객체지향 분석의 완전성과 일관성 검증을 위한 툴의설계

김 치 수[†] · 진 영 진[††]

요 약

소프트웨어 개발방법에서 객체지향 분석방법은 많이 있고 계속적으로 새로운 기법이 소개되고 있다. 그러나 기존의 객체지향 분석방법에서는 정확한 객체의 식별과 확인이 어렵고 소프트웨어의 문제를 데이터에 근거해서 초기에 분해하기 때문에 상위 레벨의 제어 측면을 소홀히 하는 경향이 있다. 그 결과 사용자가 요구하는 소프트웨어에 대한 부정확한 이해와 분석오류를 낳는다.

따라서 본 논문에서는 이러한 문제점에 인식을 갖고 소프트웨어의 분석 단계에서 사용자의 요구가 충분히 반영될 수 있도록 객체모델의 메서드와 STD의 트랜지션 사이에 상호참조를 통해 완전성과 일관성을 검증할 수 있는 TOVERC를 설계하였다.

A design of a tool to verify completeness and consistency of object-oriented analysis

Chi Su Kim[†] · Jin Young Jin[††]

ABSTRACT

Among the methods of developing software there are many object-oriented analysis(OOA) techniques, and the new ones are being introduced continuously.

The present OOA techniques, however, have difficulty in the identification and the verification of the objects and tend to ignore high-level control aspects of the problem due to the initial partitioning of them on the basis of the data. As a result, it brings inaccurate understanding and faults in the software which is required by users.

Therefore the purpose of this paper is to design the TOVERC to verify completeness and consistency through cross-reference between the state transition diagram and the methods of object model in order to reflect the requirements of users in the analysis of software considering this problem.

1. 서 론

소프트웨어 개발에 분석, 설계, 코딩, 테스팅의 4가지 단계가 일반적인 소프트웨어의 생명주기이다. 최근에는 이러한 단계에서 분석단계의 중요성이 대두되고 있는데 그 주된 이유는 분석가가 문제를 이해하는데에는 많은 시간이 소요됨에도 불구하고 문제를 정확하게 이해하기가 어렵고 이로 인하여 오역과 사용자의 요구를 만족하지 못하게 되는 결과가 올 수 있기 때문이다.

소프트웨어 개발단계에서 이러한 오류수정은 분석

† 정 회 원:공주대학교 전자계산학과
†† 준 회 원:충남 당진여고 교사
논문접수:1997년 2월 26일, 심사완료:1997년 8월 26일

단계가 구현단계 보다 100배 정도의 경제적 비용을 줄일 수 있을 뿐만 아니라[1] 오류가 없는 소프트웨어를 사용함으로써 사용자의 요구를 만족 시킬 수 있다[2].

본 논문에서는 기존의 객체지향 분석 방법에서 소홀이 하고 있는 상위레벨의 제어흐름 측면과 분석 단계에서의 사용자 역할에 중점을 두고 객체모델의 메서드와 STD의 트랜지션 사이에 상호참조를 통해 완전성과 일관성을 검증할 수 있는 객체지향 분석방법의 TOVERC(TOol for VERification of Completeness and consistency)를 설계하였다.

본 논문에서 설계한 TOVERC 분석 방법은 주어진 문제를 초기에 데이터와 제어흐름으로 분할하고 후에 다시 한 곳으로 모아서 검증과 확인을 할 수 있게 하였다.

이 논문의 구성은 다음과 같다. 2장에서는 관련연구를 소개하고 3장에서는 소프트웨어 개발방법인 TOVERC를 제시하였으며 4장에서는 실제의 예를 보여주고 5장에서는 향후 과제와 결론을 맺는다.

