는 것으로서 전통적인 개발방법과 아주 다른 방법인 컴포넌트 기반 개발(CBD: Component-Based Development) 방법의 출현이다. CBD 는 낮은 비용으로 빠르게 어플리케이션을 개발하는 최선의 방법으로 간주되며, 이 개발 방법은 그 동안 산업계에서 장점으로 부각되고 있는 분산객체 기술(distributed object technology)과 자연스럽게 조화될 수 있다. 즉, 어플리케이션 개발자는 인터넷을 통해 해당 기업이나 조직의 network 상에 분산되어 있는 사용 가능한 바이너리 컴포넌트들의 인터페이스만을 간단히 정의하고 통합하여 응용 시스템을 쉽고(easier), 빠르고(faster), 저렴하게(cheaper) 개발할 수 있다.

컴포넌트는 관점에 따라서 다양하게 정의되고 있다. 비즈니스 컴포넌트는 자치적인(Autonomous) 비즈니스 개념이나 비즈니스 프로세스의 소프트웨어 구현물이며 보다 큰 비즈니스 시스템의 재사용 요소로서의 개념을 표현하고, 구현하고, 전개하기에 필요한 소프트웨어 산출물로 구현되어 있다고 정의되고 있다.

컴포넌트의 특성은 개별 컴포넌트들이 독립성이 강하고, 조립 가능하다는 것이다. 독립성을 얻기 위해서는 개별 요소간의 의존성 및 연관성이 적어야 한다. 또한 조립가능하기 위해서는 상호간의 연결을 위한 인터페이스가 잘 정의되어 있어야 하고, 인터페이스에 유연성을 갖고 있어야 한다. 이러한 특징은 객체지향 시스템으로서 표현하기 힘든 부분으로서 기존의 객체지향 시스템을 컴포넌트 기반 시스템으로 변환하기 위해서는 보다 더 독립성과 인터페이스를 재 정의 할 수 있도록 재구성되어야 한다..

3. 객체지향 프로그램에서의 컴포넌트 추출

본 연구의 컴포넌트 추출의 기본 목표는 컴포넌트화 시켰을때 가장 재사용성이 높은 부분을 찾아서 컴포넌트화 시키는 것이다. 여기서 재사용성이 높다는 것은 다른 시스템에서의 활용가치가 높다고 말할 수 있다.

현재 본 연구에서의 컴포넌트 추출방법의 개략적인 절차는 다음과 같다.

1. 기존 코드에서 컴포넌트로 변환할 만한 부분 검색
2. 컴포넌트로 변환할 가치가 있는지 사용자 결정
3. 변환할 컴포넌트를 EJB Style 로 변환

구조 분석에 의한 컴포넌트 추출방법은 소스코드의 모듈간의 관계 분석을 통해서 컴포넌트를 식별하는 것이다.

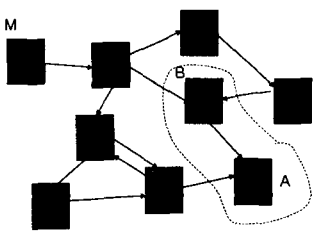
클래스간의 관계인 의존성(dependency)는 다음과 같이 정의할 수 있다.

- 메소드에서 다른 클래스의 인스턴스의 메소드를 호출
- 메소드 내에서 다른 클래스의 멤버 데이터를 사용 및 변경, 생성
- 클래스 내에서 다른 클래스를 멤버변수로 정의

이러한 관계중 중복되어 나타나는 경우는 하나의 의존성으로 생각한다. 의존적인 관계는 단지 사용의 빈도로서 결정될 수 없기 때문이다.

일반적으로 컴포넌트의 사용방법은 컴포넌트가 서비스 제공을 하고 이 서비스를 응용프로그램측에서 이용하는 스타일이다. 이를 통해서 클래스간의 관계인 dependency 에서 다른 클래스에 대한 의존성이 낮을수록 컴포넌트로서의 독립성이 높다고 할 수 있다.

<그림 1>은 이와 같은 의존성을 찾은 예이다.



<그림 1> 클래스간의 호출 관계

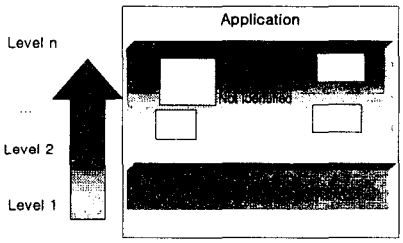
다른 클래스로의 의존성이 없는 경우를 base core 클래스라고 정의한다. base core 클래스는 해당 응용프로그램의 제일 하위 레이어를 구성하는 부분으로서 Level 1 클래스라고 정의한다. 보다 상위로 올라갈수록 Level 의 숫자는 늘어난다.

