컴포넌트 랩핑을 위한 설계 패턴의 적용

차정은*, 양영종*
한국전자통신연구원, 소프트웨어공학부 S/W 재사용 연구팀
e-mail:{mary2743, yangyj}@etri.re.kr

Applying the Patterns for Component Wrapping

Jung-Eun Cha*, Yung-Jong Yang*
*Dept of Software Engineering, ETRI

요약

웹 중심의 인터넷 환경은 개인의 정보 처리 뿐 아니라 기업간의 업무 전개를 위한 핵심 기술로 발전해 나가고 있다. 하지만, 조직의 중요한 지식과 프로세스들을 처리하는 시스템들은 대부분 과거(Legacy)의 기술에 의해 개발되어졌으며, 이러한 시스템들은 웹 환경을 위한 분산 아키텍처의 결여와 개방성과 표준화 미흡으로 시스템의 유지보수에 많은 어려움을 가진다. 따라서 재사용성이 높은 레가시 시스템들을 현대적인 기술 요구를 수용하여 향상된 비즈니스 처리가 가능한 새로운 시스템으로 재공학함은 필수적인 일로 여겨진다. 따라서 본 논문에서는 COBOL 레가시 프로그램을 대상으로 컴포넌트 랩핑을 위한 설계 패턴의 적용을 통해 컴포넌트 환경으로의 재공학 방법을 제시한다. 이를 위해 COBOL 프로그램에서 비즈니스 로직 추출을 위한 절차와 방법을 설명하고, 그 결과 생성된 클래스들을 패키징하고 랩핑하기 위해 보편적인 양식으로 이용할 수 있는 재공학 설계 패턴을 나열하고 실 예에 Facade 패턴을 적용한다.

1. 서론

인터넷은 개인 및 조직의 정보 공유 매체일 뿐만 아니라 업무 처리를 위한 핵심 기반 기술로 인식되고 있다. 따라서 웹 중심의 인터넷 환경에서의 정보 시스템의 구축과 운영은 필수적이다. 하지만, 시스템의 핵심 프로세스들은 대부분 과거의 기술에 의해 개발된 레가시 시스템에 포함되어 있으며, 이러한 시스템들은 웹 환경을 위한 분산 아키텍처가 결여되어 있고, 개방성과 표준화 미흡으로 시스템 자체의 유지보수가 매우 어렵다. 또한 사용자가 기대하는 친밀한 사용자 인터페이스를 통해 동적인 의사 결정이 가능한 대화적 방식을 지원할 수 없다. 특히, 웹 아키텍처에 부합하는 컴포넌트 기반 개발(CBD : Component Based Development)에 대한 인식의 보편화는 레가시 시스템을 컴포넌트 기반 웹 환경으로의 변환에 대한 명확한 당위성을 제시한다.

재공학(Reengineering)은 레가시 시스템을 대상으로 새로운 기능적, 환경적 요구를 충족시키기 위해 공학적 관점을 적용하여 새로운 산출물로 접근해 나가는 방법과 절차를 의미한다. 특히 컴포넌트 환경으로의 재공학 방법 중 컴포넌트 랩핑(Wrapping)은 컴포넌트 환경으로의 단일화된 인터페이스를 제공함으로써 레가시 시스템의 변경 없이 빠르고 쉬운 변환이 가능한 방법이다. 특히, COBOL과 같이 프로그램 전체가 하나의 캡슐화 단위가 되는 전역화된 시스템들을 컴포넌트와 같은 독립 단위로 추출, 변환하기 위해서는 여러 클래스들의 공통 API를 통해 외부와 통신하며, 요청된 서비스를 개별 클래스가 독립적으로 제공할 수 있는 랩핑이 최상의 솔루션으로 인식된다.

따라서 본 논문에서는 COBOL 프로그램에서 비즈니스 로직을 식별하여 클래스로 추출하기 위한 절차와 방법을 설명하고, 컴포넌트 랩핑을 위한 레가시 패턴을 나열하며 Facade 패턴의 적용 사례를 제시한다.

