패턴지향 CASE도구의 저장소 구축에 관한 연구

서영준[U] 최한용 송영재
경희대학교 전자계산공학과
{cionblue, hychoi}@soft.kyunghee.ac.kr yjsong@nms.kyunghee.ac.kr

A Study on Building Repository of pattern-oriented CASE Tool

Young-Jun Seo[U] Han-Yong Choi Young-Jae Song
Dept. of Computer Engineering, KyungHee University

요 약

본 연구에서는 통합 멀티미디어 통신시스템을 구축하기 위한 CASE 도구의 개발에서 시스템 구축에 필요한 객체를 설계, 확장하거나, 설계된 객체의 재사용성을 높이기 위한 방안으로 패턴지향형정보저장소를 설계하였다. 그리고 시스템 구축에 필요한 객체의 재사용시 중복성을 피하며 복잡도를 감소시키기 위해 CASE 도구의 설계단계에서 객체를 통합관리/이용할 수 있도록 하고, 재사용에 필요한 객체를 패턴화하여 저장함으로써 저장소의 재사용 효율성을 증가시킨다. 따라서 본 연구에서는 정보저장소에 저장된 패턴-객체의 재사용시 유지보수 비용을 감소시키고 멀티미디어의 특성을 수용할 수 있도록 표준화된 단일 패턴-객체와 이를 이용한 복합 패턴-객체로 확장 가능하도록 혼합형 정보저장소를 설계하였다.

1. 서론

객체지향 소프트웨어(object oriented software)는 캡슐화된 구현 도구인 클래스들이 계층관계를 이루어 결합한 것으로 객체 사이에 메시지를 교환함으로써 구성요소의 독립적인 확장 및 재사용 효과를 얻을 수 있다. 이러한 객체지향 방법론은 재사용 객체의 패턴화에 의해 재사용성을 극대화 할 수 있다. 그러나 객체지향 방법론을 지원하는 많은 도구들이 있으나 표준화되고 효율적인 재사용을 지원하지 못하고 있다. 그리고 통합 멀티미디어 통신 시스템 개발을 위한 설계 부분에서 나타날 수 있는 많은 문제점 중의 하나가 반복적인 설계에 의한 분석/설계 시간과 경비의 낭비 문제와 시스템의 유지보수 문제다. 따라서 객체지향 CASE 도구의 재사용성을 극대화하기 위한 방안으로 정보저장소(repository)에 패턴을 적용함으로써 패턴지향형 CASE도구를 설계하는 기반을 마련하였다. 그리고 컴포넌트를 패턴화하고, 패턴 정보를 이용하여 시스템 구축 개발기간을 단축시키며 추상화된 패턴정보에 의한 재사용성과 통합된 저장소를 구축할 수 있게 하여 설계 정보를 공유하고 설계자들은 반복적인 설계를 피할 수 있게된다[1,2].

따라서 본 연구에서는 객체지향 이론을 바탕으로 표준화된 공유패턴(public pattern)을 정의하고, 정의된 패턴 내의 중복저장을 제거하기 위해 패턴 생성기에 의해 분류 저장되어진다. 그리고 분류된 패턴을 이용하여 설계하려는 패턴-객체의 모델을 설정하며 상속(inheritance)과 다형성(polymorphism)을 적용한 재사용 가능한 패턴-객체를 설계하였다. 그리고 상속은 정보 저장소 구축에 필요한 패턴-객체의 중복저장을 막기 위한 방안이며, 다형성은 설계된 패턴-객체를 이용하여 새로운 패턴을 파생시켜 설계자 고유의 개인패턴(private pattern)을 구축하게 된다. 따라서 공유패턴과 개인패턴의 두 가지 설계정보를 저장하는 혼합형 정보저장소(hybrid repository)를 이용하여 표준화된 패턴정보를 이용한 설계가 가능하며, 또한 설계 도메인에 유연성을 제공하게 된다.

2. 관련연구

2.1 정보저장소

정보저장소의 기능은 도구들이 정보를 공유하면서 개발하고자 하는 시스템에 관한 정보를 생성하고 관리하는 기본적인 기능을 가지고 있다. 각 도구들은 분석 및 설계정보를 획득하여 이를 분석하고 제어하며, 명세서 정보를 프로그램 코드나 문서로 변경시키는 기능을 수행한다. 따라서 CASE 도구의 정보저장소의 대표적인 기능은

다음과 같다[3].

