DOI: 10.3745/KIPSTD.2009.16D.6.927

UML CASE 도구 프레임워크를 위한 공통성 및 가변성 분석 기법

최 환 복[†] · 이 은 서[††] · 김 윤 호[†††]

요 약

본 논문에서는 UML CASE 도구 프레임워크 구축에 이용할 수 있는 공통성 및 가변성 분석 기법을 제안하고자 한다. 공통성 및 가변성 분석은 동일 도메인에서 어플리케이션 특성에 따라 변할 수 있는 영역과 변하지 않는 영역을 구분함으로써 확장과 재사용성을 향상시킬 수 있는 방법이다. 공통성 및 가변성 분석 기법으로 클래스에 기반한 분류기법을 제안하였으며, 이를 명시적으로 나타낼 수 있는 표기법을 제시하였다. 또한 분석 기법을 바탕으로 프레임워크를 구현하였으며, 결함 제거 효율성을 이용해 분석 기법의 검증을 수행하였다.

키워드 : UML, CASE 도구, 프레임워크, 공통성, 가변성

Commonality and Variability Analysis Method for UML CASE Tool Frameworks

Choi Hwan-Bok[†] · Lee Eun-Ser[††] · Kim Yun-ho[†††]

ABSTRACT

This paper presents a commonality and variability analysis method for UML CASE tool frameworks. Commonality and Variability analysis increase extension and reusability by separating common area and variable area. We suggest class category based on property and the notation to represent commonality and variability. It is also implements frameworks based on analysis method and verify method using defect removal efficiency.

Keywords : UML, CASE Tool, Framework, Commonality, Variability

1. 서 론

UML(Unified Modeling Language)은 일반적이고 통합적인 모델링 표기법으로써 시스템의 기능적인 요구사항을 이끌어내고, 시스템의 정적인 구조 및 동적인 행위 표현과 같은 분야에서 널리 사용되고 있다. CASE(Computer-aided software engineering) 도구는 소프트웨어의 개발을 구조화하고 제어하는데 있어 컴퓨터의 지원을 제공하는 소프트웨어이다. UML을 이용한 소프트웨어 모델링에서 컴퓨터의 지원을 위해 [1-3]와 같이 다양한 측면에서 UML CASE 도구에 관한 연구가 제시되었다. 하지만 이들 연구는 UML의 특정부분에 초점이 맞추어져 있으며, 변경, 확장, 재사용이 고려되지 않았다.

공통성 및 가변성 분석 기법은 동일 도메인에 대해 변하지 않는 특성과 변하는 특성을 분리시키는 방법이다. 공통성 및 가변성을 분리시킴으로써 공통적인 특성은 다른 어플리케이션에 대해 재사용이 가능하며, 가변적인 특성은 어플리케이션의 특성에 따라 확장이 가능해진다.

본 연구에서는 기존 UML CASE 도구 연구에서 제시하지 못한 변경, 확장, 재사용이 가능한 유연성을 가지는 프레임워크를 위해 공통성 및 가변성 분석 기법을 제안하고자 한다. 프레임워크를 구성하는 클래스에 대해 공통성 및 가변성 분석 방법을 제시하고 이를 명시적으로 표기할 수 있는 표기법을 제안한다. 마지막으로 제시한 분석 기법을 실질적으로 적용하여 효용성을 평가한다.

† 준 회 원 : 안동대학교 컴퓨터공학과 석사과정
†† 종신회원 : 안동대학교 컴퓨터공학과 조교수
††† 정 회 원 : 안동대학교 컴퓨터공학과 정교수

논문접수 : 2009년 10월 14일
수 정 일 : 1차 2009년 11월 4일
심사완료 : 2009년 11월 6일

2. 기반 연구

본 절은 연구 수행을 위해 필요한 기반연구 내용을 기술한다. 2.1절에서는 기존에 제시된 공통성과 가변성 분석 기법을 살펴보고, 2.2절에서는 프레임워크의 정의와 프레임워크 사용상의 이점을 기술한다. 마지막으로 2.3절에서는 분석 기법 검증에 활용할 수 있는 결함제거 효율성에 대해 살펴보도록 한다.

2.1 공통성과 가변성 분석 기법

가변성(variability)은 동일한 도메인에 속한 여러 어플리케이션에서 특성에 따라 변할 수 있는 성질을 의미한다[4]. 많은 연구들이 소프트웨어 개발 비용을 줄이기 위해 재사용 방법론에 관심을 두었고, 컴포넌트 기반 개발(Component-based development, CBC)과 프러덕트 라인 공학(Product line engineering, PLE)에서 재사용성을 향상시키기 위해 가변성 개념을 고안하였다.