2. 관련 연구

2.1 레가시 시스템의 재공학 기술의 분류

재공학은 레가시 시스템에 대한 분석과 재구조화를 통해 다양한 관점의 형상을 식별하고 유지보수 정보를 획득하며, 나아가 새롭게 기대되는 기능적, 환경적 요구에 맞도록 재구축하는 광의의 접근이다.

<표 1> 레가시 변환 기술의 비교

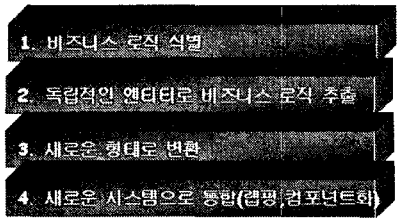
방법		설 명
객체지향 프레임워크 변환	특징	레가시 코드를 분석하고 업무 지식을 추출하며, 자동 변환을 통해 표준화된 객체나 컴포넌트로 패키징
	장점	레가시 분석/모델링에 효율적이며 단력적인 솔루션 제공
	단점	레가시 영역의 제한성 및 자동화에 따른 성능 저하
랩퍼 (Wapper)	특징	컴포넌트로의 변환 위해 OPEN API 인터페이스 제공
	장점	분산아키텍쳐 지원과 레가시 코드로부터 컴포넌트 생성 용이
	단점	제한된 융통성에 의한 유지보수 문제 발생
스크린 스크랩핑 (Screen Scraping)	특징	레가시의 UI를 추출하여 웹 상의 스크린으로 변환
	장점	UI 수준에서 웹으로 변환하기 위한 가장 빠르고 간단함
	단점	실질적 변환이 아니며, 기능성과 융통성이 제한됨
미들웨어	특징	시스템간 매핑 통해 레가시 시스템을 웹과 응용 서버에 연결
	장점	트랜잭션 수준 통합으로 밴더들에 의해 워크플로우가 지원
	단점	다른 코드와 연관해 사용하며, 프로세스 지식 확보가 불가능
EAI (Enterprise Application Integration)	특징	특정 컨넥트를 통한 레가시와 웹 응용 사이의 시스템 통합
	장점	레가시 기능들을 바탕으로 밴더에 의한 제품 지원이 다양
	단점	레가시 시스템과의 중복된 투자와 높은 비용 요구

현재의 재공학 기술은 이슈는 레가시 시스템을 개방적인 웹 환경으로 전환하는 것으로, 크게 두 가지로 분류된다. 첫 번째는 레가시의 기능성과 로직의 흐름을 그대로 유지하면서 새 인터페이스를 통해 이용하는 방법으로 쉬운 기술과 적은 비용이 요구되나, 레가시 시스템의 변경에 매우 민감하여 융통성이 적다. 두 번째는 레가시 시스템의 실제적인 변경을 통해 새로운 시스템으로 재개발하는 방법으로 레가시 시스템의 분석과 수정에 많은 어려움과 비용이 필요하지만, 새로운 아키텍쳐와 표준화에 적응하는 진보된 시스템 생성이 가능해 높은 효과를 제공한다.

<표 1>은 레가시 시스템의 변환을 위한 접근 기술들의 특징들이다[1].

2.2. 레가시 패턴

재공학으로의 접근은 원시 코드의 파싱과 그 결과에 따른 데이터와 제어 흐름 분석에 따라서 특정 문제 영역 내에서 레가시 시스템이 가지는 한정적인 의미와 특정 목적 달선을 위한 특정 역할을 구문적인 해석만으로는 레가시 시스템이 가진 이나 의미를 확보하기가 매우 어렵다. 그러므로, 레가시 시스템의 재공학 과정에서 발생할 수 있는 공통적인 문제들과 그 해결책들을 레가시 패턴을 통해 정의하고 모델 수준의 재사용 요소로서 획득함으로써, 레가시 시스템의 이해를 증가시키고 보편적인 재공학 접근 방식을 정규화 한 설계 수준의 재사용 요소를 생성할 수 있다. 재공학 패턴으로는 일정한 설계 구조가 형성되어 있고 구현을 위한 템플리트 코드는 물론 연관된 다른 패턴과의 상호작용과 그 결과에 대한 상반 관계가 명시되어 있는 Gamma의 패턴들을 이용한다[2].



