UML 분석을 위한 함수 기반 설계내역 항해기의 구현

(Implementation of a Function-Based Design Document Navigation Tool for UML Analysis)

김 원 중 † 배 명 남 †† 양 재 동 ††
(Won-Jung Kim) (Myung-Nam Bae) (Jae-Dong Yang)

요 약 개발자는 다양한 설계 도구들을 통해 많은 설계내역들을 생성한다. 따라서, 생성된 설계내역들의 분석을 용이하게 해주는 도구들이 필요하다. 이 도구들은 설계내역들 사이에 관계성을 정의하고 자유로운 항해 메소드를 제공함으로써, 전체 시스템을 이해하고 검증하는데 사용될 수 있다.

본 논문에서는 설계내역들간의 복제, 인스턴스, 그리고 전이 관련성 파악을 통해 시스템을 체계적으로 분석할 수 있도록 지원하는 항해기를 제안하고 구현하였다. 항해기는 기존의 UML 설계 도구들이 갖는 설계요소로의 항해 방식과는 달리, 클래스 다이어그램내 정적 구조 정보뿐만 아니라 이와 연계하여 함께 파악되어야 하는 시퀀스 다이어그램, 스테이트 다이어그램내 동적인 면을 기술하는 설계내역을 구성하는 설계요소들도 함께 항해할 수 있도록 확장되었다. 즉, 설계내역 항해기는 UML이 갖는 모든 설계 시멘틱에 따라 설계요소들을 유기적으로 파악할 수 있도록 함으로써 시스템의 행위를 체계적으로 파악하고 검증하는데 사용될 수 있다. 이를 위해, 본 논문에서는 1) 복제, 인스턴스, 전이 등의 관련성을 정의하고, 2) 정의된 방식에 따라 관련된 설계요소들에 관련성을 부여한 뒤, 3) 이 관련성에 의해 관련된 설계요소들을 항해하기 위한 일련의 함수들을 제시한다.

키워드 : 관련성, 항해 정보, 항해 함수

Abstract System developers create a lot of design documents by various case tools. It is necessary to have the tools for facilitating the analysis of the documents. These tools can be used to understand and verify the whole process of a system, by defining relationships among the documents and providing free navigation methods.

In this paper, we develop a navigation tool that enables the developers to systematically analyze the system by capturing duplication, instance, and transition relationships between the documents. Different from the navigation facilities of the other UML design tools, this tool makes it possible to navigate design elements in design documents such as sequence diagrams, state diagrams and class diagrams. In other words, it can be used to systematically capture and verify both the static structure and the dynamic behavior of the system by keeping track of such elements. To provide such a facility, 1) we define three relationships: duplication, instance, and transition, 2) assign relation to the related design elements according to the predefined way, and then 3) present a set of functions for navigating related design elements

Key words : relationship, navigation information, navigation function

· 본 논문은 전북대학교 영상정보신기술연구소의 지원으로 수행되었음.
† 학생회원 : 전북대학교 컴퓨터통계정보학과
wjkim@cs.chonbuk.ac.kr
†† 비 회 원 : 한국전자통신연구원 네트워크 S/W 플랫폼팀 선임연구원
mnbac@etri.re.kr
††† 정 회 원 : 전북대학교 전자정보공학부 교수
jdyang@cs.chonbuk.ac.kr
논문접수 : 2001년 2월 26일
심사완료 : 2002년 6월 19일

1. 서 론

시스템 개발에 잘 정의된 설계 시멘틱을 제공하는 모델링 기법의 적용은 전체 개발 프로세스의 비용과 품질에 영향을 미치는 주요한 요소이며[8], 시스템의 복잡성이 증가함에 따라 설계 시멘틱의 효과적인 시각화 방식도 핵심적인 사항이 되어가고 있다. UML(Unified Modeling Language)은 표현 규칙이 잘 정의되어 있고, 구현에 대한 고려없이 소프트웨어 전 개발과정에 걸쳐 모든 프로세스를 수용하도록 고안되었다. 또한, UML에서는 모든 설계요소들을 체계화된 그래픽 표기로 모델링함으로써, 프로젝트 개발에 참여한 팀들간의 자료 교환 및 시스템의 구조를 손쉽게 파악할 수 있다[2,3,4]. 그 결과, 유지보수가 용이하고 재사용성이 높은 소프트웨어를 생산할 수 있다는 장점이 있다[5,6,11].

그러나, 현재의 CASE 도구들은 단순히 생성된 설계 정보와 그들간의 기초적인 설계 시멘틱만을 저장하고 검색하는 수준에 있다. 따라서, 이를 보완하여, 설계 정보의 각 단계별 내역들을 구체화된 설계 시멘틱으로 서로 연관시키고, 이들간의 자유로운 항해를 지원하는 항해 도구가 제공된다면, 시스템을 보다 체계적으로 파악할 수 있을 것이다. 본 논문에서는 이를 위한 항해 모델과 항해 도구에 대해 기술한다.

UML은 기본적으로 여러 종류의 다이어그램을 제공한다. 대표적 다이어그램인 클래스 다이어그램은 시스템을 구성하는 단위 특성들을 속성(attribute), 연산(operation), 클래스들간의 정적 관계(association, dependency, aggregation, generalization)로 명세함으로써 클래스를 모델링한다. 클래스 다이어그램은 시스템의 생명 주기와 수명을 같이하며, 주로 시스템의 기본 골격을 명세하는 데 사용된다. 시퀀스 다이어그램은 객체들에게 적용될 동적인 행위의 흐름들로 설계요소들 사이의 상호작용을 표현한다. 여기에서 말하는 행위의 흐름이란 객체의 조건이나 상황에 따라 지속적으로 변화하는 연관성을 구체화하는 과정이라고 할 수 있다. 예를 들어, 클래스 다이어그램에서 연관 관계는 클래스들 간의 정적인 참조 관계만을 명시한다. 반면에, 시퀀스 다이어그램에서는 연산의 바인딩 혹은 참조를 통해 이를 보다 구체화하여 명세한다. 이외에도 유스케이스(usecase), 오브젝트(object), 콜레보레이션(collaboration), 스테이트(state), 액티버티(activity), 컴포넌트(component), 디플로이먼트(deployment) 다이어그램 등이 있으며, 이들은 모두 자체 고유의 설계 특성을 기술하는 설계 시멘틱들[16]이다(이에 대한 자세한 설명은 [5,15]를 참조). 따라서, 설계내역을 체계적으로 분석하기 위해서는 전체 UML 다이어그램에 대해 설계 시멘틱들이 충분히 분석되고 분류되어야 하며, 이들을 융통성 있게 검색하기 위한 방법이 추가로 제공되어야 한다.

그림 1은 객체지향 방법에 따라 모델링된 전형적인 예이다. 클래스 다이어그램에서 클래스 c_1, c_2간에는 연관 r_1이 정의되었다. 이때, 대응되는 시퀀스 다이어그램의 메시지 m_i는 c_1, c_2간의 관계 r_1을 구체화한 사례임을 알 수 있다. 이 시멘틱을 바탕으로 "특정 메소드(예를 들면, c_1)의 실행에 직접 필요한 클래스들을 검색하라"라는 사용자의 질의에 정확한 결과를 제시할 수 있다. 하지만, 현재의 설계도구들[1, 10, 13, 14]은 이와 같은 설계요소들간의 인스턴스화(instanceOfclass), 전이, 관계의 사례화(instance of Relationship) 등의 설계 시멘틱 분석이 이루어지지 않아, 설계내역내 설계요소들간의 관련성이 충분히 파악되지 않는다는 단점이 있다. 물론, 클래스 다이어그램의 분석을 통해 정적 관계가 있는 클래스들을 제공하기도 하지만, 해당 메소드와 관련이 없는 클래스가 추출될 수도 있어 정확한 분석이라고 할 수 없다.