CBD는 재사용 단위인 컴포넌트를 조립하여 효율적으로 어플리케이션을 개발하는 방법론이며, PLE는 재사용 단위인 미리 정의된 핵심 자산을 목적에 맞게 인스턴스화하여 목표 어플리케이션을 개발하는 방법론이다. 기존에 제시된 공통성 및 가변성 분석 기법으로는 컴포넌트 기반의 [5-6]와 PLE 기반의 [7-9]등이 있다.

2.2 프레임워크

프레임워크는 재사용이 가능하고 소프트웨어의 가격을 낮추고 품질을 향상시키기 위한 소프트웨어 설계 및 구현 기술이다[10]. 프레임워크는 전통적인 객체지향 접근법의 초석이 되었으며, 컴포넌트로써 소프트웨어 프러덕트 라인에서 재사용되었다.

프레임워크의 대표적인 정의는 다음과 같다.

- 소프트웨어의 특정 클래스에 대해 재사용 가능한 설계를 가능하게 하는 협업하는 클래스들의 집합이다[11].
- 프레임워크는 시스템의 일부 또는 전체의 재사용 가능한 설계로, 추상 클래스와 클래스의 인스턴스의 상호작용으로 표현된다[12].

프레임워크를 사용함으로써 얻을 수 있는 이점은 다음과 같다[13].

- 작성해야 하는 코드가 줄어든다.
- 코드의 신뢰성과 유연성이 증가된다.
- 일관성 있고 모듈화 된 코드가 된다.
- 연관된 문제에 대해 재사용할 수 있는 일반적인 해결책이다.
- 유지보수성이 증대되며, 일관성 있는 프로그램 발전이 가능하다.
- 관련된 프로그램의 통합이 용이하다.

2.3 결함 제거 효율성

결함 제거 효율성(Defect Removal Efficiency, DRE)은 해당 개발 단계에서 발견된 결함수가 완전히 결함 관리를 통해 발견되고, 이를 얼마나 제거하는가의 효율성을 분석하는 방법이다. 따라서 결함 제거 효율성을 분석하기 위해서는 해당 개발 단계에서 발견된 결함수와 다음 단계에서 이전 단계의 결함이 발견된 개수를 산정해야 한다. 결함 제거 효율성의 산출 방법은 다음과 같다[14].

DRE = E / (E + D)
E = 현 단계에서 발견된 결함 수
D = 다음 단계에서 이전 단계의 결함이 발견된 수

결함제거 효율성은 1에 가까워야 이상적인 값이 된다. 즉, 1에 가깝게 되면 결함을 해당 단계에서 완전히 발견하여 제거를 하게 되는 것이다.

3. 공통성 및 가변성 분석 기법

3.1 공통성 및 가변성 분류를 위한 특성

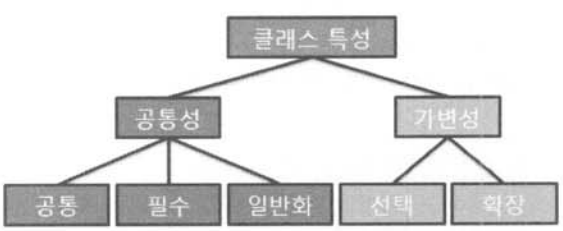
공통성 및 가변성 분류는 동일 도메인 내에서 추후 변경을 미리 예상하여 가변하는 영역과 불변하는 영역을 구분함으로써 재사용성을 향상시킨다. 프레임워크를 유연하게 만들기 위해서는 공통성과 가변성을 분리시킬 필요가 있다.

2.2절에서 언급한 바와 같이 클래스는 프레임워크를 구성하는 가장 일반적인 요소로써 프레임워크의 재사용을 위해서는 클래스를 기반해야 한다. 이를 위해 클래스에 특성을 부여하고 이 특성에 따라 클래스의 공통성과 가변성을 분류하도록 한다. 공통성과 가변성을 적용하기 위해 필요한 클래스는 유스케이스 분석을 통해 추출한다. 공통성 및 가변성 분류를 위해 정의한 클래스 특성은 (그림 1)과 같다.

클래스 특성은 성질에 따라 공통성과 가변성으로 나누어진다. 공통성에 해당하는 클래스 특성은 공통(common), 필수(mandatory), 일반화(generalizable)이며, 가변성에 해당하는 클래스는 특성은 선택(optional), 확장(extendable)이다.

