디자인 패턴 기반의 컴포넌트 모델링도구에 관한 연구

서영준*, 송영재
경희대학교 전자계산공학과
e-mail:cionblue@case.kyunghee.ac.kr

A Study on Component Modeling Tool based on Design Pattern

Young-Jun Seo*, Young-Jae Song
Dept of Computer Engineering, KyungHee University

요약

현재 소프트웨어 개발 생산성과 품질을 개선하기 위해 컴포넌트 재사용에 관한 연구가 활발히 이루어지고 있으며, 특히 분산 개발 환경에서 팀 개발의 생산성을 향상시키기 위한 컴포넌트 재사용 방법이 연구되고 있다. 그러나, 이들 컴포넌트들을 효과적으로 개발, 공유, 관리하기 위한 방안이 부족한 상황이다. 따라서, 본 논문에서는 컴포넌트의 재사용성을 최대화하기 위하여 반복적 설계 문제를 해결하기 위한 구조를 갖고 있는 디자인 패턴을 대상으로 하는 공유 가능한 디자인 패턴 기반의 컴포넌트 개발 시스템을 제안하였다. 본 논문에서 제안하는 DPCM(Design Pattern Component Modeling) 도구는 추상적인 디자인 패턴 구조와 구체적인 컴포넌트 구조를 저장하기 위해 각각 별도의 Library를 구축한다. 그리고, 두 Library 사이에는 패턴 구조를 특정 애플리케이션에 적합하게 구체화하는 과정을 시각적으로 모델링 하며, 구체적인 구현 코드와 함께 COM 사양의 컴포넌트로 변환한다. 본 논문에서는 개발자들이 애플리케이션 개발에 필요한 컴포넌트를 제공 받을 수 있는 디자인 패턴 기반의 소프트웨어 컴포넌트 개발 도구의 방향을 제시하였다.

1. 서론

소프트웨어 재사용은 그동안 코드 단위로 이루어져 왔으나, 최근에는 컴포넌트 단위로 바뀌어 가고 있다. 사용자들은 컴포넌트 재사용에 의해 개발 생산성과 품질 개선을 기대할 수 있다[6]. 특히, 소프트웨어 설계 컴포넌트를 분산된 소프트웨어 개발에 사용하는 것은 팀 개발의 생산성을 향상시킬 수 있다. 또한, 이들 컴포넌트들을 하나의 애플리케이션에 통합함으로서 재사용성과 유지보수의 용이성이 기대된다. 그러나, 분산된 개발 팀 사이에 소프트웨어 설계 컴포넌트를 공유, 관리하는 것은 어려우며 내부적으로 설계 컴포넌트들을 교환하기 위한 방안이 필요하다.

디자인 패턴은 반복적 설계 문제를 해결하기 위한 구조를 가지며, 컴포넌트로 구성하여 의사소통과 이해의 수단으로 사용될 수 있다[1][3]. 현재 일부 객체지향 모델링 도구는 인터페이스를 갖는 아키텍쳐 구성요소로서 디자인패턴을 지원하고 있으나[5], 대부분의 설계자들은 수작업으로 디자인 패턴의 구조를 설계에 적용하고 있어 실수가 발생할 가능성이 있다. 또한, 동일한 패턴일지라도 적용되는 애플리케이션마다 다른 특성이 추가되어야 한다.

따라서, 본 논문에서는 디자인 패턴의 사용을 증진하기 위해 디자인 패턴을 컴포넌트로 구성 할 수 있는 시스템을 제안한다. Gamma의 구조, 행위 패턴들을 대상으로 Pattern Library가 구현되며, 개발자들에 의해 선택되어진 디자인 패턴은 에디터를 통한 시각화된 구체화 과정을 거치게 된다. 구체화된 패턴 구조는 변환 모듈을 통해 COM 사양으로 변환되어 Component Library에 저장, 공유된다. 개발자들은 컴포넌트들을 애플리케이션 개발에 이용할 수 있으며, 변환 모듈의 변경으로 다른 객체 기반 컴포넌트로도 확장이 가능하다. 본 논문에서는 디자인 패턴을 기반으로 하는 소프트웨어 설계 컴포넌트를 생성, 공유하는 과정을 제시하며 공유 가능한 소프트웨어 컴포넌트 개발 도구의 향후 방향을 제시한다.

