행위패턴을 이용한 소프트웨어 재사용 프레임워크 구축방법

이 기 오† · 류 성 열††

요 약

소프트웨어 도메인모형(Domain Model)으로부터 행위패턴을 식별하고, 개발 시스템의 동적 행위를 구체화한 사용 사례 추출 및 구조화로 재사용을 증진시키는 소프트웨어 프레임워크 (Framework)구축방법을 제안한다. 대부분의 소프트웨어 행위모형이 이질적인 개발자의 요구사항이나 의도를 일관성 있게 모형화 하지 못하며, 재개발과 유지보수를 위한 구체적인 활용방안이 마련되어 있지 못하다. 따라서, 행위패턴을 식별하고 시스템에 반응하는 구체적인 기능성 항목인 사용 사례(Use Case)를 구조화하여 개발된 모형의 일관성을 보완하고, 재사용과 유지보수를 용이하게 하는 재사용 프레임워크를 구축한다. 사용 사례의 구조화를 위해 격자모형(Lattice Model)이 이용되며, 재사용 구성항목을 추출하도록 유도할 수 있는 재사용 프로세스 구조와 세부절차를 소개한다.

A Construction Method of the Software Reuse Framework using Behavior Patterns

Kee-O Lee† · Sung-Yul Rhew††

ABSTRACT

We propose the software framework construction method that increases reusability through use case extraction and structuring of software system's dynamic behavior of which identifying behavior patterns from software domain models. Most behavior models do not provide a consistent modeling technique for harmonizing user's heterogeneous requirements, and not yet prepared to a detailed optimizing technique for redevelopment and maintenance. Therefore, we need a software reuse framework to support consistency and reusability of existing development models using use cases with functional characteristics. A lattice model is used to this approach for structuring use cases, and the reuse process that can be driven to reusable components is introduced in this paper.

1. 소 개

소프트웨어 개발 모형화는 일관성과 완전성을 증진시키고, 재사용시 개발자의 이해도를 증진할 목적으로 작성되어진다. 하지만 대부분의 개발모형들은 완전성과 재사용성이 부족하며, 시스템과의 상호작용을 모형화한 행위모형도 이질적인 추상화수준을 갖는 다양한 사용자 요구사항을 모형에 반영하지 못하기 때문에 재사용에 많은 어려움이 있다[3]. 그러므로 소프트웨어의 재사용을 증진시키고, 개발모형의 완전성을 증진시키는 방법은 분석 및 설계모형으로부터 재사용에 필요한 인자를 추출하여, 재사용 요구 및 질의발생시 효과적으로 처리할 수 있는 관리체계의 구축과 적용 방법의

† 정 회 원 : 혜천대학 의료정보시스템과 교수
†† 정 회 원 : 숭실대학교 정보과학대학원 원장
논문접수 : 1999년 4월 19일, 심사완료 : 1999년 7월 7일

개발을 통해서 가능하다.

세부 연구방법으로는 소프트웨어 재사용 관련연구로 도메인 모형, 객체지향 개발기법, 재사용 설계 패턴에 대한 기초연구를 수행하여, 도메인 모형과 사용 사례 분석을 연계한 재사용 프레임워크의 필요성을 유도한다. 재사용 프레임워크의 구축은 도메인 모형에서 추출된 행위패턴을 사용 사례로 구체화하며, 구체화된 사용 사례를 재사용에 이용하기 위한 재사용 참조구조를 구축한다. 재사용 참조구조는 다양한 사용 사례의 추상 계층성을 표현하고 있기 때문에, 이질적인 개발모형의 완전성을 향상시킨다. 따라서, 3장에서는 재사용 프레임워크 구축을 위한 재사용 프로세스 구조와 세부절차가 기술된다. 또한, 재사용 대상이 원시코드일 경우는 코드의 기능성을 분할하여 추상화할 수 있는 PSDG(Post-State Dependency Graph)가 적용되어진다.

따라서, 도메인 분석모형과 연계한 사용예의 추출과 재사용 참조구조의 생성, 소프트웨어 재사용을 위한 재사용 항목추출 등의 재사용 모형화는 기존의 객체지향 개발 방법론이 순차적인 단방향성 산출물 생성에 치중함으로써[5] 발생될 수 있는 일관성, 완전성 부족 및 재사용 콤포넌트 추출의 어려움을 극복할 수 있게 된다. 또한, 행위패턴들로부터 추출한 사용 사례들은 시스템의 내·외부 기능성 항목들이기 때문에, 서로 다른 프로젝트의 개발 방법론이 이질적인 모형일 지라도 재사용성 증진에는 영향이 별로 없다.