공통 클래스 특성은 도메인 내에서 어플리케이션 특성에 관계없이 공통적으로 적용이 가능한 클래스를 나타낸다.

필수 클래스 특성은 어플리케이션 특성에 관계없이 반드시 존재해야 하는 클래스로써 프레임워크 구동에 꼭 필요한 클래스를 나타낸다.



(그림 1) 공통성 및 가변성 클래스 특성

〈표 1〉 클래스 공통성 및 가변성 분석을 위한 클래스 특성 분석표의 예

특성 / 클래스	공통성			가변성	
	공통	필수	일반화	선택	확장
클래스1					
클래스 2					
⋮					
클래스 n					

일반화 클래스 특성은 식별된 클래스들 사이에 공통점이 존재하여 일반화를 통해 기반 클래스(base class)의 추출이 가능함을 나타낸다. 일반화 클래스 특성은 구상 클래스(specific class)에서 기반 클래스를 추출하는 bottom-up approach에 해당한다. 즉, 유스케이스 분석을 통해 클래스를 바로 추출할 수도 있지만 다수개의 클래스에서 공통점을 찾아 클래스를 추출할 수 있기 때문에 일반화 클래스 특성이 필요하다.

선택 클래스 특성은 어플리케이션 특성에 의거해 정의할 수도 있고, 하지 않을 수도 있는 클래스를 의미한다.

확장 클래스 특성은 현재 정의된 클래스가 어플리케이션 특성에 따라 구체적인 클래스로 세분화 될 수 있는 클래스를 의미한다. 확장 클래스 특성은 기반 클래스를 바탕으로 구상 클래스로 확장하는 top-down approach에 해당한다. 즉, 추후 확장 가능성을 명시적으로 나타내기 위한 특성이다.

공통성은 불변영역이며 가변성은 가변영역으로 서로 반대의 성질을 가진다고 할 수 있다. 본 논문에서 정의한 클래스 특성 중 확장은 기반 클래스를 구상 클래스로 세분화하는 것으로 가변성을 의미하지만 기반 클래스를 수정하지 않고 이를 확장 하는 것이기 때문에 공통성에 정의한 클래스 특성과 조합이 가능하다.

공통성의 일반화 클래스 특성은 여러 구상 클래스에서 공통점을 찾아 일반화된 클래스를 추출하는 것을 의미한다. 따라서 일반화 클래스 특성을 가지는 구상 클래스에서 기반 클래스를 추출하면 구상 클래스는 일반화 특성이 제거되며, 추출한 기본 클래스는 확장 특성을 가지게 된다.

<표 1>은 앞서 정의한 특성을 이용하여 클래스의 공통성과 가변성을 분석 시 사용할 수 있는 클래스 특성 분석표이다. 좌측에 클래스 이름을 작성하고 우측의 클래스 특성 중 해당하는 특성을 표기하면 된다.

3.2 공통성 및 가변성 분류 표기법

보다 명확하게 공통성과 가변성을 분류하기 위해서는 분류기법뿐만 아니라 이를 표현할 수 있는 표기법이 필요하다. UML을 이용해 가변성을 표기한 [15-16]과 같은 연구가 제시되었다. 하지만 이들 연구는 특정 시점의 변경점(variant point)를 기반으로 가변성을 표현하기 때문에 클래스의 구조 표현에 적합하지 않다. 따라서 본 연구에서는 이를 해결하는 방안으로써 3.1절에서 정의한 클래스 특성을 표기할 수 있는 표기법을 정의한다.

〈표 2〉 클래스 특성 표기법

클래스 특성	표기법
필수	«mandatory»
공통	«common»
일반화	«generalizable»
선택	«optional»
확장	«extendable»

(그림 2) 공통성 및 가변성을 적용한 클래스 다이어그램

공통성 및 가변성 분류를 위한 표기법은 <표 2>와 같으며 이를 클래스 다이어그램에 적용한 것은 (그림 2)와 같다. 단 새로운 기호를 정의하지 않고 UML의 표기법 확장 기술인 스테레오타입을 이용하도록 한다.

4. 공통성 및 가변성 분석 기법의 적용

본 장에서는 3장에서 제안한 클래스의 공통성 및 가변성 분석기법을 유스케이스 다이어그램 CASE 도구 구현에 적용한 내용을 기술한다. 클래스에 공통성 및 가변성 특성 적용 순서는 먼저 유스케이스 기술서의 시나리오에서 명사 추출법을 이용해 클래스를 추출하고, 추출한 클래스를 바탕으로 공통성 및 가변성 분석을 수행하여 클래스 특성을 적용한다. 마지막으로 클래스 다이어그램을 작성하여 공통 클래스와 가변 클래스를 명시적으로 나타내고 클래스 간의 연관관계를 표현한다.