2. 관련 연구

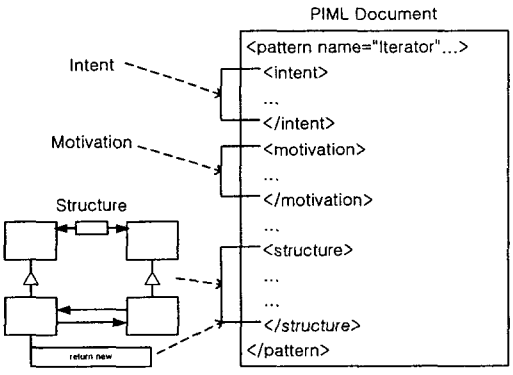
본 장에서는 디자인 패턴 기반의 컴포넌트 모델링 도구에 대한 관련 연구에 관하여 살펴보도록 하겠다.

2.1 디자인 패턴(Design Pattern)

디자인 패턴은 시스템 설계시에 자주 발생하는 문제들에 대한 "재사용 가능한 해결책"이라 할 수 있다[1]. 경험 많은 소프트웨어 엔지니어들의 해법들을 정리해서 이름을 부여하고 디자인적인 추상성을 주어 패턴화 시키면 시스템 설계시에 디자인 재사용성을 제공할 수 있다. Gamma가 제안한 디자인 패턴 카탈로그 책에는 23개의 패턴들이 사용 목적에 따라 기본, 생성, 분류, 구조, 행위 패턴의 분류에 속해 기술되었으며, 각 패턴은 의도(intent), 동기(motivation), 구조(structure), 참여자(participants)등의 섹션으로 구성된다.

2.2 PIML(Pattern Information Markup Language)

PIML은 SGML을 사용하여 디자인 패턴을 기술하기 위한 언어이며, 의도(intent), 동기(motivation) 같은 설명문(explanation text)과 구성 클래스들과 관계와 같은 구조 정보(structure information), 클래스의 행위를 표현하는 행위 정보(pseudo-code) 세 부분으로 구성된다[4].



(그림 1) PIML Document의 구조

(그림 1)은 PIML 문서의 전체 구조를 개략적으로 나타낸다. 특히 구조 정보는 role(클래스), operation(메소드), role 사이의 관계(aggregate, reference, inheritance, creation)로 구성된다. Pattern Library의 입력으로 사용되는 디자인 패턴 문서를 PIML로 기술함으로서 XSL과 같은 스타일시트에 따라 다른 형태의 패턴 정보를 제공할 수 있는 부가적인 장점이 있다.

2.3 COM(Component Object Model)

COM은 Microsoft에서 제공하는 프로그램의 컴포넌트 객체들을 개발하고 지원하기 위한 하부 기반 구조이다[2]. COM의 장점으로는 첫째, 언어 독립적이어서 어떠한 프로그래밍 언어로도 작성될 수 있으며, 서로 다른 언어로 작성된 COM 컴포넌트일지라도 서로 완전하게 커뮤니케이션을 한다. 둘째, 버전 호환성을 제공하므로 이전 버전의 COM 컴포넌트를 사용하는 애플리케이션도 문제가 발생하지 않는다. 셋째, 위치 투명성을 제공한다. 즉, COM 컴포넌트가 설치되고 실행하는 위치에 관계없이 같은 방법으로 해당 COM 컴포넌트를 사용할 수 있다.

3. 디자인 패턴 기반의 컴포넌트 개발을 위한 프로토타이핑 도구

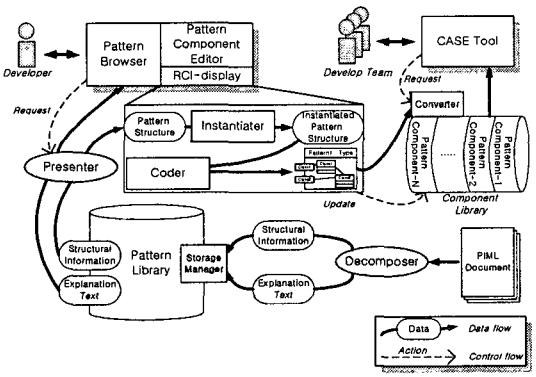
본 논문에서는 디자인 패턴 기반의 컴포넌트 개발과 저장, 공유를 위해 시각화된 관점을 제공하는 DPCM(Design Pattern Component Modeling) 도구를 제안한다.

3.1 시스템 구성

DPCM 도구의 전체 시스템 아키텍쳐는 (그림 2)와 같이 4개의 소프트웨어 대리인들로 구성된다: Decomposer, Storage Manager, Presenter, Converter. 소프트웨어 대리인은 개발자의 지시 또는 개입을 요청하지 않고 변화에 응답하기 위해 기능을 수행하는 소프트웨어 엔터티이다[7].