2. 소프트웨어 재사용 관련연구

소프트웨어의 재사용은 초기의 재사용 요구사항 인식으로부터 재사용 가능한 컴포넌트의 추출까지 많은 시간과 노력이 투입된다. 효과적인 재사용을 위해서, 소프트웨어에 대한 원시코드 뿐만 아니라 문서정보 등을 효과적으로 이용해야 하며, 기계적인 재사용 처리 과정과 정보시스템에 대한 지식이 필요하다. 대부분의 재사용 관련 연구에 의하면 소프트웨어 재사용을 위해서 도메인 지식이 유용하게 사용되며, 도메인 지식은 정형 또는 비정형의 모형이나 전문가의 경험지식 등을 일컫는다[1]. COREM은 캡슐지향의 재공학 방법으로, 절차언어를 객체지향 언어로 변환하는 2단계 변환구조를 가지고 있다[3]. RIGI는 소프트웨어 시스템을 요약, 질의, 표현 및 가시화, 평가의 단계를 거쳐 공간적으로 나 시각적으로 관계를 구체화하여 재 구조화하는 추상화 지향의 방법이다[10]. 또한, 최근의 연구로는 국내 보고서 중에 이들 모형의 장점을 이용하여, 객체지향 모형을 반복 정재(Refine)하는 기법인 ORT(Object-orient model Refinement Technique)를 통해서 역방향과 순방향 모형을 비교하여 일관성을 높이는 역공학 방법도 소개되고 있다[11].

이러한 방법들의 공통적인 추세는 도메인 모형화와 객체지향화의 특성을 갖고 있다는 점이며, 최근의 연구에 의하면 소프트웨어의 설계패턴과 프레임워크 생성기법이 지속적으로 연구되고 있다[2]. 이 방법에 의하면 설계패턴을 크게 생성, 구조, 행위로 분류하고 각 패턴을 이용하여 개발에 적용하는 연구들이 소개되고 있다. 따라서, 도메인 분석이나 객체지향 개발방법론의 세부기법을 기반으로 설계패턴을 고찰하여 개발에 활용하는 방안과 재사용을 극대화하기 위한 방법 및 절차가 마련된다면 소프트웨어 개발비용과 효율성이 증대되리라 기대된다.

2.1 도메인 모형

Neighbor에 따르면 도메인은 문제 영역이며, 도메인 분석은 "도메인에 대해 객체와 연산자 그리고 도메인 전문가가 중요하다고 여기는 것들 간의 관계를 식별하려는 노력"이다[10]. 기존의 시스템 분석과는 하나의 독립 시스템이 아닌 여러 시스템이 대상이 된다는 점에서 다르다. 도메인 모형은 구현된 모형이나 개발중인 모형에 대한 환경이나 개발자의 의도를 파악할 수 있으며[12], 재사용이나 유지보수와 같은 변경에 대한 이해의 기초자료로 이용될 수 있다. 도메인 모형화는 역공학자에게 코드나 문서를 통해 예상 가능한 정보를 제공하는데, 데이터 구조나 알고리즘 등과 같은 원시코드에 대한 논리적 흐름을 알 수 있도록 하여주며, 문서정보를 통해서 전체구조나 자료의 변환 및 기능성을 예측할 수 있도록 해준다.

2.2 객체지향개발 세부기법

객체지향 개발은 절차형 중심의 개발 모형들이 갖는 기능중심의 표현과는 달리, 실세계의 대상들을 그대로 분석과 설계에 반영하는 특징이 있으므로, 개발의 여러 단계를 거치면서 발생하는 모형의 이질성을 동질화시키고 재사용성과 독립성을 극대화한 개발 방법론으로 평가된다. 대표적인 개발방법론인 OMT(Object Mod-

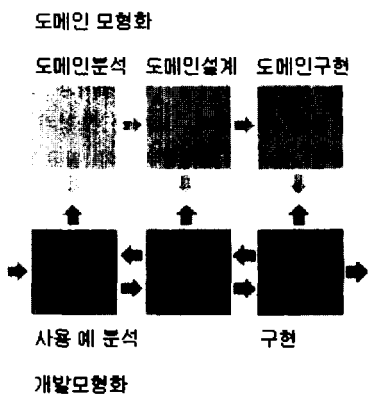
eling Technique)와 UML(Unified Modeling Language)을 비롯한 다양한 방법론들이 있으며, 각각의 방법론마다 강조하는 개발모형의 관점이 서로 다르다는 것은 이미 잘 알려져 있다[2].

하지만, 대부분의 객체지향 방법론들이 순차적인 개발 산출물 생성에 편중되어 있으며, 요구사항을 모두 만족하는 완전한 개발모형 작성은 불가능하다. 또한, 산출된 개발모형들은 재개발의 입력으로 사용되어지지 못하고 있으며, 모형의 대부분은 재사용 콤포넌트로 사용되지 못하고 있다[5]. 따라서, 개발모형 자체를 재사용의 대상으로 삼거나 새로운 개발시스템에 이용하는데는 경험을 바탕으로 한 많은 도메인 지식들이 필요하며, 적절한 형태의 재사용 인자를 추출하는 부가적인 작업이 필요하다. 이러한 부가적인 작업은 시스템에 반응하는 자극이나 활동과 같은 동적 행위와 내부활동들을 모형화 할 수 있는 Jacobson의 사용 사례 분석 기법[6]이나 반응중심의 개발방법론의 모형화 원리를 적용함으로써 가능하다. 따라서, 도메인 모형에서 사용 사례와 같은 설계패턴을 식별하는 기술이나 이를 재사용에 이용하는 기술 등이 필요하다.

2.3 재사용을 위한 설계패턴

재사용을 위한 설계패턴은 도메인에 대한 연구와 객체지향에 대한 연구가 선행되어져야 하며, 객체지향개발의 특정기법과 연동 하는 재사용 중심의 설계패턴이어야 한다. 이미, 앞 절에서 기술한 바와 같이 설계패턴은 생성, 구조, 행위로 세분화되어지며, 생성과 구조는 개발의 유형에 따라 각각 적용형태가 달라지기 때문에 재사용하기 어려운 요소로 평가되어진다. 따라서, 기능성 항목이나 독립적인 정보와 같은 행위패턴을 재사용의 대상으로 삼는 것이 바람직하다.