① CASE 도구 통합
② 시스템 정보공유
③ 문서 표준화
④ 소프트웨어 재사용

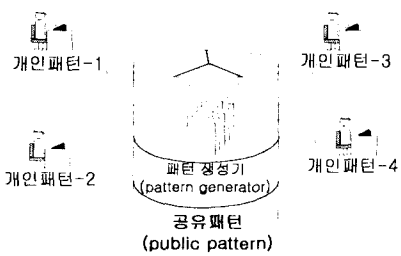
2.2 설계패턴

최근 객체지향 시스템설계에 관한 많은 연구들이 이루어지고 있다. 이중에서 설계패턴은 재사용 가능하고 복잡한 객체지향 시스템의 개발에 관한 일반적인 설계 문제점들을 설명하고, 문제들에 대한 명확하고 상세한 해결책을 제공한다. 그리고 설계패턴들은 재사용 가능한 객체지향 설계들을 생성하기에 유용한 일반적인 설계구조를 추상화함으로써 설계단계의 재사용방법을 도입할 수 있다[4]. 따라서 설계패턴은 과거의 성공적이고 올바른 경험으로부터 얻어진 재사용 가능한 생성물이다. 또한 설계패턴들은 재사용 가능한 시스템 설계구조를 추상화하고 있다. 현재 객체지향 분야에서 가장 각광을 받는 설계패턴은 Gamma가 제안한 것이다[5]. 따라서 본 연구에서 공유패턴(public pattern)설계시 표준화를 지향하기 위해 Gamma의 패턴정보를 이용한다.

3. 패턴지향 객체저장소

3.1 저장소 구축방안

기존의 CASE 도구에서 사용하는 저장소 구축방법에서는 설계정보를 저장할 때 사용자의 설계정보를 모두저장하고 있다. 그러나 본 시스템에서 제안하는 저장소에서는 설계의 중복성을 줄이기 위해 중복되는 설계상의 정보는 상속구조로 저장하게 된다. 따라서 전체 시스템은 설계서를 재사용하기 위해 소프트웨어의 패턴을 추출하여 패턴 저장소에 저장되며, 이때 저장되는 패턴들간의 중복데이터는 객체의 상속 매커니즘을 이용하여 저장하게 된다. 재사용 컴포넌트로 저장된 패턴은 소프트웨어 설계시 객체의 요청이 있을 경우 저장된 패턴을 검색하여 사용자에게 그래픽 정보를 이용하여 사용자가 원하는 패턴-객체를 사용하게 된다. 이때 저장된 패턴의 검색은 개선된 Spreading Activation 방법을 이용하게 된다.



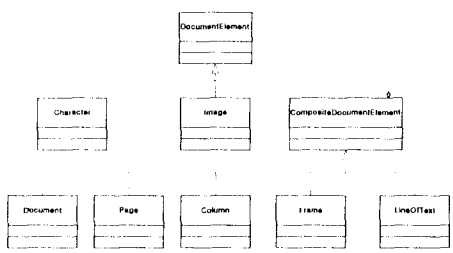
<그림 1> 혼합 정보저장소 구조

저장소에 재사용 정보를 구축한 경우 이를 재사용하기 위한 검색에서 사용자가 원하는 재사용 컴포넌트를 정확히 모르는 경우 사용자가 검색하려는 컴포넌트와 유사한 여러 개의 패턴 정보를 보여줄 수 있도록 <그림 1>과 같이 시스템의 재사용 표준 컴포넌트를 제공하는 공유(public) 저장소가 있게 된다. 그리고 본 시스템은 패턴을 편집하여 저장할 수 있는 패턴 편집기를 제공하고, 저장된 패턴을 바탕으로 설계자의 고유한 설계목적을 이룰 수 있는 고유패턴을 구성할 수 있도록 설계자만의 개인(private) 저장소를 구축할 수 있다. 이와 같이 복합구조의 저장소로 구성하여 시스템의 표준으로 제공되는 공유영역의 패턴-객체를 이용한 재사용 설계와 공유 정보로부터 파생된 사용자 고유의 설계정보를 저장하게 된다.

현재 나와있는 대부분의 CASE 도구들에서 저장소의 컴포넌트를 클래스로 구성하였으나 코드 단계에서의 재사용성을 유도하기 힘들며 반복적인 설계를 하게 되는 단점들을 갖고 있다. 이러한 반복설계의 문제점을 해결하기 위해 CASE 도구에 패턴을 이용함으로써 개발기간을 단축시킬 수 있다.

기본 패턴을 설계하기 위해 다음과 같이 3가지의 패턴으로 분류하여 CASE 도구에 적용될 패턴을 구성한다.

① 개념 패턴(Conceptual pattern)
응용 도메인으로부터 용어와 개념 등의 의미에 의해 기술된 형식을 갖는 패턴으로 제한된 응용 도메인과 동작한다.
② 설계 패턴(Design pattern)
소프트웨어 설계구조의 의미에 의해 기술되는 형식을 취하며 예를 들면 객체(object), 클래스(class), 상속(inheritance), 집단화(aggregation)등이 있다. 디자인 패턴은 Conceptual pattern에 의해 개방된 부분을 설계하거나 보조한다.
③ 프로그래밍 패턴(Programming pattern)
프로그래밍 언어 구조의 의미로써 기술되는 형식을 갖는다.