대상 디자인 패턴은 Gamma의 구조, 행위 디자인 패턴이 해당되며, 패턴 구조를 제공하지 않는 패턴은 제외하였다. PIML로 기술된 디자인 패턴은 Decomposer를 통해 파싱 되어 트리 형태로 설명문과 구조 정보로 분리된다. Storage Manager에서는 이 두 정보를 계층 구조의 테이블들로 Pattern Library에 저장한다. Presenter는 개발자가 요청한 패턴을 Pattern Library에서 검색하여 사용자 인터페이스로 표현되는 Pattern Browser와 Pattern Component Editor에 각각 전송한다. Pattern Component Editor에서는 가시적으로 표현된 구조 정보를 Instantiater를 통해 애플리케이션에 구체적

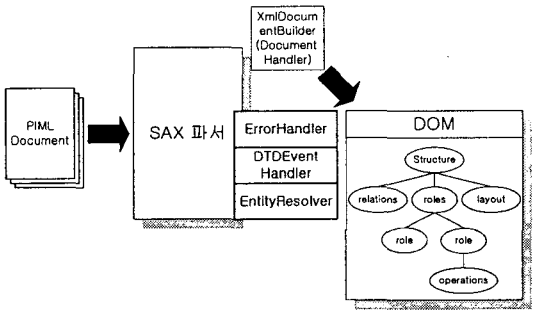
인 구조로 변환하며, Coder에서는 개발자에게 구체적인 구현 코드를 입력 받는다. 구체화된 디자인패턴 구조는 COM 변환 모듈인 Converter를 통해 COM 컴포넌트로 변환되어 Component Library에 저장되어 같은 개발팀들에게 공유된다. 개발 팀이 필요한 Component를 Converter에게 요구하면, Converter는 Component Library를 검색하여 해당 컴포넌트를 전달한다.



(그림 2) DPCM 도구의 시스템 아키텍쳐

3.2 패턴 분해기(Decomposer)

Decomposer의 XML 파서는 (그림 3)과 같이 입력으로 들어온 PIML 문서가 XML 문서 스펙에 맞게 쓰여있는 지를 검사하고, SAX를 이용해 특정 태그명을 지정하면 그 태그를 포함한 하위 노드를 찾는다.

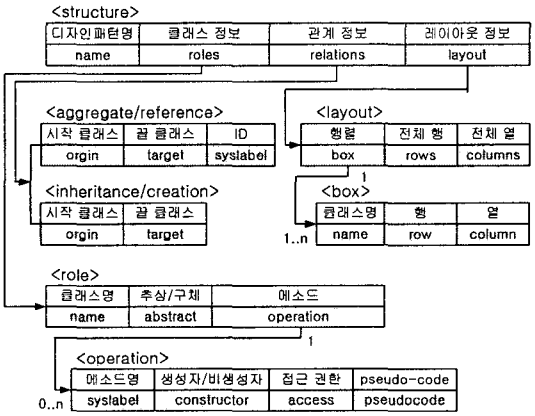


(그림 3) Decomposer의 파싱 과정

그 결과물은 DOM 트리로 구축되고 이를 트리 형태로 보여준다[8]. 특정 태그명으로 설명문의 경우 intent, motivation, applicability등을 사용하며, 구조 정보는 structure를 기술한다.

3.3 저장 관리자(Storage Manager)

Decomposer에서 파싱된 설명문과 구조 정보는 Storage Manager를 통해 데이터베이스 형태로 Pattern Library에 저장된다. (그림 4)는 Pattern Library에 저장된 구조 정보의 저장 구조를 나타내고 있다. 각 테이블의 이름은 태그명으로 기술되며, 필드들은 structure 테이블을 제외하고는 해당 태그의 속성들로 구성된다. 최상위 테이블인 structure 테이블의 필드는 바로 하위 단계의 태그명으로 이루어지며, 각 필드는 하위 단계의 태그 테이블에 연결되어 패턴의 전체 구조 정보가 계층 구조를 이룬다.



(그림 4) Pattern Library의 저장 구조

3.4 패턴 표현기(Presenter)

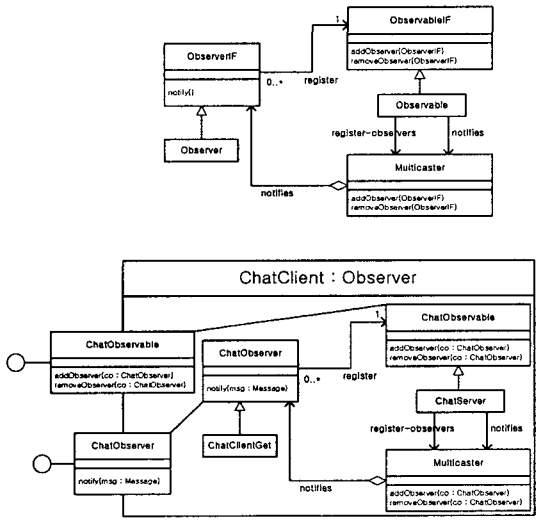
Presenter는 개발자에 의해 선택된 디자인 패턴을 Pattern Library에서 검색하여, 설명문과 구조정보로 구분된 데이터베이스 정보를 얻는다. 두 정보는 각각 Pattern Browser와 Pattern Component Editor에 보내져 표현된다.