(그림 1) 비즈니스 로직 추출 절차

3. COBOL 프로그램의 비즈니스 로직 추출

본 논문에서 먼저, COBOL 프로그램을 대상으로 J2EE 플랫폼에서 전개 가능한 컴포넌트로 변환하기 위해 COBOL 프로그램으로부터 핵심 비즈니스 로직을 추출한다. COBOL 프로그램은 객체지향 언어로 작성된 컴포넌트와 달리, 지역 변수(Local)의 사용이 극히 제한된다. 이는 COBOL 프로그램 전체가 하나의 캡슐화 단위를 형성함으로써 컴포넌트와 같이 분리된 기능을 독립적으로 수행할 수 있는 단위로의 분리를 어렵게 한다. 그렇지만, COBOL은 계층적인 변수의 선언과 데이터 중심적인 프로세스 흐름은 파라그래프 단위별 식별에 도움을 준다.

특히, 본 논문에서는, 기존의 레코드 변수들은 응용 도메인 엔티티의 상태를 표현하기 때문에 인스턴스 변수를 위한 좋은 후보가 되며, 파라그래프는 적은 변수를 선언하는 처리문이므로 좋은 메소드 후보가 될 수 있다고 가정한다[3, 4]

3.2 비즈니스 로직 추출 프로세스

본 논문에서는 COBOL과 객체지향 언어의 차이점을 최소화하기 위해 두 단계의 점증적인 접근을 시도한다. 단계 1에서는 기존 COBOL 프로그램의 인스턴스 변수와 메소드 식별하며, 단계 2에서는 객체지향 클래스 형성에 적절하도록 최적화 시킨다. 이를 위해 (그림 1)과 같은 추출 절차를 정의하였다. 이를 위해 일반적인 재공학 및 역공학 분석 기법인 데이터 흐름과 제어 흐름 분석 메소드를 사용하였다.

(1) 비즈니스 로직의 식별 : 프로그램 흐름에서 특별히 의미 있는 변수의 처리나, 흐름에 의사 결정을 포함하는 레가시 코드 부분이다. 이 부분은 사용자의 목적과 영역의 특수성을 반영하기 위해 주관적인 사용자 선택이 이루어지며, 레가시 코드의 규칙을 찾고 Self-Contained 루틴을 식별한다.

(2) 레가시 코드에서 비즈니스 로직 추출 : 식별된 레가시 코드의 데이터을 분석한다. 데이터 흐름에서는 COBOL의 파라그래프에서 정의하고 사용한 레코드 및 하위의 필드 변수를 조사하고, 파라그래프 내에서 참조된 변수의 위치를 찾는다. COBOL은 레코드를 중심으로 하위의 필드 변수를 선언함으로써 코드 상에 선언된

레코드 번호를 중심으로 클래스로 변환하고 하위의 필드는 클래스 에트리뷰트로 선언하며, 파라그래프를 클래스의 메소드로 변환한다. 이 과정에서, COBOL 필드의 변수는 적절한 OOP 타입으로 대체되며, "OCCURS" 구문은 상위 레코드 타입의 배열 형식으로 바꾸어준다. 그리고 프로시쥬어 디비젼의 변수 초기화 부분은 클래스의 초기화 메소드로 변환시키며, 클래스의 전체의 제어를 위해 메인 함수나 start()를 추가한다.