1. base core 클래스를 먼저 찾는다.
2. base core 클래스를 중심으로 두번째 클래스를 찾아서 클러스터링한다.
3. 클러스터링 한 결과를 base core 로 삼아서 Level n

까지 2 의 과정을 반복한다.

<그림 1>에서 클래스 A 는 base core 클래스로서 선택되었다. 이 클래스 A 를 사용하는 의존성 관계를 가진 클래스 B 까지 포함하여 Level 2 컴포넌트가 만들어진다. 클래스 B 를 컴포넌트화 시킬 경우 클래스 A 는 반드시 필요하므로 클래스 A 와 B 를 함께 묶어주는 것은 합당하다. 그러나 Level 2 에서 핵심적인 역할을 하는 클래스는 클래스 A 가 아니라 클래스 B 라고 말할 수 있다. 클래스 B 는 클래스 A 가 가지고 있지 않은 응용 로직을 가지고 있기 때문이다.

이러한 과정을 반복적으로 거치다보면 최상위 레벨에 도달하게 되는데, 가장 최상위 Level 일 경우의 컴포넌트는 Application 자신이 된다 이러한 추출방법을 통해 얻어질 결과물은 <그림 2>와 같은 형태를 띄게 된다.



<그림 2> 컴포넌트 추출 결과

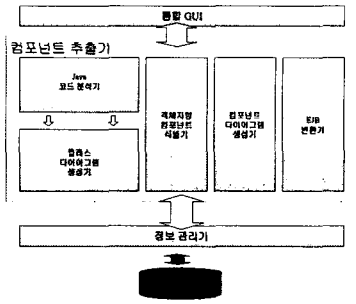
대상 응용프로그램이 계층구조(Layered)를 가지고 있다고 하면, 계층구조상에 하부계층에 속하는 부분부터 컴포넌트로 인식되어 컴포넌트로 추출되어진다.

사용자에게는 Level 1 에 해당하는 클래스부터 우선순위를 두어 컴포넌트 후보로서 제시한다. 이때 사용자가 컴포넌트로 만들기 위해 선택한 클래스는 해당 상위 클래스들까지 모두 포함하여 컴포넌트로 만들어져야 한다.

4. 컴포넌트 추출 시스템의 구성

컴포넌트 추출시스템은 객체 지향 언어인 Java 로 개발된 소프트웨어를 분석하여 사용할 수 있는 컴포넌트를 추출하는 컴포넌트 추출 지원도구 이다. 기존 객체지향 프로그램은 프로그램 모듈간의 상호 의존성이 높기 때문에 이를 용이하게 식별하여 식별된 모듈간의 의존관계를 단절시켜 컴포넌트로 만든다..

컴포넌트 추출기의 구성은 <그림 4>과 같다.



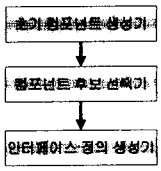
<그림 3> 컴포넌트 추출기의 구성 블록

컴포넌트 추출기에 관련된 통합 GUI 나 정보 관리기 블록에 관한 설명은 컴포넌트 추출기의 특성상 사용자의 상호작용 및 저장 정보의 일관성 유지 등을 정교하게 관리할 필요가 없으므로 컴포넌트 추출기를 구성하는 블록 중 해당 기능이 필요한 블록 내에 포함되는 형태로 구현된다.

Java 코드 분석기는 기존에 작성된 Java 코드를 분석하여 클래스들에 대한 정보를 추출하는 블록이다. Java 코드 분석기는 SUN 사에서 컴파일러 생성 도구로 제공하는 JLEX 와 CUP 을 이용하여 구현된다. Java 코드 분석기의 기능은 일반적인 컴파일러 구현 기법을 이용하여 개발할 수 있다. 먼저 입력으로 들어온 소스 프로그램을 문법의 기본 단위인 토큰들의 스트림으로 변환한다. 이러한 토큰 스트림을 처리하여 소스 코드의 모든 의미 있는 정보를 분석기 내부 자료구조에 저장되는 파스 트리로 변환한다. 이때, 분석 작업 과정에서 심볼 테이블을 관리하거나 내부 자료구조의 내용을 일관성 있게 유지하는 과정이 필요하다. 파스 트리를 이용하여 클래스 다이어그램을 만들어내기 위한 기본적인 각 클래스에 대한 정보와 클래스간의 관계에 관한 정보를 생성한다.

클래스 다이어그램 생성기는 클래스에 대한 정보와 클래스간의 관계 정보를 바탕으로 클래스 다이어그램 편집기에서 사용하는 형식으로 클래스 다이어그램을 생성한다. 상속관계 등의 쉽게 찾아낼 수 있는 관계정보는 클래스 정보를 이용하여 바로 생성해내며, association, dependency, aggregation 등의 정보는 클래스 내의 메소드간의 관계정보를 바탕으로 규칙에 따라 생성해낸다. 저장된 클래스 다이어그램은 클래스 다이어그램 편집기에 로딩시켜 사용자에게 보여준다.