4.1 클래스의 공통성 및 가변성 분석

프레임워크의 공통성과 가변성을 분석하기 위해서는 프레임워크를 구성하는 클래스가 필요하다. 클래스는 유스케이스 기술서의 시나리오에서 명사 추출법을 사용하여 추출한다. 여기서는 UML의 유스케이스 다이어그램이 대상이며, 클래스 추출의 대상이 되는 기술서는 유스케이스 다이어그램 작성으로 제한하여 공통성 및 가변성을 분석한다. 클래스를 추출의 대상이 되는 유스케이스 기술서는 (그림 3)과 같다.

(그림 3)의 유스케이스 기술서의 시나리오의 본문을 대상으로 명사 추출법을 이용하여 후보 클래스를 나열하고, 의미가 중복되는 후보 클래스가 존재한다면 하나의 클래스만 남기고

유스케이스 명: 유스케이스 다이어그램 작성
액터: CASE 도구 사용자
시나리오 1. CASE 도구 사용자는 메뉴 영역에서 작성하고자 하는 다이어그램 요소(액터, 유스케이스, association, include, extend, generalization)을 선택한다. 2. 메뉴 영역은 사용자가 선택한 요소를 저장한다. 3. CASE 도구 사용자는 다이어그램 작성 영역에 다이어그램 요소 작성을 요청한다. 4. 다이어그램 작성 영역은 메뉴 영역에게 사용자가 어떤 요소를 선택하였는지 질의한다. 5. 메뉴 영역은 사용자가 선택한 요소를 다이어그램 작성 영역에게 반환한다. 6. 다이어그램 작성 영역은 메뉴 영역이 전달한 요소를 표현한다.

(그림 3) 클래스 추출의 대상이 되는 유스케이스 기술서

나머지 의미가 중복되는 클래스는 삭제하였다. 이와 같은 방법을 이용하여 추출한 클래스 및 역할은 <표 3>과 같다.

다음으로 프레임워크를 구성하는 클래스의 공통성과 가변성을 추출하기 위해 3장에서 정의한 클래스 특성을 적용한다. 유스케이스 시나리오에서 추출한 클래스를 바탕으로 공통성과 가변성을 적용한 것은 <표 4>와 같으며 이를 클래스 다이어그램으로 표현한 것은 (그림 4)와 같다.

CommandArea 클래스는 다이어그램 작성에 필요한 명령어를 포함하고 있는 클래스로 어플리케이션 특성에 관계없이 반드시 필요한 클래스이다. 또한 어플리케이션 특성에 따라 다른 명령들을 포함 할 수 있기 때문에 확장 특성을 가진다.