(그림 1)은 도메인 분석과 도메인 설계 그리고 도메인 구현 모형들이 객체지향 개발기법의 사용 사례 분석기법과 상호 연계되어, 재사용 요구를 포장단계의 컴포넌트까지 세분화시키는 단계별 처리구조를 보여주고 있다. 도메인 분석과 사용사례 분석의 상호 연계과정을 통해서 재사용 요구가 구체적인 재사용 컴포넌트로 변환되어짐을 알 수 있다. 이러한 변환 과정을 재사용 모형화 과정으로 이해할 수 있으며, 모형화 세부 전략과 방법 및 기법을 수립하여 새로운 소프트웨어 개발 프로세스와 연동할 수 있는 방법론을 구축한다.



(그림 1) 도메인 모형화와 사용 사례 분석

3. 소프트웨어 재사용 프레임워크

3.1 구축 전략

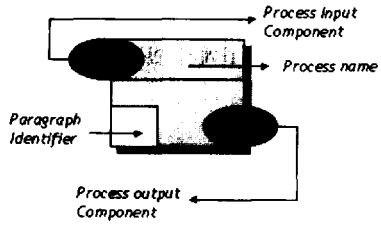
재사용 프레임워크를 구축하기 위한 전략으로 첫째, 문서정보와 원시코드 정보를 효과적으로 활용하기 위한 도메인 모형의 활용방안으로, 재사용 프레임워크 구축의 초기입력을 도메인 모형으로 설정한다. 둘째, 객체지향 개발기법 중에서 Jacobson의 Use Case Modeling에 사용되는 사용 사례와 같은 기능성 행위패턴을 프레임워크 구축의 기초자료로 이용한다. 셋째, 프레임워크 구축은 재사용이 용이한 구조성과 변경에 대한 확장성을 지원한다. 넷째, 원시코드에 대한 재사용 프레임워크는 기능성 분할과 추상화의 원리를 바탕으로 한다. 본 연구는 적용환경의 이질성과 실행시간 종속적인 인자와 같은 성능요소 및 잠재하고 있는 위험성 항목과 같은 의미적인 환경인자 등은 재사용 프레임워크 구축의 범주에서 배제한다.

3.2 소프트웨어 재사용 프로세스 구조설계

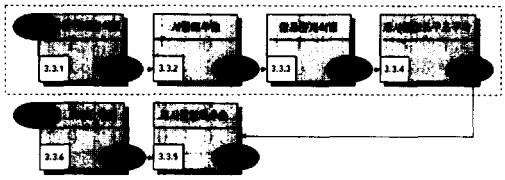
소프트웨어의 재사용을 위한 구조 및 프로세스를 설계한다. (그림 2)의 사각형은 프로세스를 의미하며, 사각형안의 조그만 사각형(Paragraph Identifier)부분은 해당 프로세스가 언급된 논문의 장 절 구분번호가 기술되는 부분이고, 상단의 좌측 원은 프로세스 입력항목을, 하단의 원은 프로세스 출력 항목을 의미한다.

(그림 3)은 재사용 대상이 문서화 정보일 때는 도메인 모형으로부터 문서정보를 입력하여 행위패턴을 식별하고 사용 사례를 추출하는 과정이며, 사용 사례를 기반으로 한 요구사항과 사용 사례와의 이진 참조구조

를 바탕으로 재사용 참조구조를 생성해 내는 과정, 그리고 클래스 설계 및 클래스구조 구축과정이 있다. 그리고 재사용 대상이 원시코드일 경우 재사용 가능한 기능성 모듈을 만들어 내는 PSDG 적용과정이 있다.



(그림 2) 소프트웨어 재사용 프로세스 모형구조



(그림 3) 소프트웨어 재사용 프로세스 구조

(그림 3)은 소프트웨어 재사용 프로세스간의 구조적인 연계성을 표현하고 있는 그림으로, 그림의 상위 4개 프로세스에 파선이 표시되어 있는데, 이는 본 연구에서 핵심으로 취급하는 프로세스들로써 연구의 범주를 제한하고 있다. 또한, (그림 2)의 모형구조에서 입력과 출력 항목이 표시되도록 되어 있으나, 프로세스간 출력요소가 그대로 입력으로 전이되기 때문에 중복성 배제 측면에서 초기의 입력을 제외한 입력요소는 생략되었다. 그림에서 출력요소인 B.Pattern은 행위패턴의 약어이며, Use Case는 사용 사례를 C.R.Table은 상호참조테이블을, R.R.St.는 재사용 참조구조를 각각 의미한다. 결국, 도메인 모형으로부터 재사용 참조구조를 구축함으로써 재사용 프레임워크가 만들어지고, 원시코드는 PSDG를 통해서 추상화 및 조각화 되어진 후 재사용 항목(Component)로 재구성되어진다. 본 연구는 PSDG나 재사용참조구조로부터 재사용 항목을 추출하는 세부방법은 소개하지 않고 있다.