(3) 새로운 형태로 변환 : 추출된 클래스들 간의 상관 관계를 정의한다. 이를 위해 COBOL 파라그래프의 상관 관계를 "PERGORM" 문을 중심으로 파악한다. 따라서 가장 상위 수준에서 정의한 레코드가 변환한 클래스와 그 하위의 레코드들이 변환된 클래스의 연관 관계가 메소드간의 호출 관계를 통해 파악됨으로써, 클래스의 근본적인 기능적 메소드가 도출되며, 이것은 컴포넌트로의 랩핑을 위해 사용된다.

(4) 컴포넌트화 : CBD에서 정의한 컴포넌트 요구를 수용할 수 있도록 변환한다. 이를 위해서는 클래스 자체를 표준 스펙에 맞도록 변경(Transformation)하는 것과 클래스를 그대로 두고 인터페이스를 통해 표준 환경과 통신할 수 있도록 하는 랩핑(Wrapping) 방법이 있다. 랩핑 과정에서는, 호출 관계의 가장 상위 단계에 위치한 클래스의 메소드를 랩핑을 위한 인터페이스로 정의하며, 관련된 클래스들이 가진 메소드들은 랩핑 인터페이스에 의해 제어되는 set_()과 get_(), 그리고 exec_() 메소드를 통해 지정하고 호출함으로써 클래스가 가진 기능성들을 수행한다.

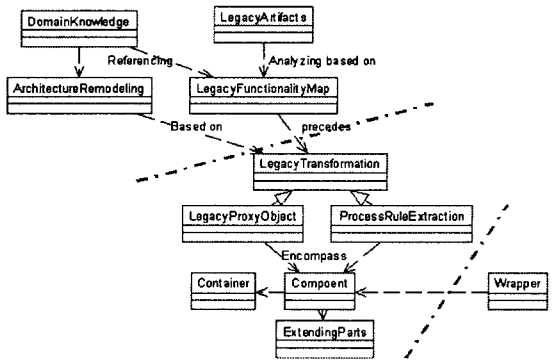
4. 컴포넌트 랩핑을 위한 레가시 패턴 적용

4.1 레가시 패턴 기반의 접근

재공학 과정 중에 영역이나, 특정한 재공학 지식들을 보다 구체화된 형태로 시스템에 반영하고 새고자 한다. 본 논문에서 재공학 과정 과정 중 패턴을

<표 2> 재공학 패턴

유형	유형	후보패턴	특성
GoF패턴	GUI 상호 작용 설계	Mediator	레가시 시스템의 내부 처리 부분과 GUI 부분을 분리하여, 간단한 외부의 인터페이스에 의한 접근을 허용함으로써 레가시 시스템의 복잡한 상호작용을 제거함.
	모듈 간 의 독립적 분리	Mediator Command	기능적인 흐름에서 순차적인 서브 모듈의 집합으로 구성되는 복잡한 레가시 시스템 내의 결합도를 낮춤
	모듈의 계층 관계 설립	Mediator Facade,	구조적인 서브 모듈의 집합으로 구성되는 복잡한 레가시 시스템 내의 결합도를 낮춤으로써 계층 관계 설정
	레가시 인터페이스 조절	Template	레가시 시스템의 처리 모듈에 대한 적절한 인터페이스 제공
	상태 제어	State	프로세스 전이의 상태와 전체 시스템의 상태를 제어하고 모니터링
	통신 인터페이스	Strategy, Template	하나의 인터페이스를 통해 다양한 외부 요구에 대한 응답을 제공할 수 있도록 인터페이스에 의한 패키징 제공
	랩핑 기능	Facade Interface Proxy	레가시 처리 로직을 실행하는 객체의 위치나 구현의 상세항을 숨기기 위한 랩핑 서비스 제공
	동적인 객체 실행	Builder, Factory	재구조화된 객체들이 동적으로 실행될 수 있도록 처리 알고리즘의 캡슐화