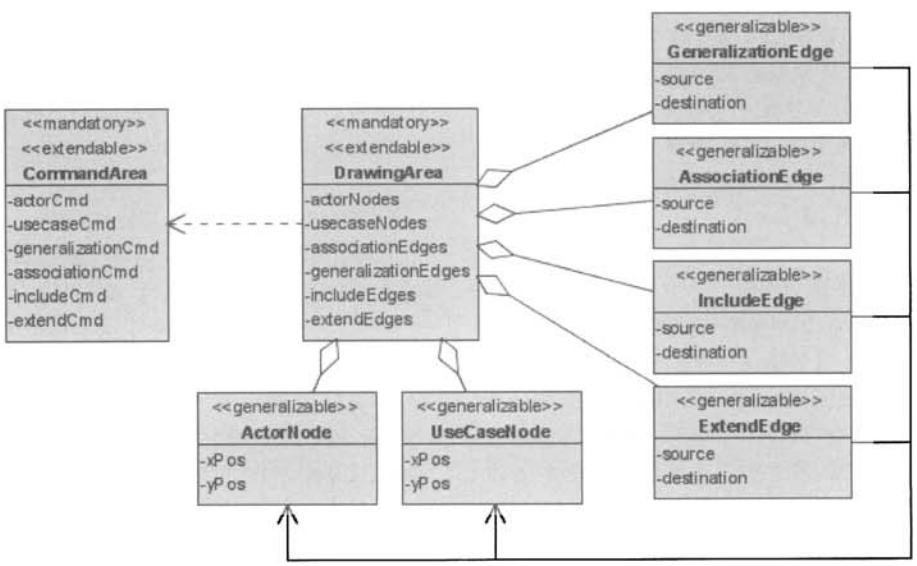
DrawingArea 클래스는 다이어그램이 실제 표현되는 클래스로

〈표 3〉 유스케이스 시나리오를 바탕으로 추출한 클래스

클래스	역할
CommandArea	유스케이스 다이어그램 작성이 필요한 명령어(액터, 유스케이스, 상속관계, 연관관계 그리기 등)를 가지고 있는 영역
DrawingArea	실제 다이어그램이 작성되는 영역
ActorNode	유스케이스 다이어그램의 액터를 표현
UseCaseNode	유스케이스 다이어그램의 유스케이스를 표현
AssociationEdge	유스케이스 다이어그램에서 액터와 유스케이스 간의 연관관계를 표현
InheritanceEdge	유스케이스 다이어그램에서 액터간의 상속 또는 유스케이스 간의 상속을 표현
ExtendEdge	유스케이스 다이어그램에서 유스케이스 간의 확장관계를 표현
IncludeEdge	유스케이스 다이어그램에서 유스케이스 간의 포함관계를 표현

〈표 4〉 공통성과 가변성 특성 적용한 클래스

특성 / 클래스	공통성			가변성	
	공통	필수	일반화	선택	확장
CommandArea		O			O
DrawingArea		O			O
ActorNode			O		
UseCaseNode			O		
AssociationEdege			O		
InheritanceEdge			O		
ExtendEdge			O		
IncludeEdge			O		



(그림 4) 공통성 및 가변성을 적용한 초기 클래스 다이어그램

다이어그램 작성에 반드시 필요한 클래스이다. 즉 DrawingArea 클래스가 없다면 다이어그램을 작성할 수 없기 때문에 필수 특성을 가진다. 또한 추후 추가 다이어그램 요소을 지원할 가능성이 있기 때문에 확장 특성을 가지도록 하였다.

ActorNode, UseCaseNode, AssociationEdge, InheritanceEdge, ExtendEge, IncludeEdge는 UML 정의[17]에 의거해 점(node)과 선(edge)으로 일반화가 가능하기 때문에 일반화 특성을 가지도록 한다. <표 4>를 바탕으로 작성한 클래스 다이어그램은 (그림 3)과 같다.

마지막 단계는 <표 4>에서 일반화 특성을 가진 클래스에서 공통성을 가지는 기반 클래스를 찾는 과정이다. 클래스 간의 공통성을 찾아 기반 클래스로 구성함으로써 클래스의 확장과 재사용성을 증가시킬 수 있다. <표 4>를 기반으로 갱신한 클래스는 <표 5>와 같으며 이를 클래스 다이어그램으로 나타낸 것은 (그림 5)와 같다.

ActorNode와 UseCaseNode 클래스가 UML 다이어그램에서 점(node)이라는 것에 착안, 공통 특성을 가지는 Node 클래스를 추가하였다. Node 클래스는 어플리케이션 특성에 따라 구상 클래스로 세분화될 수 있기 때문에 확장 특성을 가진다. ActorNode와 UseCase 클래스는 어플리케이션 특성에 따라 필요성이 다르기 때문에 선택 특성을 부여하였다. 또한 AssociationEdge, InheritanceEdge, ExtendEdge, IncludeEdge 클래스가 UML 다이어그램에서 연결선(edge)을 의미함으로 이를 일반화시켜 공통 특성을 가지는Edge 클래스를 추가하고 AssociationEdge, InheritanceEdge, ExtendEdge, IncludeEdge 클래스는 어플리케이션 특성에 따라 정의하도록 하였다. 공통성을 가지는 Node 클래스와 Edge 클래스가 추가되어 DrawingArea 클래스는 기반 클래스인 node와 edge 클래스를 가지게 되었다.

