2001년 한국정보처리학회 추계 학술발표논문집 제8권 제2호

클래스 명세를 기반한 재사용 컴포넌트 추출 방법에 관한 연구

김정종, 박운재, 송의철, 송호영
경남대학교 컴퓨터공학과
e-mail:humanism@zeus.kyungnam.ac.kr

A Study on How to Extract Reusable Components Based on Class Specification

Jung-Jong Kim, Woon-Jai Park, Eui-Cheol Song, Ho-Young Song
Dept of Computer Engineering, Kyungnam University

요약

컴포넌트 기반 소프트웨어 개발에서 컴포넌트 재사용을 위한 적용 방법은 중요한 문제로 대두되고 있다. 그러나 많은 연구들이 개발하려는 시스템의 컴포넌트 명세를 이용하여 기존의 컴포넌트를 재사용하는 방법을 적용한다. 이는 개발하려는 시스템의 컴포넌트 명세를 작성하여야 하기 때문에 재사용성과 유연성의 효율이 떨어지며, 개발 시간과 비용이 상대적으로 증가될 수 있다. 따라서 본 연구에서는 요구분석 단계의 산출물인 Use Case와 클래스 명세를 이용하여 기존의 컴포넌트를 재사용할 수 있는 방법을 제시한다.

1. 서론

컴퓨터 응용 분야는 점점 범위가 확대되고 복잡 다양해지고 있다. 따라서 기존의 소프트웨어 개발 방법으로는 이에 대응할 수 있는 범위가 한정되어 있으며, 이러한 방법에 대응할 수 있는 객체지향 개발 방법을 적용한 소프트웨어의 재사용에 많은 연구와 방법을 제시하였다. 그러나 객체나 클래스 개념으로는 재사용과 생산성을 극대화하기에는 부족한 면이 있어, 최근 컴포넌트 기반 소프트웨어 개발(CBSD : Component-Based Software Deveopment) 방법의 연구가 활발하게 진행되고 있다[1].

컴포넌트 기반 소프트웨어 개발은 소프트웨어 개발 시간과 비용을 줄이고, 생산성을 향상시키며, 이미 검증받은 컴포넌트를 사용함으로써 위험성을 줄일 수 있다. 또한 일관성을 높이며, 어플리케이션 개발을 위한 여러 가지 해결책을 제시함으로 가장 좋은 해결책을 선택할 수 있다[2].

현재 많은 연구들이 개발하려는 시스템의 컴포넌트와 기존의 컴포넌트를 비교하여 컴포넌트를 재사용한다. 그러나 개발하려는 시스템의 컴포넌트는 설계 단계의 산출물이므로 재사용성과 유연성의 효율이 떨어진다.

따라서 본 연구에서는 요구분석 단계의 산출물인 Use Case와 클래스 명세를 이용하여 기존의 컴포넌트를 재사용할 수 있는 방법을 제시한다.

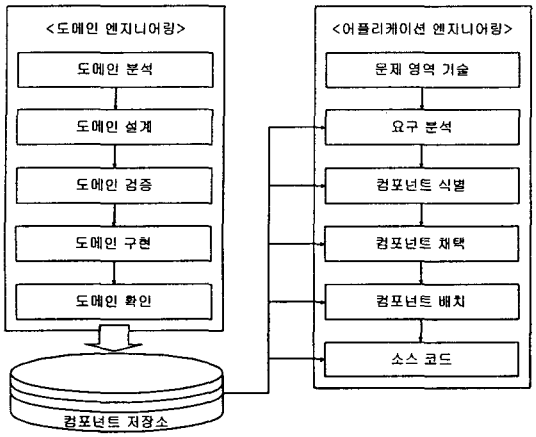
이 방법은 요구분석 단계에서 재사용할 컴포넌트를 검색하므로 재사용성을 보다 극대화할 수 있고, 개발 기간과 비용을 절감할 수 있으며, 보다 유연한 시스템을 개발할 수 있을 것이다.

2. 컴포넌트 기반 소프트웨어 개발

현재 새롭게 대두되고 있는 컴포넌트 기반의 소프트웨어 개발은 기존의 컴포넌트를 조합함으로써 시스템을 완성하는 것을 의미한다[2][4]. 컴포넌트를 사용한 소프트웨어 시스템 개발 기술은 과거 구조적인 방법이나 객체지향 방법이 해결하지 못하였던 소프트웨어 재사용과 생산성, 시스템 품질 관리 등에 대한 새로운 해결책으로 제시되고 있다. 또한 전통적인 소프트웨어 프로젝트 실패의 원인이 되었던 예산의 초과, 사용자의 요구사항과 어긋나는 소프트웨어의 생산, 개발 기간의 지연 등 소프트웨어 위기를 해결할 수 있는 패러다임으로 컴포넌트 기술은 더욱 각광을 받고 있다.

