UML 객체지향 분석모델의 완전성 및 일관성 진단을 위한 시나리오기반 검증기법

(Scenario-Driven Verification Method for Completeness and Consistency Checking of UML Object-Oriented Analysis Model)

조 진 형 † 배 두 환 ††
(Jin-Hyung Cho) (Doo-Hwan Bae)

요 약 본 논문에서 제안하는 시나리오기반 검증기법의 목적은 UML로 작성된 객체지향 분석모델의 완전성 및 일관성을 진단하는 것이다. 검증기법의 전체 절차는 요구분석을 위한 Use Case 모델링 과정에서 생성되는 Use Case 시나리오와 UML 분석모델로부터 역공학적 방법으로 도출된 객체행위 시나리오와의 상호참조과정 및 시나리오 정보트리 추적과정을 이용하여 단계적으로 수행된다. 본 검증절차를 위하여 우선, UML로 작성된 객체지향 분석모델들은 우선 정형명세언어를 사용하여 Use Case 정형명세로 변환한다. 그 다음에, Use Case 정형명세로부터 해당 Use Case 내의 객체의 정적구조를 표현하는 시나리오 정보트리를 구축하고, Use Case 정형명세 내에 포함되어 있는 객체 동적행위 정보인 메시지 순차에 따라 개별 시나리오흐름을 시나리오 정보트리에 표현한다. 마지막으로, 시나리오 정보 트리 추적과 시나리오 정보 테이블 참조과정을 중심으로 완전성 및 일관성 검증작업을 수행한다. 즉, 검증하고자 하는 해당 Use Case의 시나리오 정보트리를 이용한 시나리오 추적과정을 통해 생성되는 객체행위 시나리오와 요구분석 과정에서 도출되는 Use Case 시나리오와의 일치여부를 조사하여 분석모델과 사용자 요구사양과의 완전성을 검사한다. 그리고, 시나리오 추적과정을 통해 수집되는 시나리오 관련정보들을 가지고 시나리오 정보 테이블을 작성한 후, 분석과정에서 작성된 클래스 관련정보들의 시나리오 포함 여부를 확인하여 분석모델의 일관성을 검사한다. 한편, 본 논문에서 제안하는 검증기법의 효용성을 증명하기 위해 대학의 수강등록시스템 개발을 위해 UML을 이용해 작성된 분석모델을 특정한 사례로써 적용하여 보았다.

Abstract The purpose of a scenario driven verification method proposed in this paper is to check completeness and consistency of objected oriented analysis model generated by UML(Unified Modeling Language). In this approach, we verify the completeness and consistency between the use case scenario and the UML analysis model represented by object behavior scenario. First, the static structure models and dynamic behavior models generated by UML analysis process are transformed into use case specification via formal specification language. Secondly, the scenario information trees are generated with class structures and scenario flows of each use case specification. Then, completeness and consistency are checked using the scenario information table generated through scenario tracing process with the scenario information tree. The completeness of UML analysis model is checked through cross comparison process between the use case scenario and the object behavior scenario. For consistency checking of UML analysis model, it is needed to compare between the scenario information table and the class structure information. To demonstrate usability of the proposed verification method, we apply it to the UML model of the university course registration system as a case study.

† 정 회 원 : 동양공업전문대학 전산경영기술공학부 교수
cjh@dongyang.ac.kr
†† 종신회원 : 한국과학기술원 전산학과 교수
bae@salmosa.kaist.ac.kr
논문접수 : 1999년 8월 18일
심사완료 : 2000년 12월 26일

1. 서 론

객체지향 소프트웨어 시스템의 개발은 일반적으로 요구분석, 분석 모델링, 설계 모델링, 구현, 검사의 과정을

거쳐 진행되는데, 시스템의 규모가 대형화하면서 분석자가 사용자의 요구사항을 파악하는 업무에 막대한 시간과 노력이 투입되고 있다. 그런데, 초기 요구사항 분석의 오차로 인하여 오류 수정의 비용이 전체 개발 단계별로 막대하게 소모되는 위험성이 있기 때문에, 분석 모델링 단계에서부터의 초기 검증의 중요성이 대두되고 있다. 한편, 기존의 객체지향 소프트웨어 개발 기법들을 통합하여 객체지향 모델링 언어의 통합표준으로 제안되고 있는 UML(Unified Modeling Language)을 이용한 객체지향 개발방법이 급속하게 확산되고 있는 상황이다[3]. UML을 이용한 객체지향 개발방법은 요구분석 과정 중에 Use Case와 여기서 도출되는 시나리오를 기반으로 사용자 요구사항을 모형화 하여 객체지향 분석모델을 생성하는 것을 원칙으로 하고 있다[12]. 그런데, 이러한 과정을 통하여 생성된 분석모델(Analysis Model)은 시스템의 요구사양 명세(Requirement Specification)가 되어 시스템 개발의 이전 단계에서 사용자와 개발자간에 의사소통을 하는 도구로서 이용되기 때문에 설계, 구현, 검사의 중요한 기반이 되는 것이다[8]. 그리고, UML 객체지향 방법의 초기단계에서 수행되는 Use Case 모델링 과정에서 생성되는 시나리오들은 분석 단계에서 작성되는 객체지향 분석모델의 동적행위를 검증하는데 있어 유용한 도구로 이용될 수 있다[15]. 그러나, 기존의 객체지향 모델에 대한 검증기법과 관련된 연구 성과들을 분석해 보면, Use Case나 시나리오를 이용하여 UML 객체지향 분석모델에 대하여 효과적인 검증방법을 제시한 연구성과가 현재까지는 체계화되지 못하고 있는 상태이다.

객체지향 방법은 다른 소프트웨어 개발 방법들에 비해서 상대적으로 요구사항 모형화를 실세계에 가장 가깝게 표현하고자 하지만, 다음과 같은 이유로 사용자 요구사항과 객체지향 모델과의 상이성이 존재할 가능성이 있다[2]. 그 첫번째 이유는 대부분의 객체지향 방법들은 시스템 분석의 복잡성을 해소하기 위하여 객체의 정적 구조와 동적행위 등을 구분하여 각기 다른 관점에서 시스템을 분석 모델링하기 때문에, 객체지향 분석 모델들간의 일관성에 문제점이 존재할 수 있다. 둘째로, 객체지향 분석 단계 중에 분석자는 도메인 지식(Domain Knowledge)을 참조하여 사용자가 제공하는 문제 기술문(Problem Statement)으로부터 불필요한 정보들을 제거하고, 필요한 정보들을 추가하면서 분석 모델링 작업을 수행하기 때문에 사용자의 시스템에 대한 요구사항이 분석모델에 충분하게 반영되지 않을 가능성이 항상 존재한다.

위에서 열거한 상황들을 고려할 때, 현재 객체지향 방법론의 통합 표준 표기법으로 제시되고 있는 UML을 이용하여 생성되는 객체지향 모델의 완전성 및 일관성을 진단할 수 있는 새로운 검증기법에 대한 연구와 개발이 필요하다고 볼 수 있다. 따라서, 본 논문의 연구 방향은 초기단계에서 수행하는 Use Case 모델링 과정에서 생성되는 시나리오를 이용하여, UML 객체지향 분석모델의 일관성(Consistency)과 완전성(Completeness)을 검증하는 과정에 대하여 새로운 기법과 절차를 제시하는 것이다. 본 논문의 구성은 다음과 같이 구성된다. 제2장에서는 기존의 객체지향 모델 검증기법들에 대한 연구성과들에 대하여 기술하는데, 기존의 객체지향 모델간의 일관성 검증을 시도했던 방법들과 정보트리를 이용한 객체지향 검증 방법들에 대하여 분석하고 각각의 문제점들을 정리한다. 제3장에서는 본 논문에서 제안하는 시나리오기반 UML 객체지향 분석모델 검증기법의 전체 절차와 정형 명세화 과정, 시나리오 정보 트리 구조와 구축절차, 시나리오 추적을 통한 분석모델의 일관성 및 완전성 검증절차 등 각 세부 단계별 내용을 상세히 기술한다. 제4장에서는 본 논문에서 제안하는 검증기법의 유용성을 증명하기 위하여 특정한 적용사례를 통해 검증절차와 방법을 설명한다. 마지막으로, 제5장에서는 결론과 함께 본 연구와 관련하여 향후 진행되어야 할 연구 과제들에 대하여 논하기로 한다.