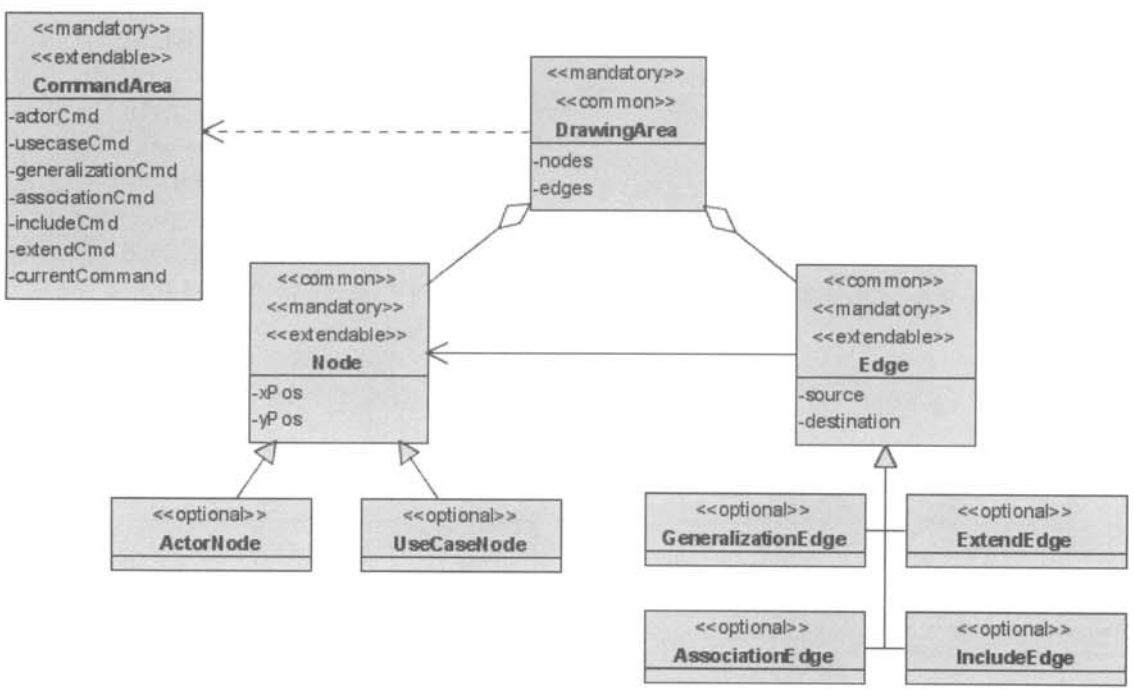
〈표 5〉 갱신된 특성 적용 클래스

특성 / 클래스	공통성			가변성	
	공통	필수	일반화	선택	확장
CommandArea		O			O
DrawingArea	O	O			O
Node	O	O			O
ActorNode				O	
UseCaseNode				O	
Edge	O	O			O
AssociationEdege				O	
InheritanceEdge				O	
ExtendEdge				O	
IncludeEdge				O	

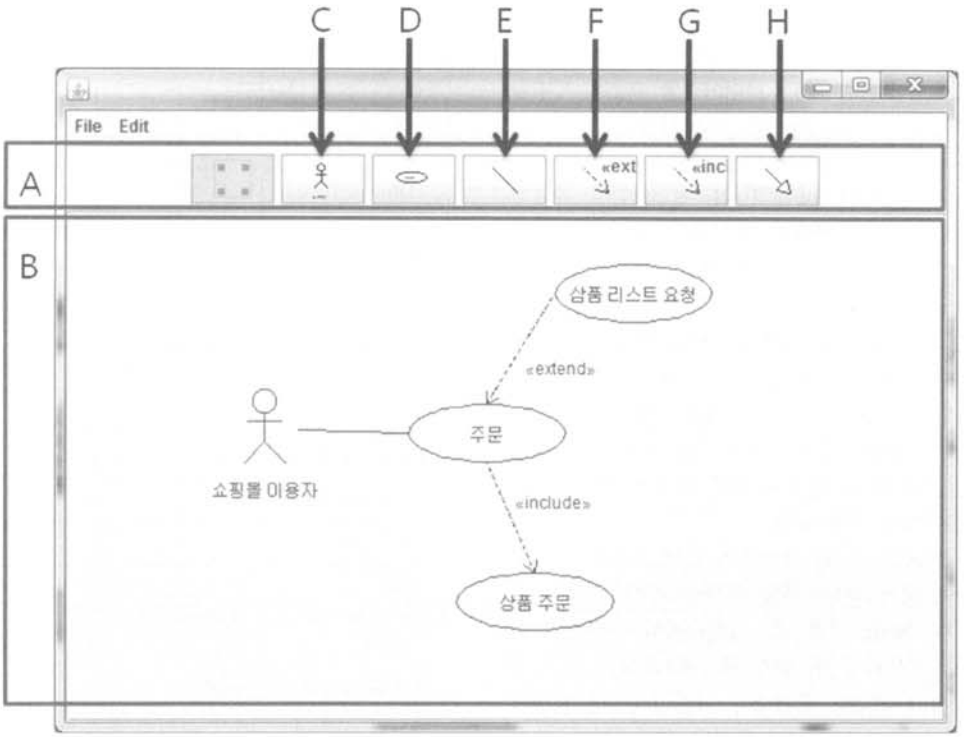
4.2 프레임워크의 구현

(그림 6)은 4.1절에서 수행한 공통성 및 가변성 분석을 바탕으로 자바언어를 이용하여 프레임워크의 프로토타입을 구현한 것이다.