컴포넌트 기반 소프트웨어 개발은 크게 컴포넌트

를 개발하려는 도메인 엔지니어링과 컴포넌트를 조립하여 어플리케이션을 개발하는 어플리케이션 엔지니어링으로 나눌 수 있다.



[그림 1] 컴포넌트 기반 소프트웨어 개발 프로세스

그림 1은 컴포넌트 기반 소프트웨어 개발 프로세스를 보여준다. 도메인 엔지니어링은 컴포넌트 기반 어플리케이션 개발을 지원하는 도구로서 개발이 이루어지며, 컴포넌트 기반 어플리케이션 개발 분야에 많은 연구가 이루어지고 있다.

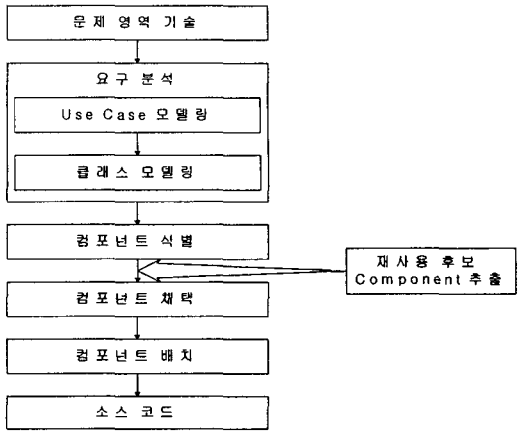
3. 컴포넌트 명세

3.1 기존의 컴포넌트 재사용

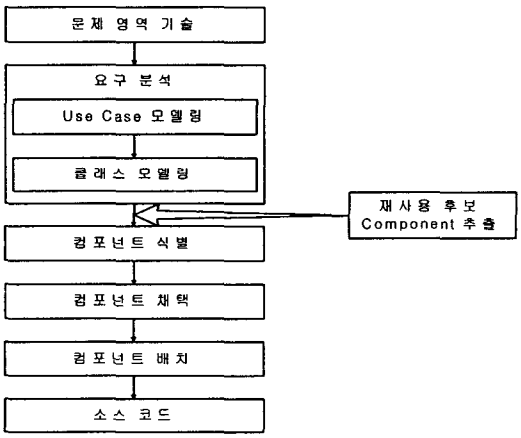
컴포넌트 재사용은 기존에 개발되어 있는 컴포넌트들을 새로운 소프트웨어 개발에 사용함으로서 소프트웨어 개발 기간을 단축시키고, 개발비용의 절감과 소프트웨어의 생산성을 향상시킬 수 있다. 현재 컴포넌트 재사용에 관한 많은 연구들이 진행되고 있지만, 대부분이 새로운 시스템의 컴포넌트를 식별한 후 기존의 컴포넌트와 비교하여 재사용을 하고 있다.

그림 2와 같은 기존의 적용 방법은 설계 단계에서 식별한 컴포넌트 정보를 이용하여 기존의 컴포넌트를 검색함으로서 재사용할 후보 컴포넌트들을 추출하였다. 그러나, 이 방법은 재사용성과 유연성의 효율이 떨어질 수 있으며, 개발 시간과 비용이 많이 든다.

따라서, 그림 3에서와 같이 요구분석 단계에서 기존의 컴포넌트를 검색하여 재사용할 후보 컴포넌트들을 추출함으로서 전체 시스템의 아키텍처를 구성하는데 효율성을 증가시키고, 컴포넌트 적용 대상의 사전 식별이 가능하여 요구사항에 가장 적합한 시스템으로 개발할 수 있고, 재사용성의 증대와 효율성을 높일 수 있으며, 개발 시간과 비용을 절감할 수 있다.



[그림 2] 기존의 재사용 대상 컴포넌트 추출 방법



[그림 3] 새로운 재사용 대상 컴포넌트 추출 방법

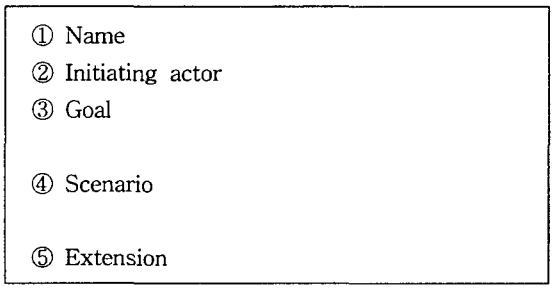
3.2 Use Case 명세

Use Case는 사용자와 시스템간의 상호작용을 정의하기 위한 기법으로 소프트웨어 경계를 명확히 정의해 주고 시스템에 대한 요구사항을 반영하며, 시스템 범주 내에서 발생하는 상호작용을 표현한다.