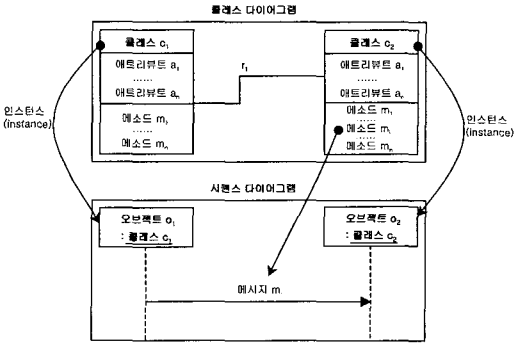

그림 1 설계요소들간의 설계 시멘틱

이러한 문제점은 오브젝트 o_2로의 메시지 m_i를 생성할 때, 단지 메시지 m_i의 표현에 필요한 기본적인 정보만을 생성할 뿐, m_i를 통해 c_1-m_i와 m_i-o_2, 그리고 o_2-c_2간의 관계를 파악하고 이 정보를 시스템 개발자들에게 제공하기 위한 체계를 갖고 있지 않기 때문이다.

이러한 문제점을 해결하기 위해, 클래스나 오브젝트 등과 같은 기본 UML 설계요소들간의 상호 의미 구조를 명확히 정의할 필요가 있다. 그리고, 이에 따라 관련 설계요소들간의 관련성을 유지한다면 설계내역들이 가지는 정보의 흐름, 상호 작용하는 관계, 그리고 실행 파정 추적 등의 특성을 일목요연하게 파악할 수 있을 것이다[7, 8, 9]. 즉, 설계요소 자체의 표현 정보뿐만 아니라 c_1-m_i와 m_i-o_2, 그리고 o_2-c_2와 같은 관계성들이 설

계도구내에 지속적으로 유지되고 관리되어야 한다.

본 논문에서는 실제 개발 환경에서 생성된 설계요소들간에 미리 정의한 관련성을 부여하고, 그 관련 정보를 시스템에 유지함으로써 해당 관련성을 추적할 수 있는 함수기반의 설계내역 항해기를 제안한다. 이 항해기는 설계요소들간의 유기적인 관련성을 설계자가 일목요연하게 파악할 수 있게 함으로써, 설계내역의 이해도와 신뢰도를 높일 수 있다. 이를 위해, 본 논문에서는 관련성 표현 함수를 정의하고, 정의된 함수에 따라 관련된 설계요소들로 항해하는 과정을 보인다. 항해가 이루어지는 설계요소들은 클래스 다이어그램뿐만 아니라 시퀀스 다이어그램 또는 스테이트 다이어그램과 같은 시스템의 동적인 면을 기술하는 다이어그램도 포함한다. 따라서, 설계내역 항해기는 이러한 다이어그램들간의 관련성을 유기적으로 파악함으로써 시스템의 행위를 체계적으로 검증할 수 있다.

여기에서, 함수 기반이라 함은 항해기가 제공하는 기본 규칙들이 함수 형태로 표현되고, 또한 이후 분석자가 자신만의 새로운 관계성을 정의할 때, 이들 기본 함수들을 조합하여 기본 함수와 동일한 형태의 새로운 함수를 재정의할 수 있도록 구축되었음을 의미한다. 이를 통해, 항해기가 제공하는 기본 함수와 분석자 자신이 정의한 항해 함수들이 모두 일관된 틀안에서 지속적인 확장이 가능하다는 장점이 있다.

본 논문의 주요 내용은 크게 두 가지로 나눌 수 있다. 첫째는 작성된 설계내역의 시멘틱에 따라 추적/항해할 수 있는 함수 기반 모델을 제시하고, 둘째, 다양한 설계내역을 사용자에게 시각적으로 제시하고, 제시된 내역에서 미리 정의한 시멘틱(관계성)에 따라 또 다른 내역을 역시 시각적 결과를 통해 참조할 수 있도록 한다.

본 논문의 구성은 다음과 같다. 먼저, 2장 관련연구에서는 UML을 지원하는 최근의 설계도구들에서 본 연구와 관련되는 기능들을 살펴보고, 3장에서는 제안한 항해기에서 관련성 정보 표현 방안과 질의어를 설명한다. 4장에서는 항해기의 인터페이스와 예를 보이며, 마지막으로 5장에서는 결론 및 향후 연구 과제를 제시한다.

2. 관련연구

이 절에서는 널리 알려진 UML 설계 도구들이 제공하는 참조 기능들을 소개하고, 이들이 제공하는 설계요소들간의 항해 방법에 대해 고려한다.

2.1 Rational Rose

Rational Rose[10]는 클래스의 상속과 같은 설계 시멘틱의 파악을 통해 클래스에 대한 참조 구조를 제공한다. 즉, 그림 2와 같은 'Navigation Tree'를 통해 클래스들간의 계층화된 구조를 제공할 뿐만 아니라 해당 클래스로의 직접 항해를 위한 편리한 인터페이스를 제공하고 있다. 이를 통해, 클래스 'CBaseRept'와 관련하여 설계내역 'Main'에서 상위클래스와 형제 클래스로의 항해가 가능하며, 다시 설계내역 'Base Part'로의 항해를 통해 하위클래스들을 쉽게 추출할 수 있다.

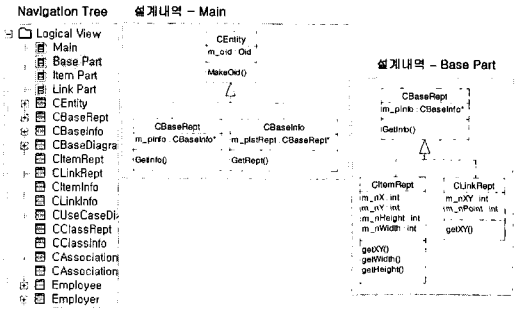

그림 2 클래스 다이어그램 설계내역

그러나, Rational Rose는 클래스와 설계내역간의 관계를 파악하지 않고 있어, "클래스 'CBaseRept'의 형제 클래스를 포함하고 있는 설계내역들을 추출하라"와 같은 질의에 대해 설계내역 'Main'을 추출하지 못한다. 다른 측면으로, 그림 3과 같이 클래스들간의 연관과 연관의 구체화된 특성을 파악하는 관점에 대해 고려해 보자.

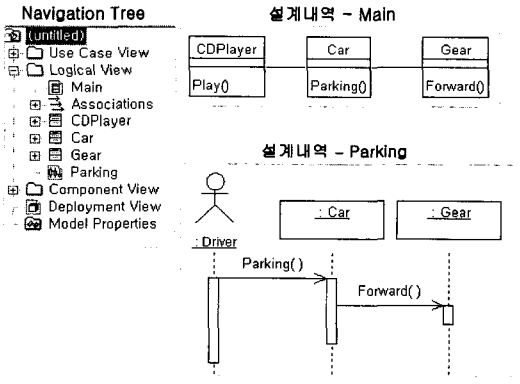

그림 3 시퀀스 다이어그램 설계내역

설계내역 'Main'에서 클래스 'Car'와 'Gear'는 모델링 관점에서 '연관'으로 표현되기 때문에, 'Car'와 'Gear'의 관련 여부는 쉽게 파악할 수 있다. 또한, 설계내역

'Parking'에서 메소드 'Forward()'가 'Car'와 'Gear'간의 연관을 보다 구체화하여 명시하고 있음을 알 수 있다. 만일, 두 클래스간의 관련성인 연관의 구체적인 특성과 이의 제어 흐름을 파악하고자 한다면, 시스템은 'Forward()'와 'Car-Gear'간 연관 관련성을 파악하고 있어야 한다. 그러나, Rational Rose는 두 설계내역간의 연관성을 파악하기 위한 수단을 갖고 있지 못하기 때문에 이러한 형태의 검색을 제공하지 못하고 있다.