3.3 프로세스 정의

3.3.1 행위 패턴식별

행위 패턴식별(Behavior Pattern Identification)은 <표 1>처럼 사용 사례 구조화에 이용될 수 있는 사용 사례를 추출하는 단계로, 생성, 구조, 행위라는 문서화 정보가 갖는 설계패턴 중에서 기능성을 표현하고 있는 행위패턴을 선별하는 단계이다. 행위패턴에는 반응연결(Chain of Responsibility), 중재자(Mediator), 저장(Memento), 관찰자(Observer), 방문자(Visitor)와 같은 객체간 또는 객체내부의 상호접속과 관련된 행위패턴이 있으며, 명령(Command), 반복(Iterator), 상태(State), 전략(Strategy) 등이 있다.

이상과 같은 행위패턴으로부터 재사용에 이용될 수 있는 사용 사례는 사건, 상태객체, 사용자 접속과 같은 프레임워크, 알고리즘과 같은 것들이다. 즉, 이들 문서정보는 결국 구문이나, 구조, 기능, 행위와 같은 시스템의 서로 다른 측면을 반영하고 있으며, 이들 서로 다른 측면의 모형들은 시스템 설계시에 통합되거나 세분화되어 진다. 그러므로 다양한 문서정보로부터 행위패턴을 식별하여 사용 사례로 이용될 수 있는 모형 구성항목을 추출해내는 것이 중요하다.

3.3.2 사용 사례 추출

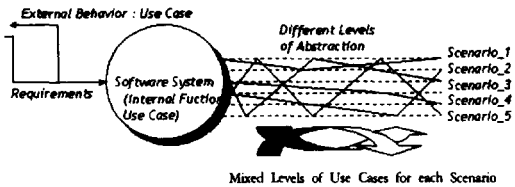
사용 사례는 <표 1>에서와 같은 기능성을 바탕으로 한 행위유형들로 사건이나 상태객체와 같은 객체지향

〈표 1〉 행위패턴 및 사용 사례

행위패턴 / 분석결과	반응연결	중재자	저 장	관찰자	방문자
사용의도 (Intents)	전송/수신분리	명시적 객체 참조 방지	객체 내부상태 포착 및 기록	객체상태변화 따른 종속성 표현	객체함수 추가시 객체에 영향 안줌
적용성 (Applicability)	시간 순서화	복잡한 객체 참조 구조해결	객체상태저장 및 상태 재사용	객체간 종속성 표현	객체에 영향없는 함수들 클래스화
실 사례 (Known Uses)	MacApp, E++ 사건제어기	E++, THINK C class library	Dylan's 반복자 인터페이스	SmallTalk MVC InterViews	IRIS 3차원그래픽 처리용 Inventor
사용 사례 (Use Case)	사건(Event)	사건관리자 (Event Handler)	상태객체, 반복인자 상태	사용자접속 프레임워크	알고리즘, 특정행위(Action)

개발 모형의 동적 모형화에서 추출되는 모형 구성항목이다. 이들 사용 사례들은 하위수준(Micro-level)에서 객체간에 발생하는 메시지나 사건, 객체의 다양한 상태전이, 특정행위, 특정기능 등을 표현하고 있기 때문에, 재사용을 위한 항목추출이 용이하다. 하지만, 이들 항목 개개의 추출이 이질적인 추상화 수준으로 인한 재사용의 효율을 저하시키기 때문에[7] 적절한 계층화 및 구조화를 통해서 재사용의 효율을 극대화시킬 수 있다.

<표 1>로부터 대부분의 행위패턴들이 사건이나 메시지 같은 객체나 시스템의 동적 행위를 사용 사례로 분류할 수 있기 때문에, 도메인내의 상태전이도나 사건흐름도와 같은 행위모형들이 재사용을 위한 사용 사례 추출의 도구로 이용된다. 하지만, 이들 행위모형들은 사용자의 관점이나 자극의 유형에 따라 나타나는 반응의 유형이 서로 상이하며, 다양한 추상화 수준에 따라 표현되어지는 사용 사례가 각각 다르기 때문에 이들 사용 사례들을 곧바로 재사용에 적용하는 데에는 많은 문제점들이 있다. (그림 4)는 소프트웨어 시스템에 반응하는 다양한 시나리오가 서로 이질적인 추상화 수준을 갖고 있음을 표현하고 있다. 따라서, 개발모형의 일관성과 정확성은 이질적인 추상화 수준을 극복할 수 있는 개발 방법론이 수반되지 않는 한 기존 시스템에서는 기대하기 어렵다.



(그림 4) 사용 사례의 서로 다른 추상화 수준표현

3.3.3 참조관계 식별

소프트웨어의 재사용을 위해 추출된 행위패턴의 사용 사례들은 객체간의 상호접속이나 내부기능성, 접속구조와 같은 기능성 항목들을 나열한 것이므로, 사용 사례 기반의 재사용 참조구조는 객체간의 상호작용을 잘 표현하고 있는 사건추적도(ETD)나 상태전이도(STD)를 사용 사례 추출을 위한 대상모형으로 삼는다.