(그림 2) 시스템 재공학을 위한 시스템 모델

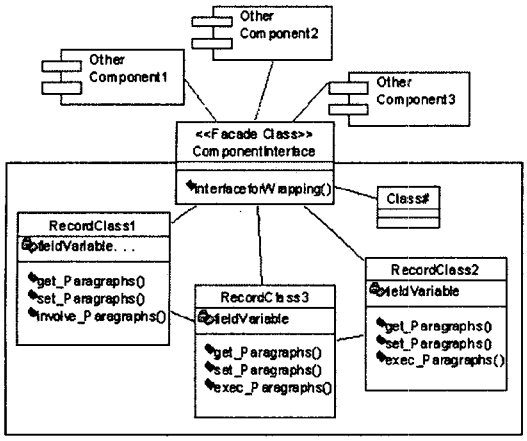
적용하는 목적은 두 가지로 설명된다. 첫 번째는 레가시 시스템이 속한 영역의 컨텍스트(Context) 의미의 이해를 통해 업무 지향적인 흐름 파악을 가능하게 하여레가시 시스템의 이해를 높이고자 한다. 두 번째, 레가시 시스템에서 식별된 핵심 로직들을 재공학하기 위한 구조화된 틀을 형성하고, 정규화된 양식으로 레가시 정보들을 통합, 맞춤하기 위한 경험적인 지침을 제공하기 위해서이다[5].

<표 2>는 재공학 과정에 적용될 수 있는 패턴의 유형을 정리한 것이다. Gamma 패턴들 중 인터페이스 중심으로 컴포넌트들간의 독립적인 상호 연계를 위한 패턴들인 Bridge, Factory Method, Adapter, Mediator 패턴이 재공학의 템플리트 형성을 위해 이용되며, 단일의 추상화된 인터페이스를 제공라기 위해 Facade의 적용으로 레가시 시스템을 제어할 수 있다[6, 7].

4,2 컴포넌트 랩핑

(그림 2)는 레가시 시스템의 재공학을 위해 보편적으로 요구되는 재공학 시스템의 아키텍쳐를 MVC(Model-View-Control) 아키텍쳐에 기반한 3계층 모델로 단순화시켜 표현한 것이다. 이 중 마지막 랩퍼 부분은 외부 인터페이스로부터의 접근을 위한 투명한 접근 방식을 제공하는 역할을 한다. 컴포넌트 랩핑을 위한 패턴으로 Facade 패턴이 가장 적절한 구조를 제시한다.

랩핑은 관련된 여러 클래스들이 하나의 독립적 기능성을 제공할 수 있는 컴포넌트로 패키징하고 가장 핵심적인 인터페이스를 제시하며, 이를 통해 외부 환경과 통신하며, 클래스의 다른 메소드들을 호출하고 지정하는 것이다. 따라서 여러 클래스를 통합하는 인터페이스를 제공하는 Facade 패턴이 가장 적절하다. 즉, COBOL에서 추출된 여러 클래스들 패키징하고, 컴포넌트로서 독립된 서비스를 제공하기 위한 통합 인터페이스로서 대표 메소드를 제시한다.