2. 관련 연구

2.1 OMT 모델간의 일관성 검증기법

UML이 실용적인 형식으로 제안되기 이전에 객체지향 개발방법을 선도했던 OMT 방법론에 의해 작성된 객체, 동적, 기능 모델간의 일관성 검증에 주로 집중되었던 연구성과들이 있다[1][7][17][18]. 이 중 Hayes가 제시한 검증기법[7]에서는 객체, 동적, 기능 모델간에 연관성을 참조하여 객체구조 모델(Object Structure Model)이라는 형식기반 모델을 작성하고 전위조건과 후위조건을 추가하여 모델간의 일관성을 검사한다. 그리고, 이 방법과 유사한 것으로서 객체행위 모델을 통한 검증기법[18]이 있다. 여기에서 제시된 검증기법은 객체, 동적, 기능 모델들의 공통된 특성을 모형화하고, 이들 사이의 상호보완 관계성을 파악하여 객체행위 모델(Object Behavior Model)이라는 진단 모델을 설계하여 객체지향 분석 모델간의 일관성을 검사하는 것이다. 위의 두 기법은 세 가지 관점의 각기 다른 모델들의 공통된 특성을 통합하여 일관성을 검증하는 시도로서 새로운 통합 진단모델의 제시의 의미가 있으나 일관성을 검

증하는 구체적인 절차의 제시를 못하였다는 점과 진단 기준을 지나치게 분석가의 경험에 의존하는 검증규칙에 문제점이 있었다. 한편, 지식베이스 구축을 통한 검증기법[17]에서는 객체, 동적, 기능 모델들을 Atomic Formula형태로 각각 정형명세화 하여 응용 지식베이스에 저장한 후 일련의 정의된 오류 검출규칙과 일관성 진단규칙들을 가지고 각 모델간의 불일치성을 검사하는 방법을 제시하였다. 그러나, 이 방법은 지식베이스를 구축하는데 너무 많은 비용이 소모된다는 단점이 있다. 그리고, 유한상태기계(Finite State Machine)에 근거하여 분석모델의 오류를 찾는 방법[1]이 있는데, 이 방법은 사용자 요구사항을 사건중심 요구명세로 표현한 다음 CTL(Computational Tree Logic)기계로 변환한 후 시간적 논리의 유효성을 검사한다. 그런데 이 방법은 검증하려고 하는 시스템의 규모에 따라 상태가 지수함수 형태로 폭발적으로 늘어날 수 있는 문제가 제기될 수 있다.

2.2 정보트리를 이용한 객체지향 분석모델 검증기법

OMT 방법을 이용하여 생성된 객체지향 요구명세의 일관성과 완전성을 검증하기 위해 정보트리(Information Tree)라는 그래픽 표기법을 이용한 기존의 검증기법으로서 정보트리(Information Tree)를 이용한 객체지향 요구명세 검증기법[16][19]과 확장정보트리(Extended Information Tree)를 이용한 객체 동적행위 검증기법[20]이 있다. 이 기법들은 모두 객체지향 분석과정에서 생성되는 객체 모델과 동적 모델을 형식사양 언어를 사용하여 형식사양으로 변환시킨 후 그 형식사양을 정보트리를 중심으로 한 그래픽 표기법으로 구축하여, 정보트리에서 표현되는 정보들을 가지고 자연어로 표현되는 요구문장과의 비교를 통하여 일관성과 완전성을 검증하는 방법을 제시하였다. 정보트리 이용 검증기법에서 제시된 방법론은 분산시스템을 위한 객체지향 소프트웨어 개발에서 요구사양(Requirement Specification)에 대한 검증방법으로서 분석 모델을 자체적으로 제안하는 정형명세 언어를 통해 형식사양으로 변환한 후 형식사양을 정보 트리(Information Tree)로 변환하여 사용자 요구사양 문장과의 상향식(Top Down)비교와 하향식(Bottom Up)비교를 통하여 검증하는 절차가 핵심적인 내용이다[16]. 그리고, 확장정보트리 이용 검증기법의 연구범위는 정보트리 이용 검증기법에서 제안한 정보트리를 수정한 확장 정보 트리(Extended Information Tree)를 가지고 객체간의 오퍼레이션 순차를 추적하여 객체간의 동적인 행위를 검증하는 것으로서, OMT 분석모델의 수정 객체 모델, 수정 동적 모델로의 변환 절차와 OMT 모델의 검증과정에서 Use Case 적용 등을 제안하고 있다.

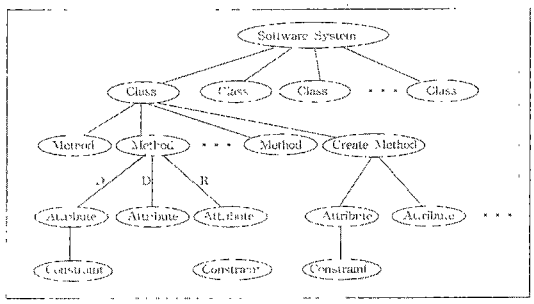

그림 1 정보트리의 구조

2.3 정보트리를 이용한 기존의 검증기법의 문제점

객체지향 분석모델의 요구명세를 정보 트리라는 그래픽 표기법으로 변환하여 완전성과 일관성을 검증하는 방법에 대한 기존의 연구[16][19][20]성과에 대하여 위에서 설명하였다. 이러한 연구성과들은 기존의 OMT 방법론에 의거한 객체지향 분석모델에 대한 검증에 있어 새로운 정형화된 검증절차를 제안하였다는데 나름대로 의미를 갖고 있다. 그런데, 위의 검증 기법들은 객체지향 요구사양과 분석모형과의 완전성 및 일관성 검증 절차에 있어 다소의 복잡성과 모호성을 노출시키고 있다. 그리고, 최근에 객체지향 개발기법의 새로운 통합 모형으로 확산되고 있는 UML을 이용한 개발방법론에 적용하는데 있어서도 상당한 문제점들을 노출시키고 있다. 따라서, 보다 향상된 객체지향 분석모델에 대한 검증방법을 제안하기 위해서는 기존의 방법들의 이러한 문제점들을 보다 체계적으로 분석해 볼 필요성이 있는 것이다.

우선 정보트리 이용 검증기법[19]은 객체지향 분석모델의 검증에 정보 트리라는 그래픽 표기법을 제안함으로써 검증방법을 구조화하였다는 데에 그 의미가 있다. 그런데, 이 방법은 향후 개선해 나가야 할 몇 가지 문제점들을 포함하고 있으며, 그 문제점들을 분석해보면 다음과 같다.

1) 그래픽 표기 중복성의 문제

전체 시스템의 정적 구조 및 동적행위를 표현하는데 있어 정보 트리 이외에 상속 그래프 및 변이추적 테이블을 사용함으로써, 검증 절차의 복잡성과 세 가지 그래픽 표기들을 모두 구축해야 하는 부담이 문제가 된다. 이것은 이 방법에서 제안하는 정보트리가 시스템의 정적 구조와 객체간의 동적행위를 통합하여 표기하는 것이 불가능하므로, 동적행위를 별도의 그래픽 표기로 구축하는 것에서 발생하는 문제점이다.

2) 시스템 규모에 따른 정보트리 표현의 한계의 문제

위의 방법에서 제안하는 정보트리의 루트 노드는 항상 전체 시스템을 표현하므로, 하나의 정보트리로 전체 시스템 요구사양을 모두 표현해야 한다. 그러나, 시스템의 규모가 대규모인 경우에는 표현해야 하는 요구사양 정보의 양이 엄청나게 증가하므로 하나의 정보트리로 전체 시스템 사양을 모두 표현하는데 문제가 발생하게 된다.

3) 문제 기술문과의 직접 비교에 따른 동적행위 검증의 정확도 저하의 문제

정보트리로 표현되는 객체지향 분석모델과 자연어로 작성된 요구 사양인 문제 기술문과의 직접적인 비교를 함으로써, 부분적인 동적행위의 검증은 가능하지만 시스템의 전체적인 동적행위를 검증하는 데에 있어 정확도가 저하될 가능성이 있다.