<표 2>에서, 추상요구사항 R1R2는 개별 요구사항 R1과 R2 모두를 포함하는 추상화된 요구사항을 의미하므로, 사건(E)는 R1과 R2사이의 AND(∩) 조건으로 표현되며, 요구사항이 증가할 때마다 이들 요구사항을 모두 만족하는 사건들은 점점 작아져 공집합(∅)이 되어진다. <표 2>에서는 편의상 R1-R4에 대한 서로 다른 추상화수준 대표로 3가지, 예를 들어 R1R2, R1R2R3, R1R2R3R4만을 기술하였다. 또한, 요구사항과 사용 사례와의 관계에서 1은 관계가 있음을 의미하며, 0은 관계가 없음을 의미한다.

〈표 2〉 요구사항과 행위패턴 사용 사례 기능성 항목과의 이진관계

요구사항 \ 사용예	사건 :E1	사건 :E2	사건 :E3	사건 :E4	사건 :E5	사건 :E6	사건 :E7
요구사항:R1	1	1	1	0	1	1	0
요구사항:R2	1	1	0	1	0	0	0
요구사항:R3	1	1	0	1	1	1	1
요구사항:R4	0	1	1	1	1	1	1
추상요구사항:R1R2	1	1	0	0	0	0	0
추상요구사항:R1R2R3	1	1	0	0	0	0	0
추상요구사항:R1R2R3R4	0	1	0	0	0	0	0

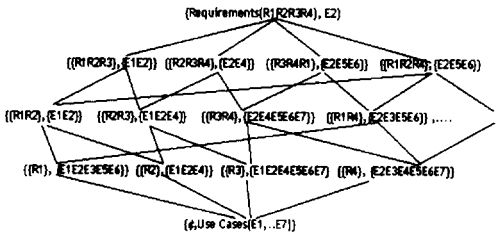
3.3.4 재사용 참조구조 구축

사용예 기반의 구조화기법은 다양한 시나리오를 갖는 사용 사례 기반 모형내의 기능성 항목들을 재 구조화하여 기존 분석모형의 완전성을 향상시킬 뿐만 아니라, 향후의 재사용 시 기능성 항목을 쉽게 추출하고 이용할 수 있는 장점을 갖는다. 재사용 사례 구조화의 산출물인 재사용 참조구조는 다음과 같은 격자(Lattice)형식을 취하고 있으며, 객체지향 소프트웨어의 상속성이나 일반화와 같은 계층구조를 지원하는 관리모형의 형식을 취하고 있다.

이와 같이, 재사용을 위한 자원 참조구조는 요구사항과 기능성과의 상관관계를 구조로 표현한 것이다. 즉, 요구사항이 구체화되지 않은 추상화 수준의 포괄적 요구사항에서부터 구체적인 요구사항에 이르는 상세 요구사항까지의 관계를 구조화 한 모형이다.

(그림 5)는 <표 2>에 대한 재사용 참조구조를 표기한 것으로, 여러 수준의 요구사항들과 추출된 사용 사례들간의 관계성이 방사형의 참조구조를 나타내고 있다. 예를 들어, 하위(수준 0)에서 3번째 추상화 수준의 요구사항(수준 2)인 R1R2는 사용 사례 E1과 E2에 반응함을 의미하며, 추가로 표현되어지는 다른 요구사항들은 다시, 상위의 추상 요구사항이 되어지면서, 더욱

좁은 범위의 사용 사례들에 반응함을 알 수 있다. 결국, (그림 5)는 최상위 수준의 추상화 요구사항이 세분화되어질수록 다양한 사용 사례에 반응하며, 다양한 요구를 만족하는 공통의 요구사항 일수록 반응하는 사용 사례가 적어지는 방사형 대응관계가 재사용 요구사항과 사용 사례간에 존재함을 표현하고 있다. 이러한 관계는 수학적 격자(Lattice)모형을 통해서 구현가능한데, 격자의 특성상 다양한 관계들이 수학적으로 표현가능하며[8], 표현된 관계는 쉽게 검색 및 변경이 가능하기 때문에 재사용 참조구조로는 적합함을 알 수 있다.



(그림 5) <표 2>에 대한 재사용 참조구조

이 모형 구조내의 기능성은 요구사항이 상세화 되어지면서 포괄적인 기능성 이외의 추가적인 기능성이 포함되어 지기 때문에, 객체지향 소프트웨어의 클래스 상속구조와 같은 성격을 지니게 된다.

(1) 재사용 참조구조의 유용성

이와 같은 다양한 종류의 요구사항에 대하여 추상화 수준에서부터 상세화 수준에 이르는 관계성 구조의 구축은 다음과 같은 특징을 지닌다. 첫째로, 모형구조 범주내의 개별 요구사항들의 조합과 같은 복합 요구사항 발생 시, 필요한 기능성들을 계층구조로부터 상속받아 이용할 수 있기 때문에 이미 정의된 기능성들을 곧바로 재 사용할 수 있으며, 재사용자의 임의적인 기능추가가 가능하다. 둘째, 모형구조 범주 밖의 새로운 요구사항 발생 시 자원참조구조의 재구축이 용이하기 때문에 객체지향 소프트웨어 구조의 장점을 최대한 활용할 수 있다. 셋째로, Use Case모형의 특징상 다양한 활동자(actor)의 자극에 따라, 시나리오와 이를 구성하는 이벤트들의 추상화 수준 차이로 개발모형의 일관성과 정확성을 감소시킨다. 하지만, 자원참조모형인 격자망에서는 이벤트의 추상화 수준이 재사용 요구사항의 추상화 수준과 반대로 구축되어 있으며, 다양한 이벤트, 즉 기능의 추상화 관계성이 형성되어있다. 따라서, 기존 소프트웨어의 개발 모형의 기능모형이 표현하지 못하는 추상화 기능의 표현으로 모형의 완전성을 보완할 수 있다.