그림 4는 Use Case 명세를 나타낸다. 명세를 구성하는 요소는 Name, Initiating actor, Goal, Scenario, Extension으로 구성되어 있다.

① Name : Use Case의 이름을 나타낸다.
② Initiating actor : Use Case를 최초로 기동시킬 수 있는 액터의 이름을 나타낸다.

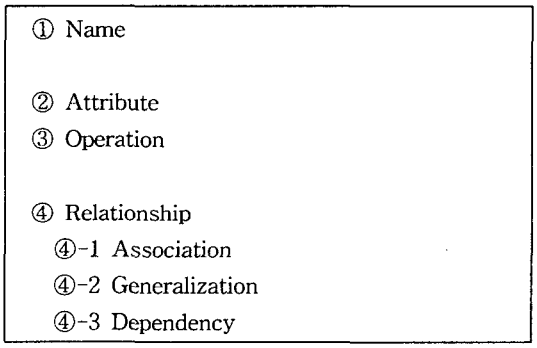
③ Goal : Use Case의 목표에 대해 간단하게 기술한다.
④ Scenario : 무슨 일이 발생하며, 언제 이와 같은 일이 성공적으로 수행될 수 있는지를 설명해 주는 사건의 흐름을 기술한다.
⑤ Extension : 시나리오에 추가적인 내용을 덧붙이거나 대안적인 내용을 기술한다.



[그림 4] Use Case 명세

3.3 클래스 명세

그림 5는 클래스 명세를 나타낸다. 클래스 명세에는 Name, Attribute, Operastion, Relationship(Association, Generalization, Dependency)으로 구성된다.

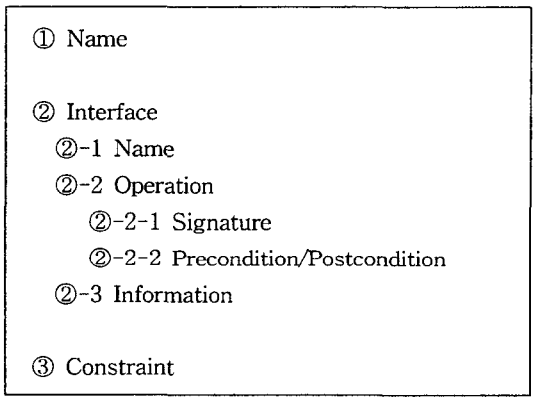


[그림 5] 클래스 명세

① Name : 클래스 이름을 나타낸다.
② Attribute : 클래스의 특성을 나타내는 것으로 이름과 타입을 기술한다.
③ Operation : 객체에게 요청할 수 있는 행동을 말하며 이름과 정보를 기술한다.
④ Relationship : 클래스 사이의 의미적 관계를 기술한다.
④-1 Association : 클래스 사이의 기본적인 개념 연결을 나타내고 각 클래스의 역할을 기술한다.
④-2 Generalization : 상속 관계를 나타내는 항목으로 상속 관계의 클래스들을 기술한다.
④-3 Dependency : 한 클래스가 다른 클래스를 사용하는 관계로 의존 관계의 클래스들을 기술한다.

3.4 컴포넌트 명세

컴포넌트 명세는 컴포넌트가 제공해야 할 인터페이스들을 정의하고 그 인터페이스 정의들이 어떻게 서로간에 대응하는지를 기술한다. 또한 사용하거나 소비하고 있는 인터페이스와 제약조건도 함께 기술한다. 그림 6은 컴포넌트 명세를 나타낸다.



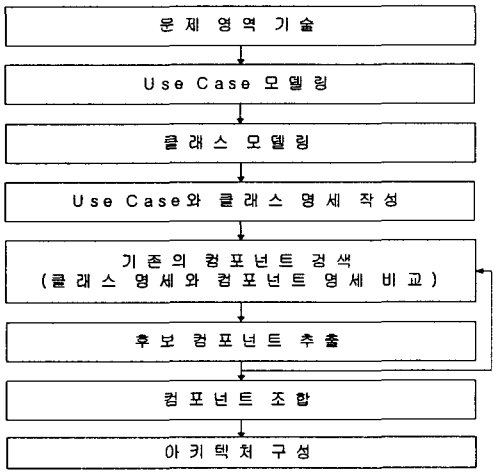
[그림 6] 컴포넌트 명세

① Name : 컴포넌트의 이름을 나타낸다.
② Interface : 컴포넌트에서 제공되고 사용되는 인터페이스들을 기술한다.
②-1 Name : 인터페이스의 이름을 나타낸다.
②-2 Operation : 컴포넌트 오브젝트가 수행하게 될 서비스나 기능을 기술한다.
②-2-1 Signature : 입력 매개변수, 출력 매개변수, 컴포넌트 오브젝트의 상태 변화의 결과를 기술한다.
②-2-2 Precondition/Postcondition : 오퍼레이션이 해야 할 일을 기술한다.
②-3 Information : 인터페이스와 관계를 가지는 클래스를 기술한다. 클래스에는 이름, 속성, 연관 관계 등을 기술한다.
③ Constraint : 컴포넌트 상호작용의 제약과 인터페이스간의 제약조건을 기술한다.