한편, 정보트리 이용 검증기법을 확장하여 연구된 확장정보트리 이용 검증기법[20]은 동적행위 확장정보트리를 제안하여 시스템의 정적 구조와 동적행위를 하나의 그래픽 표기로 통합 표현함으로써, 객체간의 동적인 메시지 흐름을 추적할 수 있는 방법을 제시하였다는 데에 그 연구의 의의가 있다. 그러나, 이 방법도 기존의 정보 트리 이용 방법이 갖고 있는 문제점들을 충분히 극복하지 못한 점이 있고, 현재 객체지향 개발 방법론의 통합모델로 제시되고 있는 UML을 이용한 개발방법론에 적용하는 데에는 한계를 지니고 있다. 이 방법의 문제점들을 정리해보면 다음과 같다.

1) 전체 시스템 동적행위 표현의 한계와 복잡성의 문제

정보트리 이용 검증기법과 마찬가지로 하나의 트리 구조에 전체시스템의 요구사양의 표현을 시도함으로써 표현의 한계와 복잡성의 문제를 발생시키게 된다. 특히, 각 클래스간의 동적행위인 오퍼레이션 순차를 트리 구조 내에 횡적으로 나열함으로써 확장 정보트리가 그래프화 되는 오류를 발생시키고, 점선 화살표로 표시되는 객체간의 메시지 흐름의 표기는 시스템을 구성하는 클래스의 수가 많은 경우에는 그 표현이 불가능하게 되는 문제가 있다.

2) 수정 객체지향 요구명세 작성의 문제

위의 방법은 객체지향 분석과정에서 작성된 객체지향 요구명세를 직접 정형화시키지 않고, 중간 단계로서 객체지향 요구명세를 수정 객체지향 요구명세로 변환시키는 과정을 포함하고 있다. 이 변환 과정은 검증하려고 하는 전체 분석모델에 대하여 모두 각각 변환하는 절차를 거쳐야 하기 때문에 전체 검증절차에 있어서 엄청난 부담이 요구되는 작업일 수밖에 없다. 그리고, 위의 방법에서 채택하고 있는 OMT 방법이 아닌 다른 객체지향 분석모델의 경우에는 수정 객체지향 요구명세로의 변환과정의 적용이 어렵다는 것이 중요한 문제점이다.

3) Use Case 적용상의 모호성 문제

위의 방법은 정보트리 이용 검증기법에서 제시되었던 정보트리의 메시지 흐름과 문제기술문과의 직접 비교에서 발생하는 검증의 정확도 저하의 문제를 극복하기 위해서, 사용자 문제 기술문을 Use Case로 변환하여 확장정보트리와 비교하는 절차를 제시하고 있다. 객체지향 분석모델 자체가 초기 요구분석 단계에서부터 분석모델 생성단계까지 Use Case를 기반으로 작성되었을 때에 객체지향 분석모델과 Use Case를 비교하는 것이 의미가 있는 것이다. 그런데, 위의 방법에서는 분석모델 생성과정은 Use Case와 전혀 무관하게 진행시키고, 자연어로 작성된 문제 기술문을 비교하기 위해 단순히 변환하는 과정에서만 Use Case를 적용시킴으로 인해서 검증절차의 모호성을 내포하고 있다.

3. 시나리오기반 검증기법

3.1 검증기법의 특징

본 논문에서 제안되는 시나리오기반 UML 객체지향 분석모델 검증기법의 특징을 요약하면 다음과 같다.

1) 객체간 동적행위 검증에 유효

사용자 중심의 시스템 추상화 도구인 Use Case 및 동적행위 순차의 표현인 시나리오를 검증기법에 이용함으로써, 개발하고자 하는 시스템의 정적인 구조뿐 아니라 사양에 포함된 객체 상호간의 동적행위검증에 효과적으로 이용될 수 있다. 또한, 사용자가 시나리오를 통하여 시스템 개발을 통해 구현하고자 하는 요구사항의 정확한 표현을 확인하기가 용이하다. 그러므로, 시스템 개발의 전 단계에서 이루어지는 검증절차에서 시나리오를 분석자와 사용자와의 의사소통의 도구로 이용함으로써 사용자 이해도가 높은 검증을 도모할 수가 있다.

2) 검증절차의 단순화

기존의 검증기법 중에는 분석모델을 정형화하거나 그래픽 표기로 변환하기 전에, 또 하나의 추상화 단계를 거쳐서 여러 가지 시각에서 형성된 분석모델들을 통합모델로 표현하거나 저장하는 절차를 포함하는 기법들[19][20]이 있다. 그런데, 객체지향 분석과정에서 생성된 분석모델들을 검증하기 위해서, 또 한번 변환하고 추상화하는 과정은 엄청난 노력과 시간을 요구하게 된다. 그러나, 본 논문에서 제시되는 검증기법은 이러한 절차들의 생략을 통해 분석모델 검증절차의 복잡성을 해소함으로써, 검증에 소요되는 시간과 비용을 절감할 수 있게 하였다.

3) UML 객체지향 모델에 적합

기존의 객체지향 모델의 검증기법들은 주로 OMT 방법에 의해 생성된 객체, 동적, 기능 모델들의 검증에 주력해 왔다. 그러나, 본 논문에서 제안되는 시나리오기반 검증기법은 UML기반의 객체지향 요구분석 과정 중에 생성되는 Use Case를 검증절차에 이용함으로써, UML 객체지향 분석모델 검증에 가장 적합하도록 구성하였다. 그리고, 분석단계 뿐 아니라 향후 설계단계 및 구현단계에서 생성되는 객체지향 모델들에 대한 검증에도 시나리오를 이용함으로써 확장 적용이 가능하다.

4) 대규모 시스템 사양 검증에 유용

기존의 검증 기법들은 전체 시스템 사양을 하나의 그래픽 표기로 표현함으로써, 시스템의 규모가 대형화할 경우에 검증절차가 매우 복잡하게 되는 문제점이 있었다. 본 논문에서 제안되는 시나리오기반 검증기법에서는 복잡한 시스템의 요구사양을 다수개의 Use Case로 분리하여 표현함으로써, 대규모 시스템의 검증절차에 유용하게 적용될 수 있다.

3.2 검증기법의 전체 절차

본 논문에서 제안하는 UML 객체지향 분석모형에 대한 검증기법은 요구분석을 위한 Use Case 모델링 과정에서 생성되는 Use Case 시나리오 기술문과 객체지향 분석과정에서 생성된 UML 분석모델에서 역공학적 방법으로 도출된 객체행위 시나리오와의 일치성 비교를 통해 완전성을 검사하고, 시나리오 추적과정에서 도출되는 시나리오 정보 테이블을 통하여 분석모델 간의 일관성을 검사하는 방법으로 수행된다. 이러한 검증절차를 위하여 UML로 작성된 객체지향 분석모델들은 우선 첫 번째 단계로, 정형명세언어를 사용하여 Use Case 정형명세(Formal Use Case Specification)로 변환된다. 그 다음에, 둘째 단계로서 Use Case 정형명세로부터 해당 Use Case에 포함된 객체들의 정적구조를 표현하는 시나리오 정보트리(Scenario Information Tree)를 구축하고, Use Case 정형명세 내에 포함되어 있는 객체 동적 행위 정보인 메시지 순차에 따라 개별 시나리오 흐름을 시나리오 정보트리에 표현한다. 다음, 셋째 단계로 시나리오 정보트리의 메시지 순차를 추적하면서 객체행위 시나리오 흐름을 생성하고, 이 과정에서 포함되는 클래스 정보들을 시나리오 정보 테이블에 표시한다. 마지막 단계로 생성된 객체행위 시나리오를 요구분석과정의 Use Case 시나리오와 비교하여 객체 동적행위의 완전성을 검증하고, 시나리오 정보 테이블을 통해 UML 분석모델의 일관성을 검증한다. 시나리오기반 객체지향 분석모델 검증기법의 전체 절차는 그림 2와 같다.