2. 관련연구

객체지향 분석·설계 방법으로 잘 알려진 Shaler & Mellor[3]의 방법은 정보 모델, 상태모델, 프로세스 모델개발의 3단계로 구성되어 있다. 이 방법은 정보 모델링을 기본으로 하며 분석을 중요시한다. 그러나 이 방법에서 메서드의 확인은 객체 안에서만 이루어지고 있다. 따라서 상위레벨의 프로세스는 객체들 사이에서 메서드의 결합에 의해서 실행되기 때문에 객체들 사이의 메서드를 확인하는 경우 제어흐름적인 측면이 모호해질 수 있다.

또한 Coad & Yourdon[4]의 방법은 E-R다이어그램을 사용하여 엔티티의 활동을 데이터 모델링하는 것을 기본 촛점으로 하고 있다. 이 기법은 스몰토크(Smalltalk)의 클래스 계층구조와 엔티티-관계를 결합시켰고 데이터와 메서드가 객체내부에 그룹화된 상태로 5단계의 계층으로 나타난다.

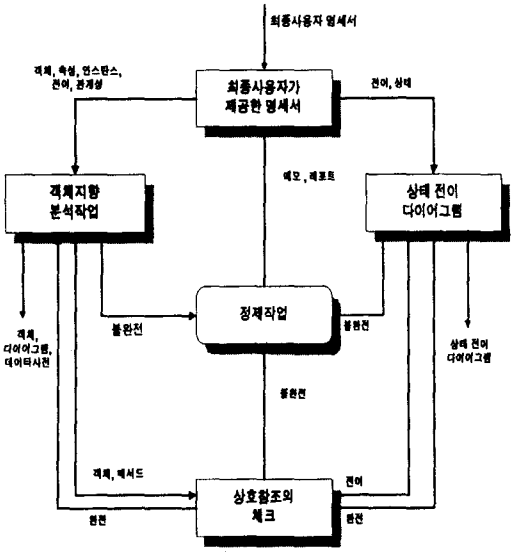
이 방법에서 제어흐름은 서비스 연결 부분에서 다루고 있으며, 객체안에 메서드를 확인하기 위하여 객체의 내용, 상태-사건-응답, 또는 STD를 사용한다. 그러나 이것은 분석결과가 아니다. 또한 데이터의 정적인 측면은 잘 표현하나 동적인 표현은 미흡한 단점을 가지고 있다.

본 논문에서 제시한 TOVERC방법은 위 방법들의 문제점을 해결하기 위해, 문제의 설명서에서 객체모델과 STD를 독립적으로 개발하며 상호 참조표를 통하여 빠뜨린 부분이나 중복된 부분의 일관성과 완전성을 체크할 수가 있다. 즉, 완전성은 개발 의뢰자가 요구하는 모든 것들이 포함되고 그 밖의 다른 사항들은 추가되지 않는 것을 의미한다.

일관성은 분석한 구성 요소들의 용어와 정의가 개발 의뢰자가 제공한 명세서의 의미에서 변하지 않고 있는 것을 말한다. 이러한 측면에서 일관성과 완결성은 개발 의뢰자가 제공한 것이 정확한지 아닌지를 확인할 수 없지만 개발 의뢰자가 제공한 서류를 분석하는 과정에서 얼마나 충실하게 이행하는지를 확인할 수 있다. 또한 내부적인 일관성과 완결성의 체크는 분석의 결과들이 불일치한 정보를 포함하지 않도록 해야 한다. 예를 들면 구성요소들의 이름이 유일해야 하고 결정되지 않은 사항들, 불완전한 부분들에 대해서도 일관성과 완결성의 체크가 이루어져야 한다.

3. TOVERC의 구성

본 논문에서 제시하는 TOVERC는 사용자가 제시



(그림 1) TOVERC의 구성도
(Fig. 1) Configration of TOVERC

하는 서류(개발의뢰와 관련된 모든 서류)를 중심으로 시작한다. (그림 1)에서 볼 수 있듯이 서류를 통하여 컴포넌트를 추출하고 문제를 이해하기 위해서 서류를 읽고 객체모델과 STD를 개발한다. 객체모델과 STD는 상호참조를 통하여 일관성과 완전성을 체크한다[5].