(2) 재사용 참조구조 구축 알고리즘

```
Begin
  Set Events or Messages as Usecases;
  Identify Reuse Requirement(RR) to each Usecase(U);
  AR:String; UAR:Boolean;
  j=i+1;
  Repeat
    Repeat
      AR(Abstract Requirements l:k)= RR(i)+RR(j);
      UAR(Usecase of AR) = URR(i):Usecase of RR(i)
          ∩ URR(j):Usecase of RR(j);
      if (UAR(k)==false) then
          AR(l:k)=ϕ;
          k=k+1;
      else if (i==n) then
          l=l+1;
      end_if
    Until(i<n-1)
  Until(j<n)
  While(l<n) do
    While(UAR(k)==true) do
      Writeln(AR(l:k));
      k=k+1;
    End_while
      l=l+1;
  End_while
End_begin
```

알고리즘은 사용 사례와 요구사항간의 이진관계가 참인 경우, 요구사항들을 결합한 추상요구사항(AR[l:k])과 참 또는 거짓의 관계성 정보를 유지하는 UAR(k)에 값을 채우며, 비교가 끝나면(i==n) 추상화 수준(l)을 한 단계 증가시킨다.

3.3.5 재사용 항목 추출

재사용 항목추출은 재사용 참조구조 구축 알고리즘에 의해 생성된 격자망으로부터 상속성 계층구조로 재구조화하는 단계로, 다양한 사용 사례들이 클래스의 메소드로 표현되어진다.

클래스내의 메소드들은 격자에서 표현된 기능성 항목이며, 상위로 갈수록 클래스가 추상화되어지기 때문에 관련 메소드들도 공통항목에 해당하는 최소한의 메소드만을 유지하게 된다. 이상의 Java 원시코드에서 볼 수 있듯이 다중 상속관계가 형성되어지며, Java언

어에서는 다중상속을 단일상속관계로 표현하도록 유도한다. 따라서, 단일상속관계로 재 구조화하거나, 대상 언어가 C++ 언어인 경우 다중상속을 그대로 표현하면 된다. 하지만, 세부 구조화 방법에 있어서, 다형성 및 다중상속시의 충돌문제 등이 문제점으로 지적될 수 있지만, 이는 대상 언어에 종속적인 부분이므로 각 언어에 따른 해결방법이 존재하며, 본 논문에서는 자세히 다루지 않는다. 즉, C++언어의 경우 다중상속시의 충돌문제가 야기되면 이에 대한 추가적인 문제해결 기법이 이용된다

```
1   import java.io.*;
2   import java.applet.*;
3
4       {
5       public void E1();
6       }
7   class R1R2R3 extends R1R2R3R4
8       {
9       public void E1();
10      public void E2();
11      }
12  class R2R3R4 extends R1R2R3R4
13      {
14      public void E2();
15      public void E4();
16      }
17  class R3R4R1 extends R1R2R3R4
18      {
19      public void E2();
20      public void E5();
21      public void E6();
22      }
```

(그림 6) (그림 5)에 대한 최상위 수준의 Class 구조화

3.3.6 PSDG

후상태 종속성 그래프(PSDG)를 이용한 확장모형화는 의미분할을 응용한 기법으로써 선조건에 대한 후기상태를 종속성 관계로 표현한 그래프에 형식화와 강조기능을 첨가한 확장모형이다[14]. 후상태 종속성 확장모형화의 기본적인 골격은 기존의 프로그램 종속성 그래프(PDG)가 프로그램의 문단 구조로부터 하향식 문장종속성을 표현함으로써, 효율적이지 못한 대상 프로그램의 경우 PDG자체가 복잡할 뿐만 아니라 이를 단순화시키는 그래프 축소기법(Graph Reduction)기법의 적용이 매우 어려웠다. 이러한 문제점을 인식하고, 프로그램 실행시의 최초 단일문장의 초기 상태에서부터 마지막 상태에 이르는 경로를 후 조건 검색 알고리즘에 의하여 그래프로 표현한다. 표현된 그래프의 각 노드는 문장내의 변수들로써, 실행시에 영향을 미치는 다음문장의 변수에 종속적인 관계를 가지고 있으므로, 이에 그래프로의 표현이 가능하다. 이는 소프트웨어 공학의 상태전이도나 Petri-net과 같은 도구를 이용하여 표현가능하며, 프로그램 구조의 정적인 구조뿐만 아니라 동적인 활동성을 추적할 수 있다. 이렇게 완성된 그래프는 사용자의 경험이나 알고리즘의 지침에 의하여 부분적인 조각화(clustering) 및 이진 그래프로의 표현 및 중요한 부분의 강조 그리고 잘못된 모듈을 표시할 수 있는 기능을 제공한다.

4. 재사용 프레임워크 구축 실험예제

4.1 예제적용

When a user swipes his security card through a card reader, the system checks to see if that user is authorized to pass through the corresponding door. If so, a security event is logged, and the door is unlocked. If not, a security violation is logged, and the door will remain locked.