(그림 3) Facade를 이용한 컴포넌트 랩핑

<그림 3>은 컴포넌트 랩핑을 위한 Facade 패턴 적용의 개념 모델이다. 각 클래스들은 WORKING-STORAGE SECTION의 COBOL의 레코드가 변환한 것이며, 에트리뷰트는 레코드 필드들이다. 그리고 파라그래프들은 메소드로 변환되는데, 그 특성에 따라 데이터 값의 입력을 위한 set_Paragraph()와 출력을 위한 Paragraph() 그리고 호출을 의미하는 exec_Paragraph()d로 변환된다. <<Facade class>>인 ComponentInterface는 레코드의 상관 관계 분석에서 가장 상위의 제어 기능을 가진 클래스로, 외부의 여러 컴포넌트는 이 클래스의 inrerfaceWrapping() 메소드를 통해 컴포넌트의 서비스를 획득하며, 컴포넌트 내의 클래스들은 이 인터페이스를 통해 주어진 명령에 의해 개별 메소드들을 실행한다. 다음은 급여 계산을 위한 컴포넌트로의 랩핑을 위해 Facade 패턴을 적용하여 3개의 클래스를 패키징한 예이다.

```
public class FacadeClient
{
    static public void main(String args[])
    {
        Facade facade = new Facade();
        facade.();

        //Accessable Interfaces from clients
        facade.CalculateSalary();
        facade.CalculateTax();
        facade.CalculateSubtraction();
        . . . . . .
    }
}
class Facade
{
        public Facade() { }

        /* Interface declaration */
        public void CalculateSalary()
        { . . . . . }
        public void ProcessSubTitle()
        { . . . . . }
        public void ProcessSubTitle()
        { . . . . . }

 //Interfave Implementaton in each class
class Salay { . . . . . }
class Tax { . . . . . }
class Subtraction { . . . . . }
    . . . . . .
```

5. 결론

웹 아키텍처와 이에 부합하는 컴포넌트 기반 개발(CBD : Component Based Development)에 대한 인식의 보편화는 레가시 시스템을 새롭게 해석된 목적과 타겟 환경인 컴포넌트 기반의 웹 환경으로의 현대화를 요구한다. 재공학은 레가시 시스템과 목표로 하는 새로운 시스템의 유연한 매핑이 필수적이며, 효과적인 재공학을 위해서는 영역 정보 및 그 전환 단계에서의 정규화된 프로세스 및 지침의 지원이 매우 중요하다.

따라서 본 논문에서는 레가시 COBOL 프로그램에서 사용자가 초점을 두는 비즈니스 로직을 추출하기 위한 프로세스를 정의하고, 그 과정에서 사용되는 몇 가지의 메소드를 서술했다. 그리고 추출된 클래스를 컴포넌트로 랩핑하기 위해 Facade 패턴을 적용하기 위한 개념적 모델을 제시하고, 간단한 자바 코드를 예로 제시하였다. 본 논문에서 제시한 것처럼, 레가시 코드의 컴포넌트화를 위한 재공학 패턴의 적용은 기존 재공학 방법들이 구문적 정보만을 이용함으로써 가지는 불확실성을 극복하고 정규화된 재공학 지침을 확보함으로써, 재공학을 위한 빠른 접근과 시스템 전개가 효율적임을 알 수 있었다.

향후, 레가시 코드의 로직 추출을 위한 정규화된 알고리즘 제안과 재공학 과정에서 패턴 적용을 확대하기 위한 패턴 식별 및 분류에 대한 연구를 진행할 것이다. 또한 아울러 실제적인 구현 컴포넌트로의 매핑을 위해 자바 패턴의 도입을 시도할 것이다.

[참고문헌]

[1]. Hurwitz Group Inc., "Integrating Your Business with the Internet", http://www.hurwitz.com, 1999

[2]. Ali Arsanjani, "Component-based Development and Integration Pattern Language", *PLoP 2000 conference*, http://www.mum.edu/cs_dept/aarsanjani/components/CBDLanguage6.PDF

[3]. Len Erlikh and Mike Ferris, "Business-Rule Extraction from Cobol to Java", http://www.devx.com/premier/mgznarch/javapro/1998/JP_junjul_98/le0698/le0698.asp

[4]. J. K. Joiner, W. T. Tsai[], "Re-Engineering Legacy Cobol Programs", http://www.acm.org/cacm/extension/joinertx.pdf

[5]. William C. Chu, "Pattern Based Software Re-engineering : A Case Study", *Software Engineering Conference, (APSEC '99) Proceedings. Sixth Asia Pacific* , pp. 300 - 308, 1999

[6]. Eric Gamma, *Design Patterns : Elements of Reusable Object Oriented Software*, Addison-Wesley, 1994

[7]. CBDi Forum, "Pattern Catalog", http://www.cbdiforum.com/patterns/index.php3