3.1 객체모델

객체모델은 문제의 전체적인 면을 나타내고 사용자가 검증하기 쉽게 이해할 수 있는 형태의 자료를 제시해 준다[6]. 본 논문의 표기법에서는 속성이 데이터를 나타내고 메서드가 변형을 표현한다. 또한 사용자가 이해하고 읽을 수 있도록 단순함을 유지한다[7].

3.1.1 객체

TOVERC 객체모델에서 객체는 속성과 메서드로 관련되어진다. (그림 2)에서 점선 아래는 메서드를 나타내고 위에는 속성을 나타낸다. 본 논문의 TOVERC 객체 표기법에서는 객체와 객체 클래스를 구별하지는 않지만 데이터 사전에서는 객체의 인스탄스의 수를 나타낸다. 만약 한개의 인스탄스가 있다면 객체로 모델화되고 여러개의 인스탄스가 있다면 객체 클래스로 모델화된다. 예를들어 (그림 2)에서 객체 MISD ivision은 병원이 한개의 MIS Division을 갖기 때문에 객체가 되고 다른 객체들은 여러개의 인스탄스를 갖기 때문에 객체클래스가 된다.

Embly, Kurtz 와 Woodfield[9]등의 모델에서 객체는 점으로 나타내고 객체 클래스는 사각형으로 표현하지만 TOVERC에서는 객체와 객체 클래스를 구별하지 않는다. 왜냐하면 객체와 객체클래스의 표기법이 세분화 되면 문제에 대하여 자세한 설명은 가능하지만 사용자가 검증이나 확인을 할 경우에는 혼돈이 일어날 수 있기 때문이다[8].

객체를 모델링하는데 있어서 TOVERC는 모델 중심 접근법을 사용한다[9]. 이 방법은 분석가가 미리 정의된 알고리즘보다 필요에 따라서 문제의 영역에서 모델링을 하도록 한다. 또한 분석가들은 속성이나 메서드 없이 객체를 추가할 수 있도록 한다.

3.1.2 속성

속성은 관련된 항목의 그룹 형태이거나 단순한 항목으로 이루어 진다. 예를들어 (그림 2)의 DateAdmitted, Physician.OfficeAddress, Person.Name 과 Person.Phone의 그룹속성은 사람의 신분확인과 관련된 항목들이며 Patient.Disposition과 Nurse.specialty는 서로 별개의 단순한 항목들이다.

모델을 개발하는 동안에 분석가는 객체에 속성을 자유롭게 추가할 수 있지만 모델이 완성되면 속성은 메서드를 통해서만 접근이 가능하다.

Coad[4]에서는 각 속성들에 대한 질의, 수정, 생성 메서드를 의미적으로 내포하고 있지만 본 논문의 TOVERC에서는 메서드에 대한 명확한 선언을 한다.

3.1.3 메서드

메서드는 속성값을 읽고 변경 시키는 기능을 가지고 있으며 외부 객체에게 사용할 수 있도록 해준다. 또한 (그림 2)의 객체 MISDivision처럼 속성 없이 메서드만 존재하는 것도 가능하다.

데이터 릴레이션쉽은 제약조건을 체크하는 메서드로 나타난다. 예를들어(그림 2)의속성 Chart.Admitting Physician을 평가할 때 의사가 병원에 근무 중인지를 체크해야한다. 이 경우 "Physician.PhysicianExist?" 메서드에 의해서 체크되어진다.

메서드는 또한 문제의 의미적 제약조건을 나타낸다. 예를들어 (그림 2)의 객체 Physician은 두개의 동일한 의미의 메서드(Physician.GetPhysicianInfoBYName 과 Physician.GetPhy-sicianInfoBYIDNumber)는 의사에 관한 정보를 얻는 점에서 같은 설명서(요구서)에는 중복되게 표현하지 않는다. 이처럼 거의 같은 의미의 메서드로 나타내어지는 것을 의미적 제약조건이라고 한다.