2.2 Argo/UML

Argo/UML[1]은 규칙에 기반한 보다 진보된 참조 방식을 제공하고 있다. Argo/UML은 설계 시멘틱을 표현하기 위한 일련의 기본 규칙들을 제공한다. 개발자는 기본 규칙들을 조합하여 검색 패턴에 맞는 규칙을 재정의할 수 있으며, 그 결과로 얻어진 설계내역으로의 항해를 제공한다.

예를 들어, 기본 규칙들인 'Project->Package', 'Package->Base Class', 그리고 'Class->Subclass'의 조합인 'Inheritance-centric'의 상속 관계를 이용하여 'CBaseRept'의 하위클래스들을 파악할 수 있다. 뿐만 아니라 기본 규칙 'Class->Attribute'의 추가를 통해 관련 속성도 함께 추출할 수 있다. 이와 같은 방식은 Rational Rose의 'Navigation Tree'를 보다 확장한 방식으로, 개발자가 설계요소 추출을 위한 방식과 결과 범위를 지정할 수 있다는 장점이 있다. 하지만, 기본 규칙들은 'Navigation Tree'의 표현 방식을 지정하기 위해서만 사용된다. 즉, 규칙에 특정 설계요소를 파라미터화할 수 없어, "클래스 'CItemRept'로부터 상속되는 모든 클래스들을 추출하라"와 같은 질의는 처리하지 못한다.

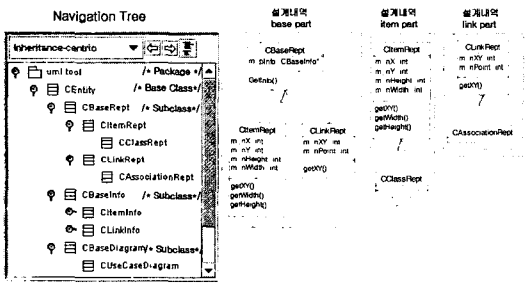

그림 4 Argo/UML Design Model

또한, Argo/UML은 설계요소의 표현정보만을 분석하여 인식하고 있다. 즉, 규칙을 통해 클래스-연관과 오브젝트-메소드식의 1차적인 설계 시멘틱은 파악할 수 있지만, 연관-메소드와 같은 설계 시멘틱들간의 연계는 이루어지지 않고 있다. 즉, "클래스 'CLinkRept'와 동적으로 바인딩되는 클래스들을 추출하라"와 같은 질의를 지원하지 못한다.

만약, 설계도구를 통해 'CLinkRept'가 다른 설계내역에 존재하는 클래스와 '연관'으로 표현된다면, '연관'은 메소드 'getXY()'와 관련성을 갖는 시퀀스 다이어그램내 메시지에 의해 보다 구체적으로 명시된다. 이 관련성을 통해, 사용자나 개발자들은 클래스들간의 동적인 흐름을 쉽게 파악할 수 있다. 하지만, Argo/UML은 개별적인 설계 시멘틱의 구체적인 특성을 파악하지 못해, 이러한 분석 정보를 제시하지 못하고 있다.

2.3 Cool:Jex

Cool:Jex[13,14]는 'Class Browser'를 통해 설계내역간 '복제' 관련성을 갖는 설계요소들을 파악할 뿐만 아니라, 클래스들의 상속 관계를 파악할 수 있는 참조 구조를 제공한다. 또한, 설계내역 'Main'에 존재하는 클래스 'Car'의 모델링 관점에서 '연관'에 의해 관련을 갖는 클래스 'CDPlayer'와 'Gear'를 쉽게 파악할 수 있다. 그러나, 이 설계도구 역시 연관-메소드와 같은 설계 시멘틱들간의 연계를 통해 '인스턴스' 관련성을 갖는 설계요소들로의 항해는 제공하지 못한다는 점에서 한계를 가지고 있다.

즉, 'Class Browser'는 클래스-연관의 관계만을 파악할 뿐 'Car'와 'Gear'간의 관련성인 연관의 특성을 통해 클래스간의 제어 흐름을 파악하지 못하고 있다. 한 예로, 설계내역 'Classifier:Flow'의 메시지 'Forward()'를 통해 'Car'와 동적으로 연결된 'Gear'를 파악하지 못한다.

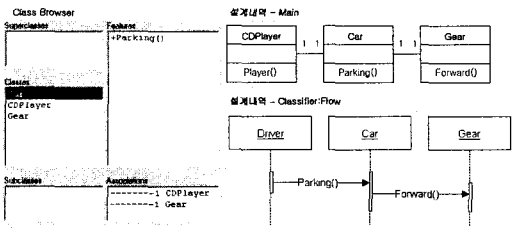

그림 5 Cool:Jex Design Model

3. 함수 기반 설계내역 항해기

이 절에서는 제한적인 참조만을 지원하는 기존의 UML 설계 도구[1, 10]들의 단점을 해결하고, 설계요소들의 설계 시멘틱을 명확하게 파악하고 검증할 수 있는 설계내역 항해기에 대해 기술한다. 설계내역 항해기는

설계요소들간의 관련성을 나타내는 복제, 인스턴스, 그리고 전이 관계 등으로 상호 연관된 설계내역들간의 항해를 지원한다.

복제 관련성은 이미 존재하는 설계요소가 다른 설계내역에 복사되어 다시 생성되는 경우에 생성되는 관련성이며, 항해기가 이를 사용해 복사된 설계요소로(혹은 복사한 설계요소로) 항해하는 것이 가능하다. 인스턴스 관련성은 클래스 다이어그램에 존재하는 설계요소가 시퀀스 다이어그램의 오브젝트로 생성되는 경우이며, 전이 관련성은 클래스 설계요소의 메소드로부터 시퀀스 다이어그램의 메시지로 생성되는 경우를 뜻한다.

3.1 항해기 구조

그림 6은 함수 기반으로 관련 설계요소들간의 검색 및 항해를 지원하는 설계내역 항해기의 구조이다. 항해기의 주요 요소인 EasyDesigner[17]는 [1, 10, 13]이 제공하는 기본적인 UML 설계내역의 작성뿐만 아니라 사용자의 정의에 따라 설계요소들간의 관련성을 함께 파악하고 유지하기 위한 기능을 추가로 포함하고 있다. 즉, EasyDesigner는 UML 설계내역이 생성될 때, 설계내역들간의 관련성을 자동으로 추출한다.