<그림 2> 객체 패턴화 표현

그리고 <그림 2>객체 패턴화 표현은 워드프로세서를 컴포지트 패턴(composite pattern)으로 표현한 것이다. 이와 같이 저장되는 패턴들은 중복되어 정보가 저장되는 것을 피하기 위해 상속성을 이용하여 정보를 공유하도록

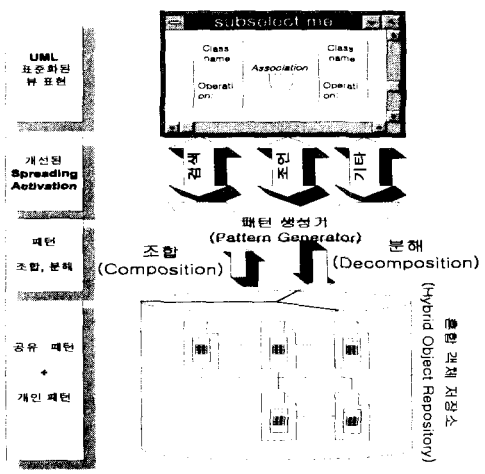
설계한다. 또한 패턴-객체의 확장에 사용하기 위한 다형성으로 새로운 객체의 생성시간을 단축시켜 전체 개발기간을 줄일 수 있도록 정형화된 공유저장소에 저장한다. 따라서 저장소에 저장되는 재사용 컴포넌트는 패턴화되어 저장되며 저장된 컴포넌트간의 구조는 객체간의 관계(relationship)를 표현하는 방식으로 저장된 정보간의 관계에 적용된다.

3.2 시스템 설계

본 시스템은 특별한 문제에 대한 알맞은 pattern의 선택이 아니라 설계와 프로그램의 생성, 재조직화(reorganization)의 발전에서 pattern을 적용하게되는 복합구조의 저장소를 갖게된다. 본 시스템에서 제안하는 <그림 3>의 패턴 생성기(pattern generator)는 다음의 기능을 갖게 된다.

① 상속성에 따른 저장구축방법을 따르고 있으므로 컴포넌트의 조합/분해하여 저장하고 이용할 수 있는 기능을 갖게된다.
② "template" 패턴의 확장된 모임으로부터 얻어진 패턴의 새로운 instance를 위한 프로그램 요소(class, hierarchies 등)를 생성한다.
③ 통합된 패턴(integrating pattern)은 프로그램 요소를 패턴의 역할에 연결함으로써 저장소를 구축한다.
④ 복합구조의 객체저장소에 public과 private를 분리하여 저장한다.

그리고 기존 설계서에서의 재조직화 동작(reorganization operation)에 의한 설계서의 패턴 추출은 역공학(reverse engineering)에 사용될 수 있다. 즉, 프로그램에서 패턴 발생의 문서화와 프로그램의 수정은 패턴의 구조에 더욱 영향을 미치게된다.



<그림 3> 저장소와 뷰 모델의 구조

패턴 생성기에 의해 얻어진 정형화된 패턴은 객체 저장소(object repository)에 분해(decomposition)되어 저장되며 이후 설계 패턴의 재사용(reuse)시에는 결합(composition)에 의해 필요한 패턴을 조직화하게 된다. 분해(decomposition)에 의해 저장될 때 저장소의 정보와 중첩을 피하기 위해 상속관계에 의해 정보를 저장하게 된다. 사용 목적에 따라 객체의 결합에 의해 패턴을 조직화하는 경우 저장되어 있는 패턴을 합성한 복합패턴을 지원하게 된다. 그리고 객체저장소는 시스템의 정형화된 표준패턴을 구성한 공유 영역과 개인적 특성화에 의한 개인 영역의 복합구조를 갖고 있다. 개인 영역의 패턴은 공유 영역의 정규화된 패턴을 다형성에 의해 확장할 수 있는 특징을 제공하게 된다.

4. 결론

본 연구에서는 기존의 CASE 도구에서 제공하는 정보저장소의 특징에 표준화된 설계정보를 제공하는 패턴을 적용하였다. 그리고 저장소 구축방법을 복합구조로 분리하여 시스템의 유연성을 높였으며, 상속개념을 이용한 저장소의 저장방법으로 컴포넌트의 중복저장을 제거하고, 다형성에 의한 설계 컴포넌트간의 복합 컴포넌트를 제공하게 된다. 따라서 CASE 도구에 패턴을 적용하여 설계정보를 표준화하고 이를 재사용 함으로써 재사용율과 확장성을 높일 수 있도록 설계하였다.

향후의 연구는 정형화된 패턴을 분류가 완성되지 않았으므로 공유 영역의 확장을 고려하여야 하며, 도메인 사이의 컴포넌트 확장을 고려한 객체생성기를 설계하여야 한다. 그리고 중복저장을 제거하기 위한 상속관계에 의한 저장방법에서 무결점을 검증하여야 한다.

5. 참고문헌

[1] Dirk Riehle, "Composite Design patterns",OOPSLA '97, pp.218-228.
[2] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerland and Michael Stal, "Pattern-Oriented Software Architecture, A Pattern System", Draft, 1995.
[3] W.Kozaczynski, E.Liongosari, J.Ning, and A.Olafsson, "Architecture Specification Support for Component Integration", IWCASE '95, pp.30-39.
[4] 김치수 "설계패턴을 이용한 객체지향 방법론에 관한 연구" 한국정보처리학회 논문지, 제6권 제6호, 1999, pp.1556-1562.
[5] E.Gamma, R.Helm, R.Johnson, and J.Vlissides, "Design Pattern : Elements of Reusable Object-Oriented Software", Addison-Wesley, 1995.