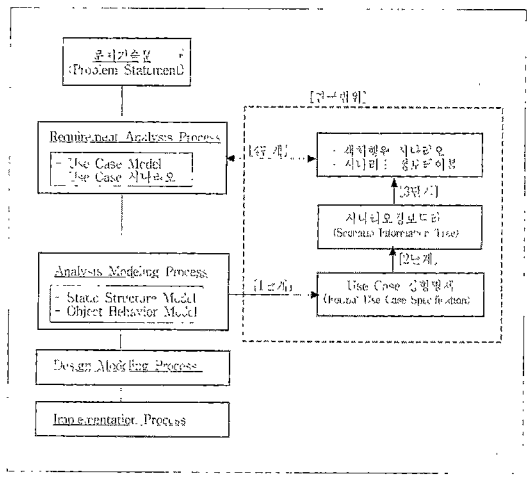

그림 2 시나리오기반 검증기법 전체절차

3.3 Use Case 정형명세 변환

3.3.1 Use Case 정형명세 구조

UML을 이용한 객체지향 숲트웨어의 분석과정은 Use Case를 기반으로 진행되기 때문에 UML객체지향 분석모델의 요구명세(Requirement Specification)는 Use Case의 집합으로 정의될 수 있다[6]. 그래서 시스템의 요구명세는 Use Case 정형명세로 구성되며 그 구조는 그림 3과 같다.

```
Use Case Spec USECASE_NAME
{
Actors              A1,A2,...,Al
Classes             C1,C2,...,Cm
Class Structure     CS1,CS2,...,CSn
Scenario Flow       SF1,SF2,...,SFo
}

// USECASE_NAME  Use Case 명칭
   Ai,  1≤i≤l     해당 Use Case관련 Actor명칭
   Ci,  1≤i≤m     해당 Use Case관련 Class명칭
   CSi, 1≤i≤n     해당 Use Case관련 Class Structure 명세
   SFi, 1≤i≤o     해당 Use Case의 파생 시나리오흐름 명세
//
```

그림 3 Use Case 정형명세 구조

1) 클래스구조 명세 표현

이 부분은 Classes부분에서 선언한 해당 Use Case에 포함된 클래스들의 정적인 구조를 표현하는 것으로써, 각 클래스를 구성하는 속성(Attribute), 오퍼레이션(Operation)과 관련성(Relation)부분을 표현한다. 클래

```
Class Structure CLASS_NAME
 {
  Attributes     A1,A2,...,Aa
  Operations     O1,O2,...,Ob
  Relations      R1,R2,...,Rc
 }
  // CLASS_NAME   클래스 명칭
     Ai, 1≤i≤a    속성명칭
     Oi, 1≤i≤b    오퍼레이션명칭
     Ri, 1≤i≤c    관련성표현 //
```

그림 4 클래스 구조 명세 구조

스구조 명세의 구조는 그림 4와 같다.

2) 시나리오흐름(Scenario Flow) 명세 표현

해당 Use Case에서 발생되는 시스템 외부의 액터와 내부의 객체들간의 동적행위(Dynamic Object Behavior)의 시간적 흐름으로서의 시나리오를 객체간의 전달되는 오퍼레이션들의 순차로서 표현한다. 시나리오는 특정한 Use Case에서 파생되는 인스턴스(instance)의 개념으로서 하나의 Use Case에서 다수 개의 시나리오 흐름이 존재할 수 있다. 시나리오 흐름 명세 부분의 표현 구조는 그림 5와 같다.

```
Scenario Flow Scenario_ID
{
  Message(Scenario_ID, Sequence #, SND_ID, OP_ID, RCV_ID)
                          .
                          .
                          .
  Message(Scenario_ID, Sequence #, SND_ID, OP_ID, RCV_ID)
}
    // Scenario_ID : 시나리오명칭
       Sequence # : 오퍼레이션이 실행되는 순서에 따른 번호
       SND_ID :    오퍼레이션을 발생시키는 액터 혹은 클래스 명칭
       OP_ID  :    수행되는 오퍼레이션 명칭
       RCV_ID :    오퍼레이션을 받는 클래스 명칭//
```

그림 5 시나리오 흐름 명세 구조

3.3.2 Use Case 정형명세 변환

객체지향 분석과정에서 객체의 정적인 구조는 클래스 다이어그램을 통하여 표현된다. 시스템을 구성하는 각 객체의 정적인 구조는 클래스변환을 통하여 명세화 하고, 각 클래스간의 관계는 관련성 변환을 통하여 명세화 한다. 그리고, Use Case내에서의 객체의 동적인 행위는 State Diagram, Sequence Diagram, Collaboration Diagram을 통하여 표현된다[6]. 그런데, State Diagram은 하나의 객체내의 동적인 행위를 표현하므로 Use Case의 객체간의 동적인 시나리오를 통하여 요구사양을 검증하려는 본 논문의 취지상 정형화 과정에서 제외한다. 분석과정 중에 생성되는 Sequence Diagram과 Collaboration Diagram은 둘 다 여러 개의 객체간의 동적인 상호작용을 표현하는데, 전자는 시간적인 측면에서의 상호작용을 표현하고 후자는 공간적인 측면에서의 상호작용을 표현한다[6]. 본 논문에서는 시나리오 흐름 명세를 통하여 객체간의 동적행위를 시간적인 순차로 표현해야 하므로, 동적행위모델의 정형 명세변환을 위해서 Sequence Diagram을 사용하기로 한다. 시나리오흐름 명세 변환과정은 그림 6과 같이 표현된다. 우선, 분석과정에서 생성되는 각 Use Case 시나리오의 객체간의 동적인 행위를 표현하는 Sequence Diagram에서 객체간에 전달되는 각 메시지들을 참조하여 Use Case 정형명세의 시나리오흐름 명세구조 부분의 오퍼레이션으로 변환한다. Sequence Diagram에서 표현되는 각각의 메시지들은 그 메시지들을 수신하는 클래스의 오퍼레이션과 일대일 관계로 대응된다[12]는 사실에 근거하여 시나리오 흐름명세 변환에 참조한다. 그리고, Sequence Diagram내에서 표현되는 각 메시지들에 대한 시간적인 제약조건(Timing Constraint)이나 반복조건(Iteration Condition)들은 시나리오 명세의 오퍼레이션 명칭부분에 괄호 ( )로써 표현한다.

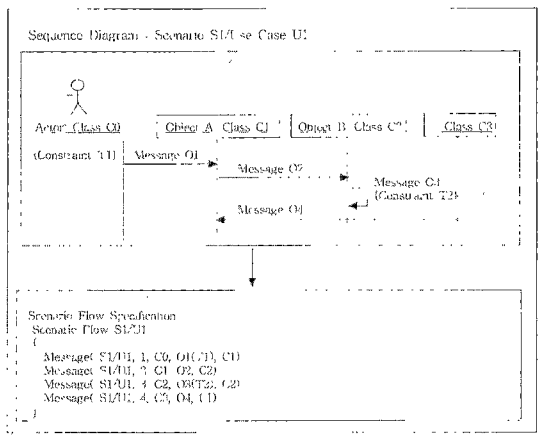

그림 6 시나리오흐름 명세 변환

3.4 시나리오 정보트리(Scenario Information Tree) 구축

3.4.1 시나리오 정보트리의 구조

본 논문에서 제안하는 시나리오 정보트리는 분석과정 중에 생성되는 각 Use Case를 구성하는 여러 객체

들의 정적인 구조를 트리 구조로 표현하고, 그 객체간의 동적인 상호 작용을 오퍼레이션의 시간적인 순차로서 추적하여 시나리오를 표현 가능하도록 한 그래픽 표기 방법이다. 기존의 객체지향 모델 검증 기법들[19][20]에서 제안했던 그래픽 표기법들은 전체 시스템 사양을 하나의 그래픽 표기로 표시하려고 하는 데서 발생하는 표현의 복잡성과 객체간의 동적인 상호작용을 하나의 그래픽 표기로 효과적으로 표시하지 못하는 문제점들이 있었다. 그러나, 시나리오 정보트리는 전체 시스템을 사용자의 관점에서 생성되는 Use Case 단위로 분리하여 표기하기 때문에 사용자 관점의 검증이 용이할 뿐 아니라, Use Case내에서 표현되는 객체간의 상호 작용인 메시지 흐름을 추적하여 시나리오를 검증하는 것이 가능하므로 기존의 표기법들의 문제점들을 개선할 수 있다는 데 그 의의가 있다. 시나리오 정보트리의 전체적인 구조는 그림 7과 같으며, 검증 대상인 해당 Use Case의 명칭이 트리의 루트 노드가 된다. 그리고, 해당 Use Case 정형명세의 클래스 구조 명세부분으로부터 클래스명은 클래스 노드로, 오퍼레이션 부분은 오퍼레이션 노드로, 속성 부분은 속성 노드로 각각 표현된다. 해당 Use Case의 시나리오 정보트리로부터 추출될 수 있는 정보를 크게 두 가지로 보면 하나는 Use Case를 구성하는 클래스들의 정적인 구조에 관한 정보들이고, 또 하나는 시나리오 정보트리 내에서 해당 Use Case의 시나리오를 추적해가면서 얻을 수 있는 각 객체간의 오퍼레이션 순차에 관한 정보들이다.