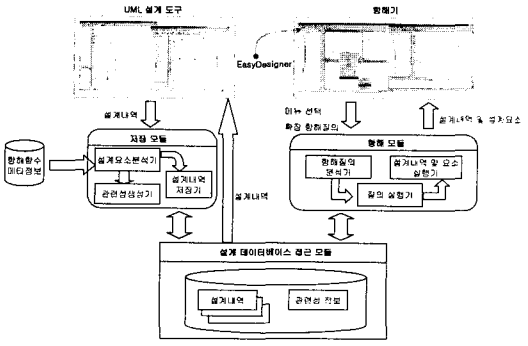

그림 6 설계내역 항해기 구조

이때, 관련성과 같은 설계 시멘틱은 저장 모듈을 통해 다음절에서 설명할 함수 기반 저장모델로 변환되어 저장된다. 관련성 정보는 설계요소간의 관련 정보를 파악하고, 해당 설계요소로의 참조를 위해 사용된다. 즉, 개발자가 항해 인터페이스를 통해 설계내역을 파악하고자할 경우, 항해 인터페이스는 저장된 설계내역과 관련성 정보를 바탕으로 사용자의 요구를 일련의 설계 데이타베이스에 대한 확장 SQL로 생성한다. 확장 SQL은 표준 데이타베이스 언어인 SQL에 다양한 설계 시멘틱을 포함하도록 정의되었다. 예를 들어, "select C.SUB.name from class as C where C.name = 'CBaseRept'"는 클래스 'CBaseRept'에서 직접 상속된 하위클래스들을 추출하기 위한 질의이다(물론, "select C.recur(SUB).name ..."을 통해 모든 하위클래스의 추출도 가능하다). 확장 SQL은 표준 SQL에 설계 시멘틱을 표현하기 위해 확장되었으며, 그 결과 검색 모듈을 통해 실제 설계 데이타베이스에 대한 SQL 처리와 그 결과와 대한 사상과 변환을 거쳐 개발자에게 제공된다.

이와 같이, 항해 인터페이스는 자체의 설계 데이타베이스에 접근하기 위한 고유의 표현 모델과 질의를 제공하기 때문에, 다른 도구들[1, 10, 13]에 비해 설계 과정에서 생성되는 다양한 관련성들에 대한 융통성 있는 검색과 얻어진 설계요소로의 다양한 항해 수단을 제공할 수 있다. 설계요소와 이들간의 관련성에 따라 항해하기 위해서는 파악된 설계요소들간의 관련성을 표현하고, 질의하기 위한 일련의 연산들을 제공하는 항해 모델이 필요하다.

3.2 관련성 정보 표현 방안

이 절에서는 항해 모델에서 관련성을 표현하는 방법에 대해 설명한다.

3.2.1 관련성의 구조

항해 모델은 설계내역의 모든 설계요소와 그들간의 관계를 명세하기 위해 고안되었다. 항해 모델에서, 모든 설계요소들은 저장시스템으로부터 유일하게 할당된 식별자로 구분된다. 이 식별자는 전 설계 과정에서 한 번만 생성되고, 재사용 되지 않는 것을 특징으로 한다. 따라서, 설계요소들간의 관련성을 명세할 때, 각 설계요소를 구분하기 위해 식별자를 사용한다. 여기에서, 설계요소는 클래스, 오브젝트, 액터, 그리고 유스케이스 등을 예로 들 수 있으며, 관련성의 종류는 클래스 설계요소들간의 상속 관련성 혹은 클래스 설계요소와 오브젝트 설계요소와의 인스턴스 관련성, 연관과 메소드의 구체화 관련성 등이 있다.

항해 모델에서 관련성은 다음과 같이 세 개의 요소를 갖는 스키마로 정의한다.

설계요소간의 관련성=(id_1, R, id_2)

여기에서, id_1, id_2는 설계요소의 식별자이고, R은 id_1에서 id_2로의 관련성을 의미한다. 예를 들어, 클래스 'Car'와 'Gear'가 'Car-Gear' 연관 관계성이 있다면, 두 설계요소간의 관련성은 ('Car', 'Car-Gear', 'Gear')와 같이 명세된다. 그러나, 연관 관계성 자체도 한 설계요소로 간주되어, 또 다른 설계요소와 새로운 관계성(그림 1에서 연관 r_1과 메소드 m_i와 같은)이 도출될 수 있다. 따라서, 항해 모델에서 위의 예에 대한 실제 표현은

아래와 같은 형태로 표현된다.

설계요소간의 관련성=(700, out_ass, 720)

700='Car'의 식별자

out_ass=외향 연관

720='Car'와 'out_ass'이 명시한 관련성을 통한 '연관'의 식별자

설계요소간의 관련성=(720, forward_item, 740)

720='연관'의 식별자

forward_item=외향 ITEM

740='연관'과 'forward_item'이 명시한 관련성을 통한 'Gear'의 식별자

또한, 관련성이 '상속'과 같이 관련성 정보의 흐름이 있어, 설계요소에 따라 해당 관련성이 다르게 해석되어야 할 필요가 있기 때문에도 구분되어야 한다. 예를 들어, 상속 관계를 갖는 클래스 계층구조가 있다면, 상위 클래스는 '상속한다'라는 관련성이 부여되고, 하위클래스는 '상속받는다'라는 관련성이 부여되어야 할 것이다.

이후, 본 논문에서는 관련성 표현을 단순화하고, 설계요소들간의 설계 시멘틱을 보다 명료하게 표현하기 위해, 설계요소간의 관계성을 그림 7과 같이 객체식별자를 노드로 갖고 관련성의 종류를 라벨로 갖는 그래프로 시각화하도록 한다.

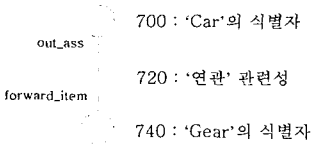

그림 7 관련성 그래프 구조

그림 7에서 'Car'.'연관'를 통해 'Gear'를, 역으로 'Gear'.'연관'을 통해 'Car'를 추출할 수 있다. 물론 앞에서 언급한 바와 같이 항해 모델의 실제 표현으로는 각각 700.out_ass.forward_item와 740.forward_item.out_ass가 될 것이다.

3.2.2 표현 방안

이 절에서는 EasyDesigner를 사용해 설계한 설계내역들이 항해모델에서 표현되는 구체적인 예와 항해 방법에 대해 설명한다.

그림 8은 클래스 'StudentInformation' 설계요소의 관련 시멘틱을 항해하기 위해 필요한 관련성을 표현하고 있다. 즉, 항해정보에서 상위 클래스 'UserInformation'외의 관련성은 부모 클래스로의 항해를 위한 관계성 'out_gen'을 통해 표현되며, 연관 관계가 있는 클래스 'CourseOffering'은 연관 관계의 방향을 고려한 'in_ass'를 통해 표현된다.

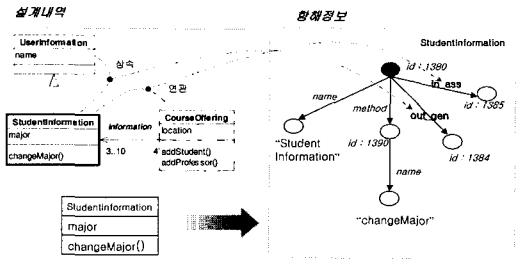

그림 8 설계요소중 '클래스' 객체의 표현 모델

그림에서 클래스 'StudentInformation'는 애트리뷰트 'major'와 메소드 'changeMajor()'로 명세되었고, 또한 'UserInformation'과 상속 관계성을 'CourseOffering'와는 연관 관계성이 있음을 알 수 있다. 항해 모델에서, 'StudentInformation'는 유일한 식별자 '1380'을 통해 구별되며, 세부 특성인 'name', 'attribute', 'method'와 기본 관계성 'out_gen', 그리고 'in_ass'를 갖고 있다. 만일 세부 특성이 복수개 존재한다면 복수개가 명세될 것이다(단, 클래스 정의에 따라 클래스 이름을 명세하는 'name'과 식별자를 명세하는 'id' 특성은 유일하게 하나만 명세된다).

클래스 항해정보에서, 'out_gen', 그리고 'in_ass'는 관련성을 갖는 클래스들을 참조하기 위해 추가된 속성들이다. 'out_gen' 속성은 'StudentInformation'와 'User Information' 사이의 관련성으로 '외향 상속'을 의미하며, 'in_ass'는 'StudentInformation'와 'CourseOffering' 사이의 관련성으로 '내향 연관'을 의미한다. 관련성이 있는 정보들을 검색하고자 할 때에는 해당 정보와의 관련성을 표현하는 개개의 속성을 통해 직접 참조할 수 있다. 즉, 1380.name을 통해 클래스의 이름을 추출할 수 있으며, 1380.method.name을 통해 모든 메소드의 이름을 추출할 수도 있다. 또한, 1380.out_gen을 통해 자신과의 상속을 명시하는 관계성에 대한 정보를 추출할 수 있다.