(1) 각 시나리오별 사용 사례 목록 : Doors, Locks, and Security Card Readers

Scenario 1 : Card Swipe Opens Doors
- Event : Swipe
- Event : Access Request
- Event : Security Event
- Event : Open
- Event : Unlock

Scenario 2 : Door Refuses to Unlock
- Event : Swipe
- Event : Access Request
- Event : Security Event
- Event : Open
- Event : Door Failure

Scenario 3 : Break-in Detected
- Event : Break-in
- Event : Security Alarm

Scenario 4 : Fire or Smoke Detection
- Event : Fire Alarm
- Event : Security Alarm

(2) 사용 사례의 이진관계

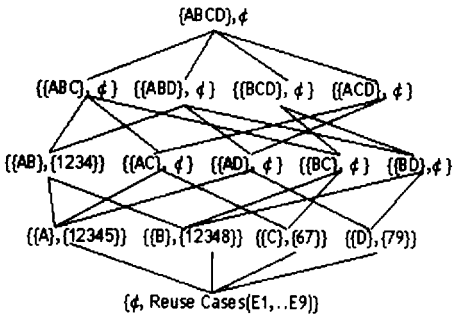
<표 3> 참조

〈표 3〉 실험예제에 대한 요구사항과 사용 사례간의 이진관계 테이블

사용 사례 번호 / 요구사항	Swipe	Access Request	Security Event	Open	Unlock	Break-in	Security Alarm	Door Failure	Fire Alarm
	1	2	3	4	5	6	7	8	9
Scenario_1:A	1	1	1	1	1	0	0	0	0
Scenario_2:B	1	1	1	1	0	0	0	1	0
Scenario_3:C	0	0	0	0	0	1	1	0	0
Scenario_4:D	0	0	0	0	0	0	1	0	1

(3) 재사용 참조구조 구축

(그림 7)은 요구사항(A-D)에 대한 추상화에 따른 사용 사례 표현 및 재사용 참조를 위한 계층구조를 보여주고 있다.



(그림 7) <표 3>의 재사용 참조구조

4.2 재사용 프레임워크 평가

소프트웨어의 재사용을 위한 프로세스와 개념적인 구조를 제공하는 CFRP (Conceptual Framework for Reuse Processes)가 STARS에 의해 제안되었다. 재사용 프로세스는 도메인 엔지니어링과 어플리케이션 엔지니어링을 결합하고, 여기에 지속적인 학습과 재사용 절차가 적용되어지는 "Plan-Enactment-Learning"의 3단계 프로세스가 반복되어지면서 재사용을 완성해 나가는 개념적인 구조화 모형이다[13]. STARS 프로젝트는 이와 관련된 연구로 ROSE(Reuse Oriented Software Evolution model)나 ODM(Organization Domain Modeling method)이 있으나, 이는 점진적인 프로세스 성숙도 모형에 가깝다. 프레임워크나 구조가 재사용의 핵심도구로 이용된 DSSA(Domain Specific Software Architecture)는 도메인 모형과 참조구조 사이의 연결을 도메인 전문가에 의존하고 있는 의미적 재사용 모형이다.

본 연구는 이들 재사용 프레임워크 관련 연구들이 도메인 모형과 연계한 프로세스 모형화에 치우친 진화 모형이거나, 도메인 전문가에 의존하는 프레임워크 구축방법들이었던 반면, 다양한 개발 모형의 산출물들로부터 기능성 항목인 사용 사례를 추출하여 재사용 참조구조라는 프레임워크 구축까지의 전 과정을 프로세스로 표현하였다. 각 프로세스의 산출물들은 유기적인 변환과정을 통해서 자동화 가능한 재사용 참조구조를 구축하게 하여, 다양한 재사용 요구에도 적절한 재사용 자원을 이용할 수 있는 재사용 구조 구축에 기여했다.

5. 결 론

도메인 모형에서 행위패턴을 식별하고, 세부 사용 사례를 추출하여 소프트웨어 재사용을 위한 프레임워크 구축방법 및 세부 프로세스를 제안하였다. 대부분의 행위 개발모형이 일관성과 완전성을 별도의 방법으로 검증하여야 하며, 재사용 시 대부분의 개발모형은 유용성을 상실한다. 따라서, 개발모형의 행위패턴으로부터 세부 사용 사례라는 기능항목을 추출하여 프레임워크로 구조화함으로써 재사용을 원활히 할 수 있는 재사용 참조구조를 구축하였다.

재사용 참조구조는 사용 사례를 요구사항과 연계한 상속성 계층구조로 표현하고 있으며, 추상화 수준에 따른 재사용 항목의 검색 및 수정 보완이 용이한 수학적인 격자모형을 바탕으로 하고 있다. 재사용 프로세스에서는 재사용 대상이 원시코드인 경우에 대하여 기능분할 및 추상화 기법인 PSDG를 이용한 기능성 항목을 추출함으로써 개발모형의 기능모형과 연계한 모형검증이나 모형보완 방법으로 이용 가능하다.

재사용 참조구조의 성능이나 효율성은 검증하지 못하였으나, 소프트웨어 모형의 일관성과 완전성 향상을 기할 수 있는 재사용 방법론 수립 및 도구개발을 위한 기초연구로서 기여하였다. 또한, 모형검증 및 보완을 위한 방법은 향후 연구과제로 설정하였다.