객체지향 컴포넌트 식별기는 소스코드로부터 생성된 클래스와 클래스 간의 관계정보를 바탕으로 하여 컴포넌트를 식별해낸다. 초기에 식별해낸 컴포넌트 중에서 사용자가 유용한 컴포넌트를 선택하고 정제하여 선택한 컴포넌트를 대상으로 인터페이스 정의를 생성한다. <그림 4>는 객체지향 컴포넌트 식별기의 세부 모듈간의 관계를 나타낸 것이다.



<그림 4> 객체지향 컴포넌트 식별기의 구조

- 초기 컴포넌트 생성기: 초기 컴포넌트 생성기는 컴포넌트 자동 식별 알고리즘을 바탕으로 컴포넌트로서의 후보가 될만한 컴포넌트들을 사용자에게 제시한다.
- 컴포넌트 후보 선택기: 컴포넌트 후보 선택기는 사용자와의 작업을 통해서 제시된 컴포넌트의 수정 및 컴포넌트로 채택할 것을 결정하는 기능을 수행한다.
- 인터페이스 정의 생성기: 인터페이스 정의 생성기는 사용자가 선택한 컴포넌트의 외부 인터페이스로 사용할 메소드를 선정하게 하고, 선정된 인터페이스에 맞는 인터페이스 정의 파일을 생성해낸다.

컴포넌트 다이어그램 생성기는 객체지향 컴포넌트 식별기에서 추출해낸 컴포넌트 정보를 이용하여 컴포넌트 간의 support 및 dependency 관계를 추출하여 컴포넌트 다이어그램을 생성한다. 생성된 컴포넌트 다이어그램은 컴포넌트 다이어그램 편집기에 로딩시켜 사용자에게 보여준다.

EJB 변환기는 식별된 컴포넌트를 EJB 형식의 컴포넌트로 변환하는 기능을 수행한다. 컴포넌트 추출기를 통해 생성된 컴포넌트들은 재사용을 증진시키기 위하여 컴포넌트를 이루는 클래스들의 메소드가 EJB 환경에서 사용될 수 있도록 변경되어야만 한다. 이를 위해 컴포넌트 다이어그램 생성과정에서 생성해낸 컴포넌트 관계 정보를 바탕으로 보다 정밀하게 어떤 메소드가 어떻게 바뀌어야 할 지에 대한 결정 정보를 만들어낸다. 이러한 정보를 바탕으로 각 클래스의 메소드를 구성하는 소스코드중의 다른 컴포넌트들에 대한 상호 의존적인 호출관계를 EJB 인터페이스를 호출하는 형태로 변환시킨다.

5. 결론

본 논문은 객체지향 프로그램에서 컴포넌트를 추출하기 위한 방법과 이를 구현하기 위한 시스템의 설계를 제안하였다. 각 클래스간의 사용관계를 이용하여 재사용 빈도를 측정하여 컴포넌트를 추출하려 하였다.

이러한 방법은 다음과 같은 문제점이 있다.

- 기본적으로는 callgraph 관계에 의해서 판정하는 방법이다.
- 상속관계등을 활용할 수 있는 방안
- 클래스의 클러스터링을 통해서 하나의 EJB 컴포넌트를 만드는 방안

클래스 메소드 호출과 인스턴스 메소드 호출은 다르게 해석되어야 하며, 인스턴스화 시켜서 사용하는 경우와 그냥 호출할 경우도 다르게 해석되어야 한다. 슈퍼클래스를 이용해서 호출되는 경우 상속관계로 이루어진 패밀리의 경우 어떻게 해석해야 하는지도 결정되어야 한다. 또한 복잡도에 의해서 근접 클래스를 하나의 컴포넌트로 묶어내는 방안도 있어야 한다. 이때, 사용관계를 분석하는데 있어 더 다양한 호출의 경우를 비교하여 효과적인 사용빈도가 될 수 있도록 하는 것이 필요하다. 앞으로 비즈니스 로직까지의 추상화된 부분을 컴포넌트화 시키는 방법에 대해서도 연구가 필요하다.

참고문헌

[1] Roger S. Pressman. "Software Engineering, A Practitioner's Approach", 3rd Ed. McGraw Hill, 1997

[2] 유영란, 김수동. "객체지향 객체 모델의 컴포넌트 모델 전환 지침", 한국정보처리학회 춘계학술발표 논문집, 2000.

[3] Grady Booch, James Rumbaugh, and Ivar Jacobson, "The Unified Modeling Language User Guide", Addison-Wesley, 1999.

[4] Kozaczynski, W. and Booch, G., "Component-Based Software Engineering", IEEE Software, pp. 34-36, Sept./Oct. 1998.

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

[6] souza, D. F. and Wills, A. C., "Objects, Components, and Components with UML", Addison-Wesley, 1998

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