4. 컴포넌트 재사용

4.1 컴포넌트 재사용 단계

그림 7에서와 같이 컴포넌트 재사용 단계는 먼저 개발하려는 시스템에 대한 요구사항인 문제 영역 기술을 분석하여 Use Case 모델링과 클래스 모델링을 한다. 클래스 모델링을 통해 산출된 클래스의 구성요소들에 대한 명세를 작성하고, 이 클래스 명세를 이용하여 기존의 컴포넌트를 검색함으로서 재사용할 수 있는 후보 컴포넌트를 추출한다. 추출된 후보 컴포넌트들 중 Use Case 명세를 토대로 가장 적합한 컴포넌트를 선택한다. 그리고, 추출한 컴포넌트들과 새로 생성한 컴포넌트들을 조합하여 새로운 시스템의 전체 아키텍처를 구성한다.



[그림 7] 컴포넌트 재사용 단계

4.2 컴포넌트 검색 방법

클래스 명세를 이용한 재사용 컴포넌트 추출을 위한 방법의 적용은 클래스 명세 내부의 구성요소들과 컴포넌트 명세의 구성요소를 매칭하여 재사용 가능한 최적의 컴포넌트를 추출하는 단계별 진행 방법은 다음과 같다.

① 클래스 명세 이름 : 인터페이스 정보의 이름
② 클래스 명세 속성 : 인터페이스 정보의 속성
③ 클래스 명세의 OP : 인터페이스 정보의 OP

위와 같은 3단계의 매칭과정을 수행하여 후보 컴포넌트를 추출하며, 후보 컴포넌트는 클래스 명세의 관계와 Use Case 명세를 통해 최적의 컴포넌트를 추출한다.

5. 결론

컴포넌트 기반 소프트웨어 개발은 소프트웨어 개발 시간과 비용을 줄이고, 생산성을 향상시키며, 이미 검증받은 컴포넌트를 재사용함으로서 위험성을 최소화 할 수 있다. 또한 일관성을 높이며, 어플리케이션 개발을 위한 여러 가지 해결책을 제시함으로서 고 품질의 소프트웨어 개발을 위한 해결책을 선택할 수 있다. 그러나 기존의 컴포넌트 재사용은 개발하려는 시스템의 컴포넌트를 식별한 후 컴포넌트를 검색하기 때문에 재사용의 효율을 극대화시키지 못하고 있다. 따라서 본 연구에서는 요구분석 단계에서의 Use Case와 클래스 명세를 이용하여 재사용할 컴포넌트를 추출할 수 있는 방법을 제시하였으며, 이러한 방법은 재사용성을 보다 극대화할 수 있고, 개발 기간과 비용을 절감할 수 있으며, 보다 유연한 시스템을 개발할 수 있을 것이라 본다.

향후 클래스 명세와 컴포넌트 명세의 관계 설정의 정형화와 각 단계별 매칭에 대한 다양하고 유연성 있는 적용을 수용할 수 있는 방안이 필요하다.

참고문헌

[1] Alan W. Brown, Kurt C.Wallnau, "The Current State of CBSE", IEEE Software, Vol. 15, No. 5, 1998

[2] Jun-Jang Jeng and Betty H.C. Cheng, "A Formal Approach to Reuse More General Components", IEEE Proceedings of 9th Knowledge-Based Software Engineering Conference, Monterey, California, September, 1994

[3] Short, K., "Component Based Deveopment and Object Modeling", Sterling Software, 1997

[4] P.Allen and S.Frost, "Component Based Develpment for Enterprise System: Applying the Select Approach", Cambridge University Press, New York, (1997)

[5] A. Mili, R. Mili and R. Mittermeir, "A Survey of Software Components Storage and Retrieval", The Institute for Software Research Technical Report, 17, October, 1997

[6] Amy Moormann Zaremski and Jeannette M. Wing, "Specification matching of software component", ACM Trans. on Software Engineering and Methodology, Vol. 6, 1997

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

[8] John Cheesman, John Daniels, "UML Components - A Simple Process for Specifying Component-Based Software", Addison-Wesley, 2000