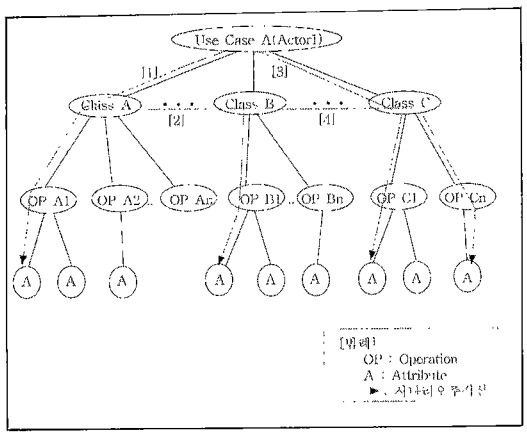

그림 7 시나리오 정보트리 구조

3.4.2 시나리오 정보트리의 구축절차

1) 정적 구조 구축

우선, 검증하려는 객체지향 분석모형의 해당 Use Case 정형명세로부터 시나리오 정보트리를 구축하는 절차부터 설명하기로 한다. 해당 Use Case 정형 명세의 Use Case 명칭을 정보트리의 루트 노드로 변환하고 Use Case 정형명세에 선언된 Actor들은 루트 노드에 괄호 안에 Use Case 명칭과 병행하여 표시한다. 그리고, Use Case 정형명세 내에 선언된 클래스 명칭들을 정보트리의 클래스 노드로 변환하여 표시한다. 다음, 클래스 구조(Class Structure) 명세 부분에 정의된 해당 클래스들의 오퍼레이션 부분과 속성부분들을 정보트리의 오퍼레이션 노드와 속성 노드들로 각각 변환한다. 그림 8은 Use Case정형명세로부터 시나리오 정보트리의 정적구조로 변환하는 예에 대하여 보여주고 있다.

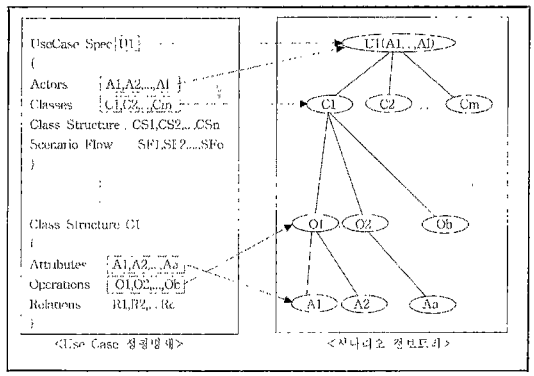

그림 8 시나리오 정보트리 정적구조 구축

2) 동적행위 변환

시나리오 정보트리 내에서 객체간의 동적인 상호작용은 Use Case 정형명세에 표현된 시나리오 흐름명세 부분을 참조하여 해당 시나리오별로 객체간에 전달되는 메시지에 해당되는 오퍼레이션 순차를 화살표를 가진 점선으로 정보트리 노드를 순회하며 표시한다. 이 때, 각 메시지별로 시간적인 순차번호를 표시한다. 이러한 시나리오 추적 과정에서 방문된 노드들은 해당 Use Case를 구성하는 정적인 정보들로 다음절에 설명될 시나리오 정보 테이블에 표시되어지며 이러한 정보들을 가지고 분석모델들의 상호 일관성을 검증하게 된다. 시나리오 흐름 명세부분에 표현된 각 메시지들은 그 메시지를 전송하는 객체에 해당하는 클래스로부터 수신하는 객체의 클래스의 해당 오퍼레이션으로 연결하여 표시한다. 이 때, 해당 오퍼레이션이 결과 값으로 제공하는 속성부분이 연관되어 있을 경우는 속성부분에 대한 검증을 위하여 속성 부분까지 연장하여 표시한다. 그리고, 해당 메시지에 관련된 제약조건이 정형명세부분에 표시되어 있는 경우에는 이를 시나리오 정보트리의 오퍼레

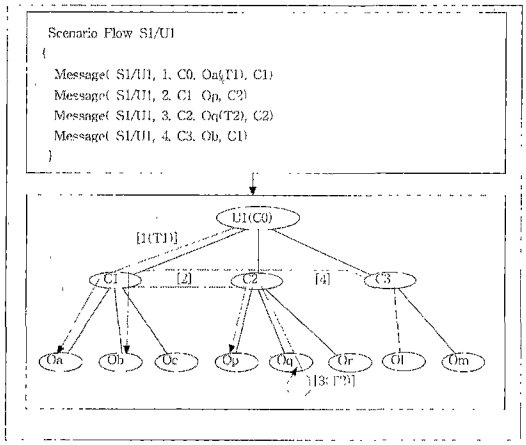

그림 9 시나리오 정보트리 동적행위 변환

이선 노드 부분에 표시한다. 시나리오 정보트리의 동적행위 변환절차의 예는 그림 9에 보여진다.

3.5 일관성 및 완전성 검증과정

시나리오기반 UML 객체지향 분석모델에 대한 검증기법의 마지막 단계로서, 일관성(Consistency)과 완전성(Completeness)에 대한 검증 작업은 다음과 같이 시나리오 정보트리 추적과 시나리오 정보 테이블 참조 과정을 중심으로 수행된다. 우선, 검증하고자 하는 해당 Use Case의 시나리오 정보트리를 구축하고 UML 분석모델의 해당 Use Case 정형명세의 메시지 순차에 따른 시나리오 추적과정을 통해 생성되는 객체행위 시나리오와 요구분석 과정의 Use Case 기술문에서 도출되는 Use Case 시나리오와의 일치여부를 조사하여 분석모델과 사용자 요구사양과의 완전성(Completeness)을 검사한다. 그 다음에 시나리오 추적과정을 통해 수집되는 시나리오 관련 정보들을 가지고 그림 10의 예와 같이 시나리오 정보 테이블을 작성한 후 분석과정에서 작성된 클래스 관련 정보들의 시나리오 포함 여부를 확인하여 분석모델의 일관성(Consistency)을 검사한다.

3.5.1 완전성 검증 절차

객체지향 분석모델의 완전성에 대한 검증 작업은 요구분석 과정인 Use Case 모델링 과정의 Use Case 기술문에서 도출되는 Use Case 시나리오와 시나리오 정보트리 추적을 통해 생성되는 객체행위 시나리오와의 일치성 여부를 비교함으로써 수행된다.

1) Use Case 시나리오 작성

UML을 이용한 객체지향 개발과정의 초기 단계로써 수행되는 요구분석 과정 중에 생성되는 Use Case로부터 Use Case 시나리오를 도출해낸다. Use Case는 주로 자연어로 작성되는 Use Case 기술문을 통하여 표현되거나 또는 별도의 UML 그래픽 표기법인 Activity Diagram을 통하여 표현되기도 한다[6]. 본 논문에서는 사용자 위주의 검증 작업을 도모하기 위해 자연어로 작성된 Use Case 기술문으로부터 Use Case 시나리오를 도출해내기로 한다. Use Case 시나리오는 객체와 메시지로 구성되며 객체들 사이의 동적행위의 흐름을 시간 의존적인 객체간의 메시지들의 순차로써 표현한다. Use Case 시나리오의 메시지 흐름은 <객체1> [메시지(제한조건)] <객체2>와 같이 표현하며, 객체1에서 객체2로 메시지를 보낸다는 것을 의미한다.

2) 객체행위 시나리오 작성

분석모델의 Use Case 정형 명세화 과정을 통하여 구축된 시나리오 정보트리에 점선으로 표시되는 객체간의 동적행위의 표현인 메시지 순차를 추적하여 해당 Use Case에 대하여 객체행위 시나리오를 도출해 낼 수 있다. 객체 행위 시나리오 역시 상호 비교를 위하여 Use Case 시나리오와 동일한 양식으로 객체와 메시지로써 표현한다. 한편, 시나리오 정보트리의 메시지 순차 추적과정을 통하여 방문어지되는 각 클래스 노드, 오퍼레이션 노드, 속성 노드들은 일관성 검증을 위하여 후에 설명되어질 시나리오 정보 테이블에 표시한다.