참 고 문 헌

[1] Biggerstaff. T, "Design Recovery for maintenance and reuse," IEEE Computer, 22(7):36-49, July 1990.

[2] Gamma. E et al, Design Patterns, Elements of Reusable Object-Oriented Software, Addison Wesley, 1997.

[3] Gall. Herald, "Object-Oriented Re-Architecturing," 5th European Software Engineering Conference (ESEC'95), 1995.

[4] Gallagher, K.B.and Lyle, J.R., "Using Program Slicing in Software Maintenance," pp.751-761, IEEE Transactions on Software Engineering, 1993.

[5] Godin. R, et al, "Learning Algorithms using a Galois Lattice Structure," In proceedings of the

3rd International Conference on Tools for Artificial Intelligence, San Jose, CA, pp.445-475, IEEE 1991.

[6] I. Jacobson et al, Object-Oriented Software Engineering, A Use Case Driven Approach, Addison Wesley, 1990.

[7] Kimbler. Kristofer et al, "Use Case Driven Analysis of Feature Interaction," Department of Communication Systems, Lund Institute of Technology, Box 118, 221 00Lund, Sweden.

[8] Noureddine Boudrigua et al, "The Lattice of Specifications: Applications to a Specification Methodology," pp.235-238, Formal Aspects of Computing, Springer-Verlag, 1992.

[9] Muller, M. Orgun, at al, "A Reverse Engineering Approach to Subsystem Structure Identification," Journal of software Maintenance, 5(4): 181-204, 1993.

[10] Neighbors, "The Draco Approach to Constructing Software from Reusable Components," IEEE Transactions on Software Engineering, Vol.11, No.5, pp.564-574, 1984.

[11] Park Wei-Jin, et al, "Object-Oriented Model Refinement Technique in Software Reengineering," Technical Paper, Department of Computer Science Korea Advanced Institute of Science and Technology, System Engineering Research Institute, 1998.

[12] Prieto-Diaz : "Domain Analysis : An Introduction," ACM SIGSOFT Software Engineering Notes, Vol.15, No.2, pp.47-54, 1990.

[13] Software Technology for Adaptable Reliable Systems (STARS), The Reuse-Oriented Software Evolution (ROSE) Process Model, Version 0.5. Unisys STARS Technical Report STARS-UC-05155/001/00, Advanced Research Projects Agency, STARS Technology Center, 801 N. Randolph St. Suite 400, Arlington VA 22203, July 1993.

[14] 류성열, et al, "소프트웨어 유지보수를 위한 PSDG기반 의미분할모형의 설계", 한국정보처리학회 논문지 제5권 제8호, 1998.

부 록

재사용 참조구조 구축 알고리즘에 관한 JAVA원시 코드

```
import java.io.*;
import Parse;
public class reuse_model extends Parse
{
        public void build_table
        {
        boolean binary[][] = new boolean[4][7];
        String [] lattice_label = new String [10];
        boolean digit;
                for (i=0;j<4;j++)
                {
                for (j=0;k<7;k++)
                {
                        int binary [j][k] = digit;
                        if (binary [j][k] == true)
                        {
                                lattice_label [1] = str1 [i];
                        }
                }
                }
        }
    public void abstract_lattice
    // Lattice 의 상위계층을 구성하는 단계이며, 최하위의 단말계층의
       AND function을 수행
        {
        x=0;
        y=2;
        int z;
        char string1;
        char string2;
        boolean [] abstract_binary = new boolean[10];
        String [] lattice_relation = new String[10];
        do {
                do {
                        for (l=0;l<k;l++)
                              {
                              z=x;
                              abstract_binary [x] = (binary[z][l]
                               && binary[x+1][l]);
                                if (abstract_binary [x] == true)
                                        {
                                        lattice_label [y] = str1 [x];
                                        string1 = char z;
                                        string2 = char x;
                                        lattice_relation  [y] = string1
                                          +string2+1;
                                        }
                              }
                        } while (x<10);
                } while (y<10);
        }
        public static void main (String args[])
        {
        for (y=0;y<10;y++)
                {
                System.out.println(lattice_label [y]);
                System.out.println(lattice_relation [y]);
                }
        }
}
```

이 기 오

e-mail : kolee@hcc.ac.kr

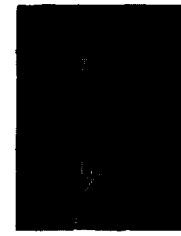
1989년 2월 순천향대학교 전자계산학과(공학사)

1994년 2월 숭실대학교 대학원 전자계산학과(공학석사)

1997년 8월 숭실대학교 대학원 전자계산학과 박사수료

1996년 3월~현재 혜천대학 의료정보시스템과 조교수

관심분야 : 분산객체컴퓨팅, 리엔지니어링



류 성 열

e-mail : syrhew@computer.soongsil.ac.kr

1997년 2월 아주대학교 컴퓨터공학부(공학박사)

1997년 3월~1998년 2월 George Mason University 교환교수

1981년 3월~현재 숭실대학교 컴퓨터 학부 교수

1998년 3월~현재 숭실대학교 정보과학대학원 원장

관심분야 : 리엔지니어링, 분산객체컴퓨팅, 소프트웨어 재사용