3.2 릴레이션쉽

본 논문에서는 객체들 사이의 관계를 나타내기 위하여 메시지 전달 릴레이션쉽, 데이터 릴레이션쉽, 상속 릴레이션쉽등의 3가지 관계성을 사용한다[10].

3.2.1 메시지 전달 릴레이션쉽

메시지전달 표기법은 (그림 2)에서와 같이 굵은 화살표를 사용한다. 예를들어 객체 MIS Division과 의사의 메시지전달 화살표는 MISDivision에서 두개의 메시지가 같은 화살표를 따른다. 이러한 메시지가 데이터 사전에서는 메서드와 트랜지션에의해서 나타난다.

메시지는 데이터, 제어흐름, 시간제약과 같은 서로 다른 응답조건을 갖는다. Coad[4]에서는 서로 다른 조건들을 표기하고 있지만 본 논문에서는 메시지 응답 조건모델링을 Booch의 객체지향 설계기법에서와 같이 설계 단계로 미룬다.

3.2.2 데이터 릴레이션쉽

데이터 릴레이션쉽은 객체 중에서 카디넬리티 제약조건을 보여준다. 카디넬리티는 객체B의 인스탄스에 관련된 객체A의 인스탄스의 수이다. 예를들어 (그림 2)에서 신청서와 챠트는 일대일 관계를 갖고, 환자와 챠트는 일대다의 관계를 갖는다. 그리고 챠트는 의사와 다대다의 관계를 유지한다.

TOVERC관계 표기법에서는 단일 및 이중 화살표들을 사용한다. 단일 화살표는 관계성의 한 목적지를 나타내고 이중 화살표는 관계성에서 여러 목적지를 나타낸다. 또한 객체들에 대한 선택적인 표기법(관계의 각 목적지로 최대와 최소)으로 나타낸다. 관계의 구체적인 제약조건은 최대값, 최소값의 쌍으로 표기된다.

3.2.3 상속 릴레이션쉽

상속성은 객체지향 언어가 일반화와 특수화의 관계를 표현하는데 사용된다. 상속성이 문제를 서술하는데 중요하고 분석과정에서 모델화되지만 TOVERC 표기법에서 상속계층구조는 분석단계보다 오히려 설계단계에서 더 많은 비중을 두고 있다[11].

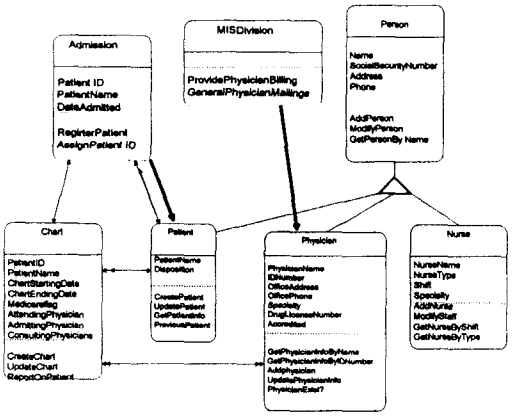
상속성을 위한 TOVERC표기법에서 삼각형의 위쪽 코너부분은 상위클래스를 나타내고 삼각형의 아래부분은 하위클래스를 나타낸다. (그림 2)에서 사람은 상위 클래스이고 환자나 간호원, 의사는 하위 클래스이다.