3) 시나리오 일치성 비교

해당 Use Case 시나리오를 기준으로 객체행위 시나리오와 객체간의 메시지 순차를 비교하여 사용자가 원하는 기능적인 요구사항과 분석모델에서 모형화한 시나리오와의 일치성을 검사함으로써 사용자의 요구사항이 분석모델에 완전하게 반영되었는가를 판단할 수 있다.

4) 완전성 검증

시스템을 구성하는 모든 Use Case의 시나리오들에 대하여 위에서 제안한 검증 단계들을 반복적으로 수행함으로써 사용자의 요구사항이 분석모델에 완전하게 포함되었는지를 검증할 수 있다.

3.5.2 일관성 검증 절차

1) 시나리오 정보 테이블 작성

분석 모델의 완전성을 검증할 목적으로 해당 Use Case의 개별 시나리오의 메시지 순차들을 추적하여 시나리오 정보트리의 간선 상에 점선으로 객체간의 동적행위의 흐름을 표시하면서 수집되는 클래스 노드, 오퍼레이션 노드, 속성 노드 관련 정보들은 분석 모델의 일관성을 검증하는 데 매우 유용한 정보들이다. 분석모델의 일관성을 검증하고자 하는 목적은 각기 다른 시각에서 모형화된 정적인 모델과 동적인 모델이 상호 충돌없이 일치하는가를 검사하고자 하는 데 있다. 따라서,

분석모델의 정적구조의 표현인 클래스 관련 정보들과 동적행위의 표현인 시나리오와의 비교를 통하여 모델간의 일관성을 검증할 수 있다. 그런데, 분석모델의 일관성을 검증하고자 하면 전체 시스템을 구성하는 클래스 관련 정보들과 각각의 Use Case와의 포함성 여부를 일일이 검사하여야 하는데 시스템의 규모가 커질수록 정보의 양적인 규모와 복잡성이 문제가 될 것이다. 그래서, 본 논문에서는 이러한 검사의 복잡성을 해결하고 전체 시스템 관련 정보들을 효율적으로 비교할 수 있는 시나리오 정보 테이블 작성을 제안하기로 한다. 이 시나리오 정보 테이블의 구조를 설명하면 그림 10과 같이 시스템의 정적구조의 표현인 Class Diagram의 클래스, 오퍼레이션, 속성 정보들을 각각 종적으로 배열하고 객체간의 동적행위의 표현인 시나리오를 해당 Use Case별로 횡적으로 배열하여 시나리오 정보 테이블을 구성한다.

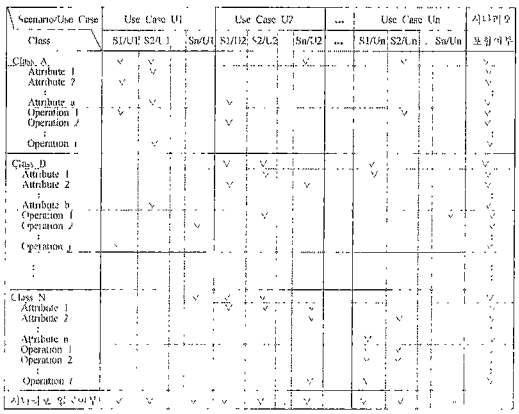

그림 10 시나리오 정보 테이블 작성의 예

2) 시나리오 정보트리 추적

분석 모델의 완전성을 검증 과정 중 수행되는 해당 Use Case의 개별 시나리오의 메시지 순차들의 추적 과정을 통하여 방문되어지는 클래스 노드, 오퍼레이션 노드, 속성 노드들을 시나리오 정보 테이블에 표시한다.

3) 일관성 검증

시나리오 정보 테이블에 종적으로 배열된 Class Diagram으로부터 추출된 모든 클래스, 오퍼레이션, 속성 정보들이 각각 최소한 하나의 시나리오에 포함되었는가를 검사하여 분석 모델간의 일관성 여부를 검증한다.

4. 사례연구

본 논문에서는 UML을 이용한 객체지향 분석모델의 간단한 사례로써 [12]에서 UML 모델링 예제로 제시되고 있는 대학의 수강등록시스템(On-line Course Registration System)의 분석모델 중 강의과목선택 Use Case를 검증대상으로 적용하였다. 본 사례연구에서 검증하고자 하는 강의과목선택 Use Case의 Use Case 기술문은 다음과 같다.

[강의과목선택(Select Course to Teach : UI) Use Case 기술문]

Use Case는 교수가 수강등록시스템에 로그인한 후 자신의 암호(Password)를 입력함으로써 시작된다. 시스템은 암호를 검증한 후에 해당 학기를 선택할 수 있게 한다. 교수는 원하는 학기를 선택한다. 그 다음, 교수는 ADD, DELETE, REVIEW, PRINT 중 하나를 선택할 수 있다.

* ADD 선택: Add a Course Offering 시나리오(S1)

시스템은 Course Name과 Course Number가 포함된 Course Screen을 Display한다. 교수는 Course Name과 Course Number를 입력한다. 시스템은 교수가 선택한 과목에 해당하는 Course Offering(개설과목)을 Display한다. 교수는 원하는 Course Offering을 선택한다. 시스템은 교수와 해당 Course Offering을 Link한다.

* DELETE선택: Delete a Course Offering 시나리오(S2)

시스템은 Course Name과 Course Number가 포함된 Course Screen을 Display한다. 교수는 Course Name과 Course Number를 입력한다. 시스템은 교수와 해당 Course Offering과의 Link를 제거한다.

* REVIEW선택: Review a Schedule 시나리오(S3)

시스템은 해당 교수가 관련된 Course Offering에 관한 Course Name, Course Number, Course Offering Number, Days of the Week, Time, Location의 정보를 Display한다.

* PRINT선택: Print a Schedule시나리오(S4)

시스템은 해당 교수의 강의일정(Schedule)을 Print한다.

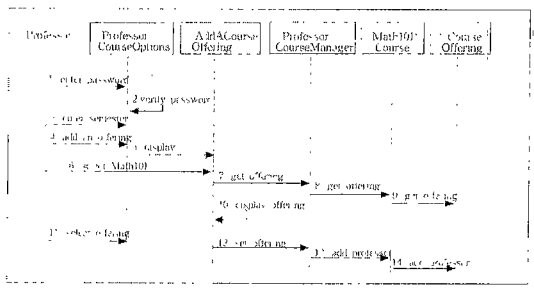

그림 11 강의과목추가(Add a Course Offering) 시나리오에 대한 Sequence Diagram

다음 그림 11은 수강등록시스템의 강의과목선택(Select Course to Teach) Use Case 중 강의과목추가(Add a Course Offering) 시나리오에 대한 Sequence Diagram을 나타낸다.

4.1 Use Case 정형명세 생성

우선 검증절차의 첫 단계로 검증하고자 하는 강의과목선택(Select Course to Teach) Use Case를 정형명

```
Use Case Spec Select Course to Teach
{
  Actors          Professor
    Classes       Course
                  CourseOffering
                  ProfessorInformation
                  ProfessorCourseOption
                  AddCourseOffering
                  ProfessorCourseManager

    Class Structure  Course
     {
       Attributes Course_Name, description, Credit, Hours
       Operations getOffering(), addProfessor(), validateProfessor()
       Relations
      }

    Class Structure  CourseOffering
     {
       Attributes Course_ID, Type, Room#, Hours, Credit
       Operations getOffering(), addProfessor()
       Relations
      }
                  .
                  .
                  .
    Class Structure  ProfessorCourseManager
     {
       Attributes Course_Name, Course_ID, Type, Room#, Hours, Credit
       Operations getOffering(), setOffering(),addProfessor()
       Relations
      }

  Scenario Flow Add a Course Offering
   {
    Message(S1/U1,1,Professor,enterPassword,ProfessorCourseOptions)

Message(S1/U1,2,ProfessorCourseOptions,verifyPwd,ProfessorCourseOption)
    Message(S1/U1,3,Professor,enterSemester,ProfessorCourseOptions)
    Message(S1/U1,4,Professor,addanOffering,ProfessorCourseOptions)
    Message(S1/U1,5,ProfessorCourseOptions,display,AddCourseOffering)
    Message(S1/U1,6,Professor,selectCourse,AddCourseOffering)

Message(S1/U1,7,AddCourseOffering,getOffering,ProfessorCourseManager)
    Message(S1/U1,8,ProfessorCourseManager,getOffering,Course)
    Message(S1/U1,9,Course,getOffering,CourseOffering)

Message(S1/U1,10,AddCourseOffering,displayOffering,AddCourseOffering)
    Message(S1/U1,11,Professor,selectOffering,AddCourseOffering)

Message(S1/U1,12,AddCourseOffering,setOffering,ProfessorCourseManager)
    Message(S1/U1,13,ProfessorCourseManager,addProfessor,Course)
    Message(S1/U1,14,Course,addProfessor,CourseOffering)
   }
                  .
                  .
                  .

 }
```

그림 12 강의과목선택(Select Course to Teach) Use Case의 정형명세

세화 하면 다음 그림 12와 같다.