A영역은 다이어그램 작성시 필요한 명령어를 포함한 영역으로 이를 이용해 다이어그램을 작성할 수 있다. B영역은 A 영역의 명령들을 이용해 실제 다이어그램을 작성할 수 있다. A영역의 C에서 H는 다이어그램 작성에 사용하는 액터, 유스케이스, 관계, 포함, 확장, 일반화 명령이다.



(그림 5) 기반 클래스 추출 후 갱신한 클래스 다이어그램



(그림 6) UML CASE 도구 프레임워크 구현

5. 클래스 기반의 공통성 및 가변성 분석기법 검증

본 장에서는 3장에서 제시한 공통성 및 가변성 분석기법을 화상회의 시스템 프로젝트에 적용하여 검증한다. 본 논문에서 제시한 공통성 및 가변성 분석기법은 결점(defect)를 줄이는 것이 목적이다. 따라서 결점 제거 효율성을 이용하는 것이 적절하다. 2.3절의 결함 제거 효율성 계산법에 의거해 분석 기법 적용 이전의 결함 제거 효율성과 분석 기법 적용 이후의 결합 제거 효율성을 비교하였다. 결함 제거 효율성을 계산하기 위해서는 결함 개수가 필요하다.

분석 기법을 적용하기 이전의 결함 개수는 <표 6>과 같다.

분석기법 적용 이전의 결함 개수를 나타내는 <표 6>에 의거해 각 개발 단계별 결합 제거 효율성을 계산하면 다음과 같다.

0.571 = 4 / (4 + 3) (요구사항 단계)
0.583 = 14 / (14 + 10) (설계 단계)
0.615 = 32 / (32 + 20) (코딩 단계)

분석기법을 적용 후 결함 제거 효율성 계산을 위해 산출한 결함 개수는 <표 7>과 같다.

<표 7>에 의거해 분석기법 적용 후 결함 제거 효율성을 계산하면 다음과 같다. 결함제거 효율성은 2.3절의 결함 제거 효율성 계산법을 의거해 계산한다.

0.571 = 4 / (4 + 3) (요구사항 단계)
0.647 = 11 / (11 + 6) (설계 단계)
0.666 = 16 / (16 + 9) (코딩단계)

<표 6>과 <표 7>에 의거해 분석기법 적용 이전의 결함 제거 효율성과 분석기법 적용 이후의 결함제거 효율성은 <표 8>과 같다.

분류기법 적용 이전과 분류기법 적용 이후의 결함제거 효율성을 비교하면, 요구사항을 추출하는 단계는 분석 기법의

〈표 6〉 분석기법 적용 이전의 결함 개수

결함수 / 개발단계	현 단계에서 발견된 결함 수(E)	다음 단계에서 이전 단계의 결함이 발견된 결함 수(D)
요구사항	4	3
설계	14	10
코딩	32	20

〈표 7〉 분석기법 적용 후의 결함 개수

결함수 / 개발단계	현 단계에서 발견된 결함 수(E)	다음 단계에서 이전 단계의 결함이 발견된 결함 수(D)
요구사항	4	3
설계	11	6
코딩	30	15

〈표 8〉 분석기법 적용 이전과 이후의 결함 제거 효율성 비교

결함제거 효율성 / 개발단계	분석기법 적용 이전의 결함 제거 효율성	분석기법 적용 이후의 결함제거 효율성
요구사항	0.571	0.571
설계	0.583	0.647
코딩	0.615	0.666

적용 대상에서 제외됨으로 결과가 동일하다. 하지만 본 논문에서 제시한 분석기법이 적용되는 설계단계 이후에서는 결함 제거 효율성이 차이가 나는 것을 볼 수 있다. 즉, 본 논문에서 제시한 분류기법을 적용함으로써 설계단계에서 결함을 줄이고 이를 바탕으로 이루어지는 코딩 단계까지 영향을 미치는 것을 볼 수 있다.

6. 결 론

본 논문에서는 UML CASE 도구 프레임워크를 위한 공통성과 가변성 분석 방법을 제시하였다. 프레임워크의 재사용성과 확장을 위해 클래스 기반의 공통성과 가변성 분류 방법을 제시하였으며, 클래스 분류를 명시적으로 나타내기 위해 클래스 특성 표기법을 제안하였다. 또한 분석 방법을 기반으로 유스케이스 다이어그램 CASE 도구를 구현하였으며, 제안한 분석 방법을 결함 제거 효율성을 이용하여 검증하였다.