3.5 패턴 컴포넌트 편집기(Pattern Component Editor)

Pattern Component Editor에서는 Presenter에서 받은 구조 정보를 우선 가시적으로 재구성한다. Instantiater에서는 애플리케이션 특성을 추가하여 디자인 패턴의 추상적인 구조를 특정 소프트웨어 디자인으로 변환한다[4][5].

그 과정을 살펴보면 첫째, 패턴 참여자인 클래스를 위한 이름을 선택하여 재명명 한다. 둘째, 패턴 클래스의 메소드를 위한 이름을 정의한다. 셋째, 혼란 방지를 위해 패턴의 유일한 이름들을 생성된 클

래스, 메소드명과 관계시키는 테이블인 이름 할당 테이블(name allocation table)을 제공한다. 넷째, 패턴에 기술된 role 사이의 관계와 생성된 class 사이의 관계를 일치시킨다. 다섯째, operation이 오버라이딩 되었다면 일치하는 method도 오버라이딩 한다. 여섯째, 외부와 커뮤니케이션을 할 인터페이스 클래스를 정의한다. 아래 (그림 5)는 Observer 패턴을 채팅 시스템에 적합하게 구체화한 그림이다[1][3].



(그림 5) Observer 패턴의 instantiation

Coder에서는 구체화된 패턴 구조에서 자동으로 패턴 구조의 틀을 이루는 코드를 생성하여, 개발자에게 제공한다. 개발자는 제공된 코드에 필요한 내부 구현 코드를 추가로 입력하게 된다.

3.6 Converter

Component Library의 Converter는 구체화된 패턴 구조를 템플릿에 기반을 둔 클래스 라이브러리인 ATL(Active Template Library)을 사용하여 COM 개체로 구현하여 저장한다. ATL에는 COM 개체 구현에 필수적인 IUnknown, IClassFactory등의 인터페이스에 대한 코드가 구현되어 있으므로, 손쉽게 COM 컴포넌트를 생성할 수 있다. 개발자들은 애플리케이션 개발에 필요한 컴포넌트를 Component Library에서 찾아 dll 파일 형식으로 제공 받는다.

4. 결론

본 논문에서는 소프트웨어 설계 컴포넌트가 어떻게 분산된 개발 팀에서 개발, 공유되고 관리될 수 있는지 소개하였다.

제안한 DPCM 도구는 디자인 패턴을 잘 정의된 인터페이스를 갖는 디자인 컴포넌트로 사용하였다. 동일한 구조의 디자인 패턴이라도 애플리케이션마다 다른 특성이 적용되어야 하므로, 시각화된 구체화 과정을 제공하였다. 또한, 그 과정에서 추상적인 디자인 패턴 구조와 구체적인 컴포넌트 구조를 저장하기 위해 각각 Pattern Library, Component Library를 구축하였다. 분산된 개발 팀의 개발자들이 Component Library에서 필요한 디자인 패턴 컴포넌트를 요청하면 연결된 변환 모듈에서는 COM 사양으로 변환하여 개발자에게 제공할 수 있다. 향후 연구 방향으로는 비 디자인 패턴까지도 포함하는 컴포넌트 통합 지원 도구로의 확장이 필요하다.

참고문헌

[1] E. Gamma, R. Helm, R. Johnson and J. Vlissides, Design Patterns : Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995

[2] Rogerson, Dale, Inside Com, Microsoft Pr, 1997

[3] Mark Grand, Patterns in Java, volume 1, Wiley, 1998

[4] M. Ohtsuki, N. Yoshida, "A Source Code Generation Support System Using Design Pattern Documents Based on SGML", Proc. of the APSEC 98, 1998

[5] S. Yacoub, H. Xue, H. Ammar, "Automating the Development of Pattern-Oriented Designs for Application Specific Software System", Proc. of the 3rd ASSET'00, 2000

[6] S. Yau, N. Dong, "Integration in Component-Based Software Development Using Design Patterns", Proc. of the 24th COMPSAC'00, 2000

[7] Y. Ye, "An Active and Adaptive Reuse Repository System", Proc, of the 34th HICSS'01, 2001

[8] Sun Project X Technology Release 2, http://java.sun.com/xml/