4.2 시나리오 정보트리 구축

사례연구 부분에서 검증하고자 하는 수강등록시스템의 UML 객체지향 분석모형의 해당 Use Case 정형명세로부터 시나리오 정보트리를 구축하기로 한다. 해당 Use Case 정형 명세의 Use Case 명칭을 정보트리의 루트 노드로 변환하고 Use Case 정형명세에 선언된 Actor들은 루트 노드에 괄호 안에 Use Case 명칭과 병행하여 표시한다. 그리고, Use Case 정형명세 내에 선언된 클래스 명칭들을 정보트리의 클래스 노드로 변환하여 표시한다. 다음, 클래스 구조(Class Structure) 명세 부분에 정의된 해당 클래스들의 오퍼레이션 부분들을 정보트리의 오퍼레이션 노드들로 각각 변환한다. 그림 13은 Use Case 정형명세로부터 시나리오 정보트리로 변환된 상태에 대하여 보여주고 있다.

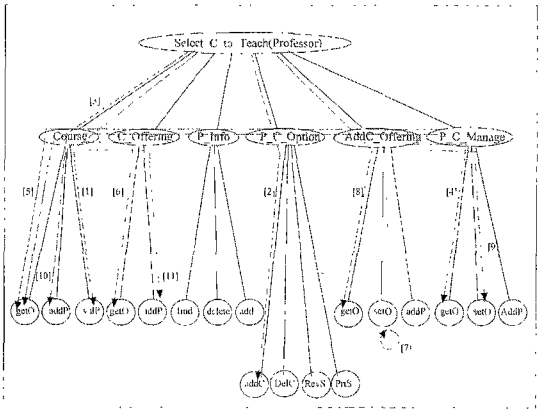

그림 13 시나리오 정보 트리(Add a Course Offering 시나리오 추적선 표현)

4.3 완전성 및 일관성 검증절차

1) 완전성 검증

본 사례연구에서 제시되고 있는 'Select Course to Teach' Use Case 중 'Add a Course Offering' 시나리오와 정형명세화 과정을 통해 변환된 시나리오 정보트리 추적과정을 통해 생성된 객체행위 시나리오와의 일치성 여부를 비교함으로써 검증사례로 제시된 모델의 완전성을 진단할 수 있다. 모델이 완전성(Completeness)을 갖추었다는 것은 사용자의 요구사항이 기능적으로 모델에 완전히 포함되어 있다는 것을 의미한다[2] 고 볼 때, 우선 해당 Use Case의 각 메시지들이 객체행위 시나리오에 각각 포함되고 있는가를 검사해 볼 필요가 있다. 본 사례에서는 그림 14의 Use Case 시나리

오의 각 메시지 흐름과 그림 15의 객체행위 시나리오를 비교해 볼 때 <1>에서 <9>까지 표현된 전자의 각 메시지들은 후자에 일대다 관계로 모두 포함되고 있음을 알 수 있다. 따라서 해당 시나리오(Add a Course Offering)에 한해서 완전성을 갖추었다고 진단한다. 한편, 객체행위 시나리오에서 표현되는 분석모델 중 <6>번, <9>번 메시지와 같이 일대다 관계로 대응되는 모델의 정적구조 중복의 필요성은 일관성 검사부분에서 시나리오 정보 테이블을 통하여 검사한다.

```
* Use Case 시나리오 작성

Use Case 시나리오(Add a Course Offering/Select Course to Teach)

<1> <교수> Password입력<시스템>
<2> <시스템>Password검증<시스템>
<3> <교수>학기입력<시스템>
<4> <교수>Add a Course Offering선택<시스템>
<5> <시스템>Course Screen Display<교수>
<6> <교수>Course Name과 Course Number로 해당 Course선택<시스템>
<7> <시스템>선택한 Course에 해당하는 Course Offering Display<교수>
<8> <교수>Course Offering선택<시스템>
<9> <시스템>해당 Course Offering에 Link<시스템>
```

그림 14 ADD a Course Offering의 Use Case 시나리오

```
* 객체행위 시나리오 작성

객체행위 시나리오(Add a Course Offering/Select Course to Teach)

① <Professor>enterPassword<ProfessorCourseOptions> : <1>
② <ProfessorCourseOptions>verifyPassword<ProfessorCourseOptions> : <2>
③ <Professor>enterSemester<ProfessorCourseOptions> : <3>
④ <Professor>addanOffering<ProfessorCourseOptions> : <4>
⑤ <ProfessorCourseOptions>display<AddCourseOffering> : <5>
⑥ <Professor>selectCourse<AddCourseOffering> : <6>
⑦ <AddCourseOffering>getOffering<ProfessorCourseManager> : <6>
⑧ <ProfessorCourseManager>getOffering<Course> : <6>
⑨ <Course>getOffering<CourseOffering> : <6>
⑩ <AddCourseOffering>displayOffering<AddCourseOffering> : <7>
⑪ <Professor>selectOffering<AddCourseOffering> : <8>
⑫ <AddCourseOffering>setOffering<ProfessorCourseManager> : <9>
⑬ <ProfessorCourseManager>addProfessor<Course> : <9>
⑭ <Course>addProfessor<CourseOffering> : <9>
```

그림 15 ADD a Course Offering의 객체행위 시나리오

2) 일관성 검증

UML 기법을 포함한 객체지향 분석방법들은 정적구조와 동적행위를 각기 다른 관점에서 모형화하기 때문에 정적, 동적 관점에서 모형화한 결과를 비교하여 일관성(Consistency)을 진단할 필요가 있다. 본 사례연구 부분에서는 제시된 UML 모델의 정형화 과정과 시나리오 추적 과정을 통해 생성되는 시나리오 정보 테이블을 통해 해당 사례의 정적구조 모델과 동적행위 모델간의 일관성을 진단하는 절차를 제시하고자 한다. 분석 모델의 완전성을 검증할 목적으로 해당 Use Case의 개별 시나리오의 메시지 순차들을 추적하여 시나리오 정보트리의 간선 상에 점선으로 객체간의 동적행위의 흐름을 표시하면서 수집되는 클래스 노드, 오퍼레이션 노드, 속성 노드 관련 정보들을 시나리오 정보 테이블에 표시하여 분석모델의 일관성을 검증하는 데에 이용한다. 시나리오 정보 테이블에 종적으로 배열된 모든 클래스, 오퍼레이션 정보들이 각각 최소한 하나의 시나리오에 포함되었는가를 반복적으로 검사하여 분석 모델간의 일관성 여부를 검증한다. 분석모델의 일관성 검증은 전체 모델의 시나리오 정보를 포함하여 진행되어야 하는 관계로 전체 시나리오 정보 테이블을 도시해야 하는 문제가 있기 때문에, 본 사례 연구에서는 해당 Use Case(Select Course to Teach: U1)에 해당하는 4개의 시나리오들만을 가지고 검증을 시도하였다. 해당 Use Case의 일관성 검증을 위한 시나리오 정보 테이블은 그림 16과 같다.