이와 같이, 특정 설계요소에서 관련성을 갖는 다른 설계요소를 파악하기 위해서는 이들간의 관계를 표현하는 설계요소들이 추가로 필요하며, 이러한 설계요소들도 자체의 독자적인 표현 구조를 갖는다. 이 설계요소들은 '연관(association)', '집합(aggregation)', '종속(dependency)', 그리고 '상속(generalization)' 등이 있다.

예를 들어, 그림 9의 설계내역에 존재하는 연관 'information' 설계요소는 클래스 'StudentInformation'

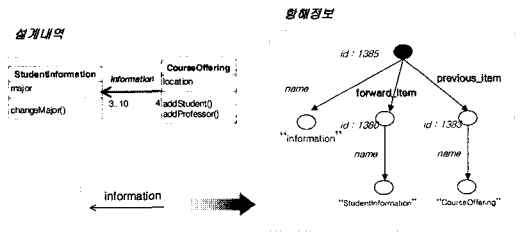

그림 9 설계요소 '연관' 객체의 표현 모델

와 'CourseOffering'간의 관련성을 표현하고 있으며, 세부 속성들로는 'previous_item', 'forward_item'이 있다. 이때, 'previous_item'은 'CourseOffering' 클래스를 참조하기 위한 관련성 정보이고, 'forward_item'은 'StudentInformation' 클래스를 참조하기 위한 관련성 정보이다.

따라서, 1380.in_ass로 얻어진 'information'에 대해 정의된 연관 관계성(=1385)의 이름을 추출하기 위해서는 1385.name 혹은 1380.in_ass.name로 얻을 수 있으며, 연관 'information'을 통해 관련되는 클래스들의 이름을 얻기 위해서는 1385.*.name와 같이 함으로서 'StudentInformation'과 'CourseOffering'을 얻을 수 있다.

3.2.3 같은 종류의 설계내역들간 관련성 표현

설계내역들은 다수의 설계내역들과 관련성으로 연결되어 있다. 이 절에서는 이들 관련성에서 같은 종류의 설계내역들간의 관련성에 대해 설명한다. 같은 종류의 설계내역이라 함은 관련성으로 연관된 두 설계내역이 동일한 분류 그룹임을 의미한다. 예를 들어, 상속관계 r_1이 있는 클래스 C_1과 C_2는 모두 클래스 그룹이므로, r_1에 대해 C_1, C_2는 같은 종류의 설계내역이라 할 수 있다.

앞 절에서 보인 설계요소의 관련성 정보 추출은 미리 정의된 관련성 항해함수의 수행 결과로 얻게 된다. 이 절에서는 항해 모델에서 관련성에 따라 항해하고 결과를 추출하기 위해 정의한 항해함수에 대해 설명한다.

항해 함수 F는 항해의 시작이 되는 설계요소의 객체식별자와 항해할 관련성 서브함수명, 그리고 그 결과로 대상 객체식별자 집합이 얻어진다.

F : 서브함수명 × 객체식별자 → {객체식별자}

서브함수는 모델내에서 설계요소들간의 항해를 위해 정의하였다. 예를 들어, 클래스 C_1의 하위클래스들의 객체식별자는 C_1.in_gen.previous_item으로 얻을 수 있으며, 적용될 서브함수는 아래와 같다.

F_{GetSub}(객체식별자) : 객체식별자집합

여기에서, GetSub에 의해 수행되는 C_1.in_gen.previous_item을 살펴보면, '.'은 명시한 관련성(in_gen과 같은)과 관련된 새로운 설계요소를 지칭하기 위해 사용된다. 'in_gen'은 일반을 나타내는 설계요소의 식별자를 얻고, 다시 얻어진 식별자로 'previous_item'을 사용하여 하위클래스로 명시된 설계요소들의 식별자를 얻을 수 있다. 항해함수는 서브함수를 조합하여 사용자에게 보다 유용한 함수를 구성하는데 사용된다.

모델내에 가능한 객체식별자 집합이 M이라 할 때, 적용되는 항해함수 F의 정의는 다음과 같다.

$M=\{o_1,o_2,...,o_n\}$: 모델내 객체 식별자

$F_{함수명}(M)=M' \cup F_{함수명}(M')$이고, 여기서,

$M'=\{o' \mid o' \in F_{서브함수명}(o_i),\ i=1, 2, ..., n\}$

여기에서, $\{o_1,o_2,o_3,...\}$와 M'내의 객체식별자를 갖는 설계요소는 함수명을 파악할 수 있는 속성을 갖는다. 이를 다시 표현하면, 다음과 같다.

함수 : DEFINE <항해함수명>
AS <서브함수명>
BASED ON <객체식별자>

예를 들어, C_1으로부터 상속된 모든 하위클래스를 얻는 항해함수 SUB는 아래와 같이 정의할 수 있다.

DEFINE SUB AS GetSub
BASED ON <C_1의 객체식별자>

정의된 SUB 함수의 실행과정을 기술해보면 다음과 같다.

$F_{SUB}(\{o\})=M' \cup F_{SUB}(M')$
$=\{F_{GetSub}(o)\} \cup F_{SUB}(\{F_{GetSub}(o)\})$
$=\{o_1,o_2,o_3,...\} \cup \{F_{GetSub}(o_1), F_{GetSub}(o_2),$
$F_{GetSub}(o_3), ...\} \cup F_{SUB}(\{F_{GetSub}(o_1),$
$F_{GetSub}(o_2), F_{GetSub}(o_3)\}, ...)$

그림 10은 다른 설계내역에 존재하는 복제한 설계요소들간의 관련성 정보에 관한 내용을 시각화한 것이다.

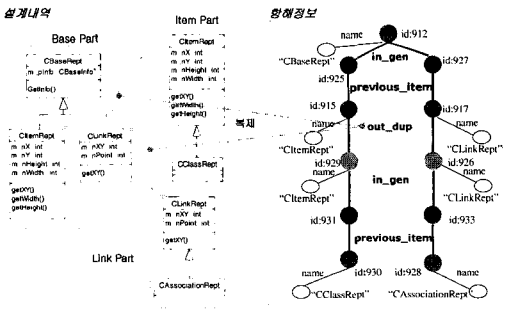

그림 10 같은 종류의 설계내역들간 관련성과 항해 정보

그림에서, 설계내역 'Base Part'내의 클래스 'CBase Rept'로부터 상속받는 클래스들을 검색하고자 한다면, 항해정보에 표현된 'CBaseRept'의 상속 관련성 정보인 'in_gen'과 'previous_item'으로부터 클래스 'CItemRept', 'CLinkRept'를 추출할 수 있다. 다시, 참조된 클래스들의 '복제' 관련성 정보인 'out_dup'를 통해 설계내역 'Item Part'와 'Link Part'에 존재하는 'CClassRept'와 'CAssociationRept'를 반복적으로 추출할 수 있다.

같은 종류의 설계내역내 관련성을 갖는 설계요소들을 검색하기 위한 항해 인터페이스(4절 구현 참조)는 내부적으로 확장 질의를 생성하며, 확장 질의는 다음과 같이 항해함수명과 서브함수가 적용되어 처리된다.