향후 과제로는 본 논문에서 제시한 공통성 및 가변성 분류를 정량화하여 클래스 특성을 적용하는 연구가 요구된다. 이와 같은 연구는 클래스 특성의 적용에 있어 경험에 의거한 직관적인 방법이 아니라 객관적으로 적용함으로써 프레임워크 설계의 정확성과 생산성을 얻을 수 있을 것이다.

참 고 문 헌

[1] Cao, S., Grundy, J., Hosking, J., Stoeckle, H., Tempero, E. and Zhu, N., "Generating Web-based User Interfaces for Diagramming Tools," Proceedings of the Sixth Australasian conference on User interface, Vol.40, pp.63-72, 2005.

[2] Khaled, R., Mackay D., Biddle, R., Nobble, J., "A Lightweight Web-Based Caase Tool for Sequence Diagrams," Proceedings SIGCHI-NZ Symposium on Computer-Human Interaction, pp.55-60, 2002.

[3] Mackay, D., Noble, J., Biddle, R., "A Lightweight Web-Based case tool for class diagrams," Proceedings of the Fourth Australasian user interface conference on User interfaces, Vol.18, pp.95-98, 2002.

[4] Bosch, J. Design and Use of Software Architectures, Addison-Wesley, 2000.

[5] 김철진, "컴포넌트 아키텍쳐 기반의 동적 컴포넌트 조합을 위한 가변성 설계 기법", 인터넷정보학회논문지 제6권 제2호, 2005.

[6] 박지영, "컴포넌트 가변성을 위한 Required 인터페이스 설계", 한국정보과학회 학술발표논문집 제30권 제2호, 2003.

[7] 이순복, "소프트웨어 제품 계열 공학의 온톨로지 기반 휘처 공통성 및 가변성 분석 기법", 정보과학회논문지 제34권 제3호, 2007.

[8] 장수호, "프로덕트라인 아키텍쳐의 실용적 설계기법", 정보과학회논문지 제32권 제3호, 2005.

[9] 문미경, "소프트웨어 프로덕트 라인에서 가변성 분석을 통한 도메인 아키텍처 개발 방법", 정보과학회논문지 제34권 제4호, 2007.

[10] Mohamed Fayad, Douglas C. Schmid, "Object-Oriented Application Frameworks", Communication of the ACM Volume 40, pp.32-38, October, 1997.

[11] Gamma E., Helm R., Johnson R., Vlissides J., Design Patterns - Elements of Reusable Object-Oriented Software., Addison Wesley, 1994.

[12] Ralph E. Johnson, "Frameworks = Component + Patterns", Communications of the ACM archive Volume 40, pp39-42, October, 1997.

[13] S. Cotter, M. Potel, Inside Taligent Technology, Addison-Sesley, 1995.

[14] Roger S. Pressman, Software Engineering A Practitioner's Approach 4th ed, McGraw-Hill, 1997.

[15] S. Robak, B. Franczyk, K. Politowicz, "Extending the UML for modeling variability for system families", International Journal of Applied Mathematics and Computer Science Volume 12, pp.285-298, 2002.

[16] Matthias Clauss, "Generic modeling using uml extensions for variability", OOPSLA 2001Workshop on Domain Specific Visual Languages, 2001.

[17] Grady Booch, James Rumbaugh, Ivar Jacobson, The Unified Modeling Language User Guide 2nd ED, Addison-Wesley, 2005.

최 환 복

e-mail : choibeta@gmail.com

2007년 안동대학교 컴퓨터공학전공(학사)

2009년~현 재 안동대학교 컴퓨터공학과 석사과정

관심분야 : 객체지향 시스템, 인터넷 컴퓨팅, 리팩토링 등

이 은 서

e-mail : eslee@andong.ac.kr

2001년~현 재 ISO/IEC 15504 국제심사원

2004년 중앙대학교 컴퓨터공학과(박사)

2004년~현 재 임베디드 산업협회 전문위원

2004년~현 재 한국정보통신기술협회 위원

2005년~2007년 숭실대학교 정보미디어기술연구소 연구 교수

2008년~현 재 안동대학교 컴퓨터공학과 조교수

관심분야 : CBD, Formal method, Quality model, SPI(Software Process Improvement), Defect Management 등

김 윤 호

e-mail : unokim@andong.ac.kr

1997년 경북대학교(박사)

1997년~현 재 안동대학교 컴퓨터공학과 정교수

관심분야 : 객체지향 시스템, 인터넷 컴퓨팅, 병렬처리 등