시나리오 추적과정을 통해 도출된 그림 16의 시나리오 정보 테이블은 'Select Course to Teach' Use Case 부분에 한정해서 작성된 것이다. 일관성 검증을

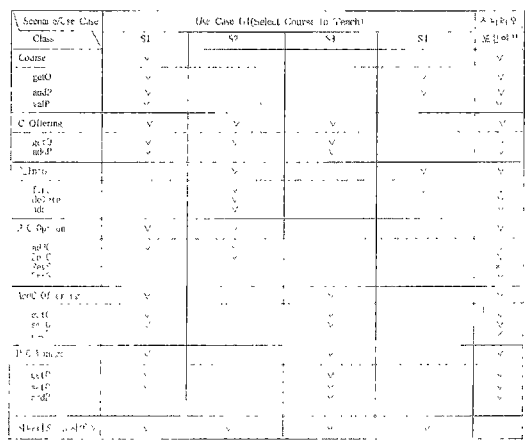

그림 16 시나리오 정보 테이블('Select Course to Teach' 부분)

위해 정보트리의 각 노드들이 각 시나리오 추적과정에 방문되어지는가를 참조한다. 이를 통해 정적구조 모델에서 추상화된 정보들이 동적행위 모델인 각 시나리오들에 포함되어지는지 여부를 진단할 수 있다. 위의 Use Case 경우, P_C_Option 클래스의 RevS 오퍼레이션과 AddC_Offering 클래스의 addP 오퍼레이션이 해당 Use Case에 포함된 시나리오 내에 포함되어지지 않음을 알 수 있다. 이는 해당 Use Case 내에서 정적구조 모델과 동적행위 모델간에 일관성에 문제가 있다는 것으로 판정할 수 있다. 그러나, 이러한 결과가 전체 시스템 분석 모델의 일관성에 문제가 있다고 미리 판정할 수는 없다. 왜냐하면 해당 Use Case의의 모든 Use Case에서 도출되는 시나리오까지 추적해야만 전체 시스템의 일관성을 진단할 수 있기 때문이다. 결과적으로 그림 16의 시나리오 정보 테이블을 통해 해당 Use Case를 표현한 분석 모델은 해당 Use Case에 한해서 완전성을 갖추었다고 볼 수 있고, 일관성에는 문제가 있다고 진단할 수 있는 것이다.

5. 결 론

본 논문에서는 최근 객체지향 소프트웨어 개발방법론의 통합 표준으로 제시되고 있는 UML 표기법을 이용하여 생성되는 객체지향 분석모델의 완전성과 일관성 검사를 위한 방법으로서 시나리오기반 검증기법을 제안하였다. 제안하는 시나리오기반 UML 객체지향 분석모델에 대한 검증기법은 요구분석과정의 Use Case 모델링 과정에서 생성되는 Use Case 시나리오와 객체지향 분석과정에서 생성된 UML 분석모델에서 역공학적 방법으로 도출된 객체행위 시나리오와의 상호비교를 통해 완전성을 검사하고, 시나리오 검증과정에서 도출되는 시나리오 정보 테이블을 통하여 UML 분석모델 간의 일관성을 검사하는 방법으로 수행된다.

본 논문에서 제안된 검증기법 연구의 중요한 의미를 열거하면 다음과 같다. 첫 번째로, UML 객체지향 분석과정에서 생성되는 Use Case와 시나리오를 이용하여 개발하고자 하는 시스템을 구성하는 객체들의 정적인 구조뿐만 아니라 전체적인 동적행위를 동시에 체계적으로 검증할 수 있는 방법을 제시하였다는 데에 가장 중요한 의미가 있다. 두 번째로, 사용자 요구분석 명세의 정형화 방법과 통합된 그래픽 표기방법을 제시하였다는 의미가 있다. 세 번째로, 사용자중심의 시스템 추상화 도구인 Use Case 및 시나리오를 검증 기법에 응용함으로써 사용자가 시나리오를 통하여 시스템 개발을 통해 구현하고자 하는 요구사항의 정확한 표현을 확인하기가 용이하고, 검증과정 중에 시나리오를 분석자와 사용자와의 의사소통의 도구로 이용함으로써 사용자 중심의 검증을 도모할 수가 있다는 데에 의미가 있다. 본 논문에서 연구된 검증기법과 연계하여 향후 진행되어야 할 연구 주제들에 대하여 논하면 다음과 같다. 우선, 제안된 검증기법의 전체적인 절차를 자동화된 검증도구(Verification Tool)로 구현함으로써, 기존의 객체지향 소프트웨어 개발도구와의 연동이 가능한 방향으로 연구를 진행할 필요성이 있다. 그리고, 제안된 시나리오기반 검증기법을 설계 및 구현단계까지 확장하여 적용하는 방안에 대한 연구가 진행되어야 한다.

참 고 문 헌

[1] Joanne M. Atlee and John Gannon, State-based Model Checking of Event-driven System Requirements, IEEE Trans. on Software Engineering, Vol.19, No.1, pp. 24-40, Jan. 1993.

[2] B.W. Boehm, Verifying and Validating Software Requirement Specification and Design Specification, IEEE Software, pp. 61-72, 1984.

[3] G. Booch, Object Oriented Design with applications, Benjamin/Cummings Publishing Company,1994.

[4] A.M. Davis, Software Requirement Analysis and Top-Down Software Development, Report HPL-91-21,Feb. 1991.

[5] Bruce P. Douglass, Real Time UML: Developing Efficient Objects for Embedded Systems, Addison-Wesley, 1998.

[6] Eriksson and Penker, Unified Modeling Language Toolkit, Wiley Computer Publishing,1998.

[7] Fiona Hayes, Derek Coleman, Coherent Models for Objected-Oriented Analysis, ACM OOPSLA 91Conference Proceedings, pp. 171-183, Oct. 1991.

[8] P. Hsia, Behavior based Acceptance Testing of Software Systems: A Formal Scenario Approach,IEEE Software, pp. 293-298, Nov. 1994.

[9] Ivar Jacobson, Object-Oriented Software Engineering: A Use Case Driven Approach, Addison-Wesley,1992.

[10] Shekhar Kirani and W.T. Tsai, Method Sequence Specification and Verification of Classes, JOOP,pp.28-37, Oct. 1994.

[11] Edward Kit, Software Testing in the Real World: Improving the Process, Addison-Wesley, 1995.

[12] Terry Quatrani, Visual Modeling with Rational Rose and UML, Addison-Wesley, 1998.

[13] Neil C. Rowe, Artificial Intelligence Through Prolog, Prentice-Hall, 1988.

[14] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy

and W. Lorensen, Object- Oriented Modeling and Design, Prentice-Hall, 1991.
[15] Klaus Weidenhaupt, Scenarios in System Development : Current Practice, IEEE Software, pp.34-45, Mar. 1998.
[16] Stephen Yau, D.H. Bae and K.H. Yeom, An Approach to Object-Oriented Requirements Verification in Software Development for Distributed Computing Systems, Proceedings of COMPSAC, pp.96-102, Nov. 1994.
[17] 김도형, 객체 모형화 기법에서 메시지행적과 상태도 간에 일관성 점검을 통한 요구사항 확인, 한국정보과학회 논문지, Vol. 23, No. 2, pp. 1527-1530, 1996.
[18] 이광용, 객체지향 분석모델의 완전성과 일관성 점검을 위한 진단모델의 설계, 숭실대학교 석사학위논문, 1992.
[19] 염근혁, 분산시스템을 위한 객체지향 소프트웨어 개발에서 요구명세에 대한 검증 방법, 한국정보과학회학술발표논문집, Vol. 23, No. 1, pp. 643-646, 1996.
[20] 백진욱, 배두환, 객체지향 요구명세의 동적행위 검증을 위한 방법, 한국정보과학회 학술발표논문집, Vol. 24, No. 1, pp. 567-570, 1997

조 진 형
1990년 서울대학교 컴퓨터공학과 공학사. 1999년 한국과학기술원(KAIST) 정보및통신공학과(컴퓨터공학전공) 공학석사. 2001년 서울대학교 대학원 기술경영협동과정(MIS전공) 박사과정수료. 1990년 ~ 1997년 현대전자 소프트웨어 연구소 선임연구원. 1999년 ~ 현재 동양공업전문대학 전산경영기술공학부 인터넷정보과 교수. 관심분야는 소프트웨어 공학, 정보보안관리, 경영정보시스템

배 두 환
1980년 서울대학교 조선공학과 학사. 1987년 위스콘신-밀워키대학 전산학 석사. 1992년 플로리다 대학 전산학 박사. 1992년 ~ 1994년 플로리다 대학 전산학과 조교수. 1995년 ~ 1996년 한국과학기술원 정보 및 통신공학과 조교수. 1996년 ~ 현재 한국과학기술원 전산학과 조교수.