$F_{SUB}(\{912\}) = \{F_{GetSub}(912)\} \cup F_{SUB}(\{F_{GetSub}(912)\})$
$= \{915, 917\} \cup \{F_{GetSub}(915), F_{GetSub}(917)\}$
$\cup F_{SUB}(\{F_{GetSub}(915), F_{GetSub}(917)\})$
$= \{915, 917\} \cup \{928, 930\} \cup F_{SUB}(\{F_{GetSub}(928), F_{GetSub}(930)\})$
$= \{915, 917, 928, 930\}$

이와 같이, 분석자의 다양한 의도를 만족시키기 위해, 의미적으로 연관성을 갖는 모든 설계요소들에 대한 접근 체계를 제공하고, 이를 운용하는 항해 함수들의 동적인 처리를 통해 의도된 분석정보를 제공할 수 있는 반면, Rational Rose등은 분석자에게 제시되는 정보가 문법적인 수준에서 미리 분석되고 어떠한 가공처리도 가해질 수 없는 정적형태의 정보로 제공된다는 점에서 본 논문에서 제시한 방법과 차별화될 수 있다.

3.2.4 다른 종류의 설계내역들간 관련성 표현

지금까지는 이미 생성된 설계요소의 내용과 동일한 내용을 갖는 설계요소를 포함하는 같은 종류의 설계내역들 사이의 관련성 표현을 다루었다. 이 절에서는 다른 종류의 설계내역들간 설계요소들의 관련성 정보를 설명한다. 이 경우 클래스 다이어그램만으로는 정보의 흐름을 파악하기 어렵기 때문에, '인스턴스' 관련성을 부여하도록 한다.

예를 들어, C 클래스에 속하는 m 메소드 수행에 직간접적으로 영향을 받는 클래스들을 모두 파악할 수 있는 항해함수 $F_{IMPACTCLASS}$는 아래와 같이 정의할 수 있다.

$M = \{o_1, o_2, ..., o_n\}$: 모델내 객체 식별자

$F_{IMPACTCLASS}(M) = M' \cup F_{IMPACTCLASS}(M')$이고, 여기서,

$M' = \{o' \mid o' \in F_{GetImpactClass}(o_i)$, $i = 1, 2, ..., n\}$

이 함수의 실행과정을 자세히 기술해보면 다음과 같다.

$F_{IMPACTCLASS}(M) = M' \cup F_{IMPACTCLASS}(M')$
$= \{o'_i\} \cup F_{IMPACTCLASS}(\{o'_i\})$이고,
$o'_i \in F_{GetImpactClass}(o_i)$, $i = 1, 2, ...$
$= \{o'_1, o'_2, ...\} \cup \{F_{GetImpactClass}(o'_1),$
$F_{GetImpactClass}(o'_2), ...\} \cup$
$F_{IMPACTCLASS}(\{o''_i\})$ 이고, $o''_i \in$
$F_{GetImpactClass}(o'_i)$, $i = 1, 2, ...$

그림 11은 '인스턴스' 관련성을 갖는 설계요소들간의 정보를 파악하는 예이고, 그림 12는 이들 설계요소들간의 관련성 정보를 그래프로 표현한 것이다.

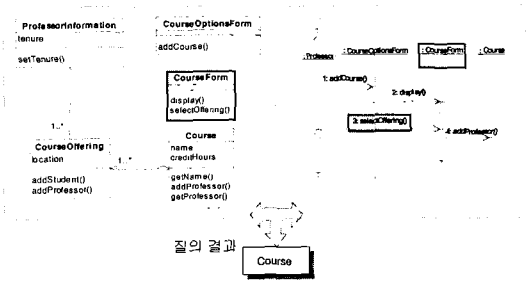

그림 11 다른 종류의 설계내역들간 관련성

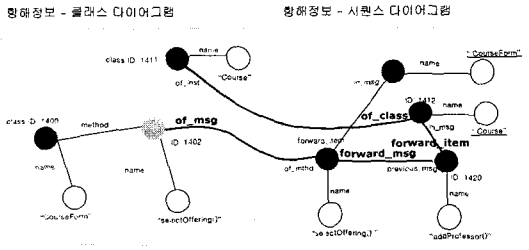

그림 12 다른 종류의 설계내역들간 항해 정보

그림 11에서 설계내역(클래스 다이어그램)에 존재하는 클래스 'CourseForm'의 메소드 'selectOffering()'의 실행시, 영향받는 클래스들을 검색한다고 가정해 보자. 먼저, 그림 12에서 'CourseForm' 식별자 1400으로부터 메소드 명을 통해 'selectOffering()'을 추출한다. 다시, '전이' 관련성중 공유하는 메시지를 참조하기 위해 'of_msg'를 통해 항해내역(시퀀스 다이어그램)의 메시지 'selectOffering()'를 참조한다. 이후, 이 메시지로부터 다음에 실행되는 메시지를 참조하기 위한 관련성 정보 'forward_msg'를 통해 메시지 'addProfessor()'를 추출하고, 이 메시지를 포함하는 오브젝트를 참조하기 위한 관련성 정보 'forward_item'을 통해 오브젝트 ':Course'를 추출할 수 있다. 마지막으로, ':Course'와 정보를 공유하는 클래스를 참조하기 위한 관련성 정보 'of_class'를 통해 항해내역(클래스 다이어그램)의 클래스 'Course'가 참조된다. 이처럼, 검색하고자하는 클래

스는 '1400.method.of_msg.forward_msg.forward_item.of_class'와 같은 순차적인 관련성 정보들을 통해 참조가 가능하다.

다른 종류의 설계내역내 관련성을 갖는 설계요소들을 검색하기 위한 항해 인터페이스(4절 구현 참조)는 내부적으로 확장 질의(3.3절 질의어 참조)를 생성하며, 확장 질의는 다음과 같이 항해함수명과 서브함수(부록 A 참조)가 적용되어 처리된다.

검색 클래스 :

$F_{IMPACTCLASS}(\{1400\})$
$=\{F_{GetImpactClass}(1400)\} \cup F_{IMPACTCLASS}(\{F_{GetImpactClass}(1400)\})$
$=\{1411\} \cup \{F_{GetImpactClass}(1411)\} \cup F_{IMPACTCLASS}(\{F_{GetImpactClass}(1411)\})$
$=\{1411\}$

시스템의 구조 및 정보공유, 그리고 제어흐름을 명확히 파악하기 위해서는 동적인 정보를 표현하는 시퀀스 다이어그램내 메시지들의 파악뿐만 아니라, 이들의 제어 순서도 중요하다. 항해 도구에서는 이러한 순서 정보를 유지하기 위해 순서화된 리스트를 사용하며, 모든 관련성에 적용된다.

이와 같이, 다른 설계내역들간의 자유로운 항해는 관련된 설계내역들간(예, 클래스 다이어그램의 메소드와 시퀀스 다이어그램의 메시지)의 연관성을 명확히 파악하여야만 가능하다는 관점에서, 항해기는 이러한 형태의 분석을 지원하지 못하는 도구[1, 10, 13, 14]들과 구분될 수 있다. 물론, 다른 도구들[10, 17]은 변경작업에 대한 일관성 등의 제한적인 기능을 제공하고 있지만, 본 논문에서와 같이 분석과 그 결과를 사용한 항해 방법을 제공하지는 못하고 있다.

3.3 질의어

이 절에서는 설계내역 항해기가 제공하는 항해 질의어를 설명한다. 항해 질의어는 표준 데이타베이스 시스템의 질의어인 SQL을 확장하였다. 기존의 SQL이 제공하지 않는 문법은_그래프로 표현된 관련성 정보에 맞게 재구성하여 제시하였고, 이들은 내부적으로 해당 관련성을 표현하기 위한 함수로 확장된다.

질의는 임의의 설계요소와 관련성을 갖는 설계요소를 파악하고자 할 때 사용된다. 예를 들면, 그림 10에서 클래스 'CBaseRept'와 상속 관련성을 갖는 다른 설계요소를 검색하고자 할 때 항해 인터페이스에서 함수와 관련된 메뉴를 선택하게 되면 다음과 같은 확장 질의가 생성된다.

생성된 질의:

```
SELECT C.SUB.name
    FROM class_repository as C
    WHERE C.name='CBaseRept'
```

여기에서, 'SUB'는 앞서 정의한 항해함수로 특정 클래스와 상속 관련성을 갖는 클래스의 추출을 명시하고 있으며, 조건절에서 대상 클래스를 'CBaseRept'로 제한하고 있다. 결과적으로, 1) 제한된 'C'에 대해 정의된 함수 'SUB'가 적용되고, 2) 결과내에서 해당 클래스의 이름을 얻는다. 이때, 설계요소는 일대일의 관계만을 갖는 것이 아니라 일대다의 관계를 갖기 때문에, 'C.SUB'의 검색 내용은 여러 개 일 수 있다. 실제적인 질의 처리는 그림 13과 같이 항해함수와 서브함수들의 조합으로 이루어진다.

$t1 = F_{GetOid}(\text{"CBaseRept"})$
$t2 \in F_{SUB}(t1)$
$\text{printf("\%s",}F_{GetName}(t2))$

그림 13 SUB 질의 처리

여기에서, $F_{GetOid}()$는 클래스 'CBaseRept'의 식별자 '912'를 얻고, t_1('912')을 인자로 갖는 함수 SUB를 통해 '915', '917'을 얻는다. 이때, t2는 t1에 의해 얻어지는 객체 식별자이며, t2의 'name' 속성에 의해 'CBaseRept'의 하위클래스 이름들이 검색된다.

다른 예로, 그림 12와 같이 인스턴스를 통해 관련을 갖는 설계요소들을 포함하는 설계내역들간 관련성 파악을 위한 참조는 항해 인터페이스로부터 확장 질의가 생성된다.

생성된 질의 :

```
SELECT C.IMPACTCLASS.name
FROM class_repository AS C
WHERE C.name='CourseForm' and
C.method.name='selectOffering()'
```

여기에서, 질의의 결과는 클래스 'CourseForm'의 메소드 'selectOffering()'이 실행된 후 영향을 받는 클래스가 검색된다. 'IMPACTCLASS'는 검색된 메시지를 포함하는 오브젝트가 참조되고, 참조된 오브젝트의 '인스턴스' 관련성을 통해 참조되는 클래스를 검색한다. 관련성을 갖는 클래스 검색을 위한 실질적인 질의 처리는 그림 14와 같이 항해함수와 서브함수들의 조합으로 이루어진다.

$t1 = F_{GetOid}(\text{"selectOffering()"})$
$t2 \in F_{IMPACTCLASS}(t1)$
$printf(\text{"\%s"}, F_{GetName}(t2))$

그림 14 IMPACTCLASS 질의 처리

앞서 정의한 항해함수와 서브함수들은 이러한 SQL 수준의 질의를 통해 UML 설계 도구로 작성된 설계내역내의 관련 설계요소들을 쉽게 추출할 수 있다.

4. 구 현

이 절에서는 기호 규칙과 관련성 및 정보의 정확성을 검증하고, 정보의 흐름을 파악할 수 있는 설계내역 항해기의 사용자 인터페이스에 대해 기술한다. 먼저, 항해기는 정보의 흐름을 쉽게 파악할 수 있는 클래스 다이어그램과 시퀀스 다이어그램으로 구성되며, 이들 다이어그램내 관련성을 갖는 설계요소로의 항해를 위한 항해 인터페이스는 팝업 메뉴로 제공되고, 선택된 메뉴에 의해 내부적으로 질의가 재구성된다.

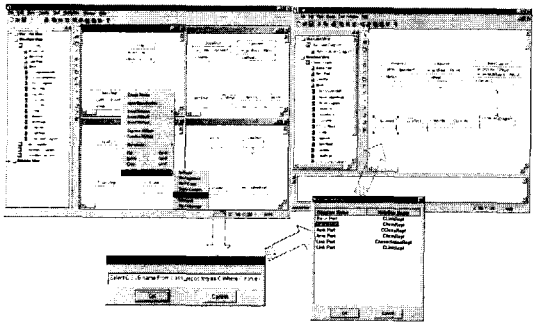

그림 15 Sub Classes의 항해 인터페이스와 결과

그림 15는 클래스 'CBaseRept'로부터 상속을 받는 모든 클래스들을 검색하고 항해하는 과정을 보이고 있다. 먼저, 1) 'CBaseRept'의 팝업 메뉴인 'Navigation' - 'Sub Classes'를 선택하면, 내부적으로 "SELECT C.SUB.name FROM class_repository as C WHERE C.name = 'CBaseRept'"와 같은 표준 질의가 자동으로 제시되며, 2) 창을 통해 제시된 수행결과에서, 항해하고자하는 설계요소를 선택함으로써, 3) 실제 설계내역으로 항해가 이루어진다.

좀 더 일반화된 예로써, 그림 16은 클래스 'Couse Form'의 메소드 'selectOffering()'이 실행된 후 영향을 받는 클래스를 항해하는 인터페이스를 보이고 있다.

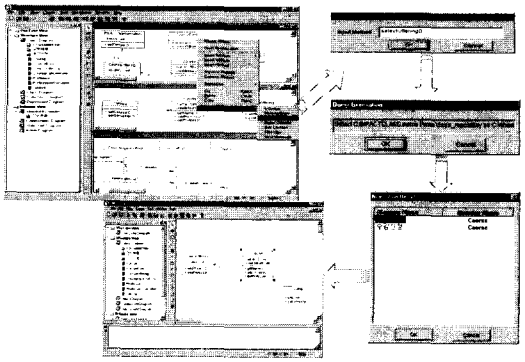

그림 16 Next Class의 항해 인터페이스

이를 위해, 1) 'CourseForm'의 팝업 메뉴인 'Navigation' - 'Next Class'를 선택하고, 해당 메소드를 입력하면 2), 내부적으로 "SELECT C.IMPACTCLASS.name FROM class_repository AS C WHERE C.name='CourseForm' and C.method.name='selectOffering'"과 같은 질의가 구성된다. 3) 질의 결과로 얻어진 '인스턴스' 관련성 중에서 항해하고자 하는 설계요소를 선택함으로써, 해당 설계요소를 포함하는 설계내역으로의 항해가 이루어진다.

5. 결론 및 향후 연구 과제

UML은 소프트웨어의 분석, 설계, 구현의 전 개발주기를 지원하는 매우 효과적인 개발 방법론이며, 객체 지향 개발 방법론의 대표적인 것이라 말할 수 있다.

본 논문에서는 이러한 UML 설계 도구에서 생성되는 설계요소들간의 관련성을 통해 시스템을 구성하는 정보의 흐름을 파악하고 검증할 수 있는 항해기에 대해 기술하였다. 정제된 의미 분석 정보를 제공하기 위해, 관련성 표현과 관련성 함수 정의를 포함한 항해 모델을 제시하였고, 또한 개발자가 다양한 관련성 정보를 보다 융통성 있게 검색할 수 있도록 하기 위해 표준 SQL을 항해 모델에 맞게 확장하여 제공하고 있다. 이를 통해, 소프트웨어 개발 중에 얻은 설계요소들간의 관련성을 파악하고 설계내역들간의 항해를 제공함으로써 효과적으로 소프트웨어의 분석과 시스템의 전체 흐름을 이해하는데 도움을 줄 수 있다. 또한, GUI기반으로 이러한 설계내역의 분석과 항해 과정을 분석자가 직관적으로 인식할 수 있도록 한 항해 인터페이스의 예를 보였다.

계속적으로 이루어져야 할 향후 연구로, 소프트웨어 개발 과정에서 현재 시스템이 정의한 관련성 함수이외에도 보다 특성화된 관련성들이 많이 존재한다. 따라서, 개발자들이 시스템이 정의한 기본 관련성들을 조합하고

재정의를 통해 계속적으로 확장할 수 있어야 한다. 이를 위해, 관련성 함수를 규칙화하고 관리하기 위한 연구가 추가로 필요하다. 또한, 설계 데이타베이스의 진화 내역을 유지하고 관리하기 위한 방법이 필요하다. 이 방법은 현재의 설계내역이 이전의 설계내역에서 진화된 방식과 그 결과 영향받는 내역들에 대한 분석을 위해서 반드시 필요하다. 이를 위해서는 항해 모델내에 관련성 정보외에도 버전 정보 및 변경 로그 정보들을 수용하고, 이들에 접근하기 위한 함수들이 추가되어야 한다.

참 고 문 헌

[1] Argo/UML v0.7: The Cognitive CASE Tool, http://argouml.tigris.org/, University of California, Irvine, 1999.

[2] Bock, Conrad. and Odell, James, "A More Complete Model of Relations and Their Part I: Relations as Object Types," Journal Of Object-Oriented Programming, Vol 10, No 3, pp. 38-40, June 1997.

[3] Bock, Conrad. and Odell, James, "A More Complete Model of Relations and Their Implementation, Part II: Mappings," Journal Of Object-Oriented Programming, Vol. 10, No 6, pp. 28-30, October 1998.

[4] Bock, Conrad. and Odell, James, "A More Complete Model of Relations and Their Implementation, Part III: Roles," Journal Of Object-Oriented Programming, Vol 11, No 2, May 1998.

[5] Booch, G., Rumbaugh, J., and Jacobson, I., The Unified Modeling Language User Guide, Addison-Wesley Publication Company, 1999.

[6] Grady Booch, Ivar Jacobson, and James Rumbaugh. Unified Modeling Language Version 1.0. Rational Software Corporation. January 1997.

[7] Hamie, A., Howse, J., and Kent, S., "Navigation Expressions in OO Modelling," Proceedings of FASE98 at ETAPS98, pp. 123-137, March 1998.

[8] Kohler, Hans J., Ulrich Nickel, Jorg Niere, Albert Zundorf, "Integrating UML Diagrams for Production Control Systems," Proceedings of the 2000 International Conference on Software Engineering, pp. 241-251, 2000.

[9] Motschnig-Pitrik, Renate, Kaasboll, Jens, "Part-whole relationship categories and their application in object-oriented analysis," IEEE Transactions on Knowledge and Data Engineering, Vol. 11, Issue 5, pp. 779-797, 1999.

[10] Rational Soft, Corp, "Rational Rose 2000," http://www.rational.co.kr/Product/Rose/.

[11] Rational Soft. Corp, "Unified Modeling Language," http://www.rational.com/uml, 2000.

[12] Robbins, J. E. and Redmiles, D. F., "Cognitive support, UML adherence, and XMI interchange in Argo/UML," Information and Software Technology, Volume 42, Issue 2, pp. 79-89, 25 January 2000.

[13] Sterling Soft, Corp, "Cool:Jex," http://cool2.sterling.com/support/.

[14] Telelogic Company, http://www.telelogic.com/.

[15] UML 1.3 Specification. OMG Documents ad990608-ad990609.

[16] 배명남, 최 완, 양현택, "웹을 사용한 객체지향 설계정보 분석", 한국정보과학회 논문지(C), Vol. 27, No. 7, pp. 702-711, 2000.

[17] 양재동, 최동운, 최재훈, 김기헌, 김원중, "객체지향형 설계정보항해기의 구현을 위한 OMT/UML 도구의 개발", 산업자원부 최종보고서, pp. 92, 2000.

부록 A. 항해함수명과 서브함수명

ITEM 복제(Duplication) 관계		
항해함수명	서브함수명	항해 정보
F_{INDUP}	$F_{GetInDup}$	in_dup
F_{OUTDUP}	$F_{GetOutDup}$	out_dup

내향 연관 관계		
항해함수명	서브함수명	항해 정보
F_{INCNT}	$F_{GetInCnt}$	in_gen.previous_item

외향 연관 관계		
항해함수명	서브함수명	항해 정보
F_{OUTCNT}	$F_{GetOutCnt}$	out_ass.forward_item

외향 집합 관계		
항해함수명	서브함수명	항해 정보
F_{WHOLE}	$F_{GetWhole}$	out_agg.forward_item

내향 집합 관계		
항해함수명	서브함수명	항해 정보
F_{PART}	$F_{GetPart}$	in_agg.previous_item

외향 종속 관계		
항해함수명	서브함수명	항해 정보
$F_{SUPERORD}$	$F_{GetSuperOrd}$	out_dep.forward_item

내향 종속 관계		
항해함수명	서브함수명	항해 정보
F_{SUBORD}	$F_{GetSubOrd}$	in_dep.previous_item

외향 상속 관계		
항해함수명	서브함수명	항해 정보
F_{SUPER}	$F_{GetSuper}$	out_gen.forward_item

내향 상속 관계		
항해함수명	서브함수명	항해 정보
F_{SUB}	F_{GetSub}	in_gen.previous_item

클래스의 오브젝트 관계		
항해함수명	서브함수명	항해 정보
F_{OBJECT}	$F_{GetObject}$	of_inst
		of_msg.forward_item

오브젝트의 클래스 관계		
항해함수명	서브함수명	항해 정보
F_{OWNER}	$F_{GetOwner}$	of_class
		forward_item.of_class

메소드의 메시지 관계		
항해함수명	서브함수명	항해 정보
F_{MSG}	F_{GetMsg}	of_msg
		of_inst.in_msg

메시지의 메소드 관계		
항해함수명	서브함수명	항해 정보
F_{MTHD}	$F_{GetMthd}$	of_mthd
		forward_item.of_class.method

오브젝트 & 오브젝트 관계		
항해함수명	서브함수명	항해 정보
$F_{IMPACTOBJECT}$	$F_{GetImpactObject}$	of_inst.in_msg.previous_item
		of_msg.forward_msg.forward_item

김 원 중

1999년 전북대학교 전자계산학과 졸업(학사). 2001년 전북대학교 대학원 전산통계학과(석사). 2002년 ~ 현재 전북대학교 대학원 컴퓨터통계정보학과 박사과정. 관심분야는 객체지향 개발 방법론, UML, CASE, 객체지향 데이타베이스

배 명 남

1991년 전북대학교 전산통계학과 졸업(학사). 1993년 전북대학교 대학원 전자계산학과(석사). 1998년 전북대학교 대학원 전자계산학과(박사). 1998년 ~ 현재 : 한국전자통신연구원 네트워크 S/W 플랫폼팀 선임연구원. 관심분야는 실시간시스템, 객체지향 개발 방법론, UML, CASE

양 재 동

1983년 2월 서울대학교 컴퓨터공학과(학사). 1985년 2월 한국과학기술원 전산학과(석사). 1991년 2월 한국과학기술원 전산학과(박사). 1995년 1월 ~ 1996년 1월 Univ. of Florida, Visiting Scholar. 현재는 전북대학교 전자정보공학부 교수, 영상·정보 신기술 연구소 연구원. 관심분야는 OODBs, Expert System, CASE