3.3 상태전이 다이어그램(State Transition Diagram)

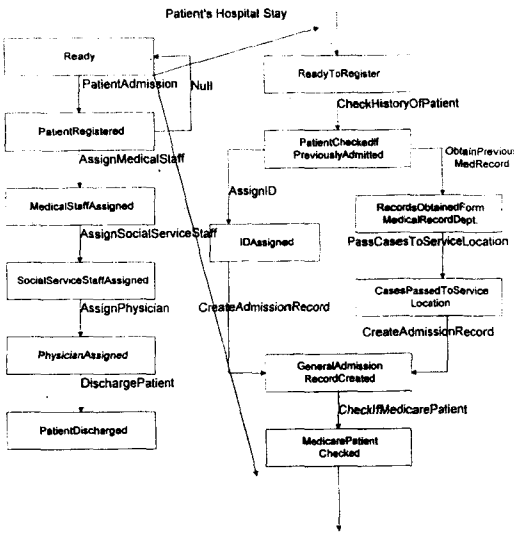
TOVERC에서 STD는 객체모델과는 독립적으로 개발되었기 때문에 전체적인 관점에서의 제어흐름의 변환과정을 다룬다[12].

STD는 상태와 트랜지션으로 구성되며 트랜지션은 더욱 세분화가 될 수 있고 STD의 세분화는 1차적인 객체모델을 더욱 자세히 알 수 있게 해준다. 또한 사용자가 STD를 읽는 동안 세분화된 여러모델을 통하여 한개의 객체모델 보다는 더욱 자세한 내용을 알 수 있고 STD는Petri nets와 Call graph[13]와 같은 검증과 확인기법을 위해 이용될 수 있다.

STD는 활동의 연속성을 다루지만 시간에 대한 제약조건을 다루지는 않는다. 예를들어 STD는 환자가 퇴원하기전에 등록을 해야 한다는 것을 보여주지만 환자등록이 30분 이내에 이루어지고 병원 기록부서



(그림 2) TOVERC의 표기법을 사용한 예
(Fig. 2) Example used by a notation of TOVERC



(그림 3) 환자 입원 과정의 STD
(Fig. 3) STD of the Patient Admission Process

의 오래된 차트를 정정하여야 한다는 제약조건을 다루지는 않는다. 시간의 제약조건은 성능에 관한 것이고 비기능적인 요구이다[14].

(그림 3)에서 상태는 박스로 나타내고 트랜지션은 화살표로 나타낸다. 또한 왼쪽의 상위레벨 STD(Patient's HospitalStay)는 하위레벨 STD(PatientAdmission) 트랜지션으로 세분화되고 분석단계의 결과인 구성요소들의 트랜지션과 상태가 다 사용되었을 때 세분화가 완성된다.

3.4 상호참조(Cross-Reference)

상호참조는 객체모델과 STD사이의 완전성과 일관성을 체크해준다. 객체모델은 메서드로써 트랜지션을 나타내고 STD는 상태사이에 트랜지션이 나타난다. STD의 각 트랜지션은 객체모델의 메서드로 나타낸다. 예를들어 PatientAdmission STD의 트랜지션은 객체모델의 어디에선가 틀림없이 발견된다.

트랜지션은 다른 객체모델 사이에서도 발견될 수 있다. 예를들어 (그림 2)의 객체모델에서 Admission.AssignPatientID와 Patient.PreviousPatient는 다른 객체들의 메서드이지만 PatientAdmission STD에서 둘다 사용되었다. 객체모델과 STD는 서로 분리되어 독립적으로 개발되기 때문에 객체모델의 메서드와 STD의 트랜지션은 같은 의미이면서 다른 이름을 가질 수 있다.

상호참조에 의해서 분석가는 객체 모델의 메서드와 STD의 트랜지션사이에서 각각의 의미를 확인할 수 있고 불일치하는 이름을 해결할 수 있다. 또한 Patient Admission의 트랜지션은 객체모델안에 나타나지 않는다. 그것은 상위레벨의 내부객체 트랜지션이다. 상위레벨의 트랜지션은 명세서의 일부분으로 간주하여 디자인 단계로 연결시킨다.

본 논문에서는 두개의 독립적인 모델링을 통하여 TOVERC가 더욱 정확한 분석을 할 수 있음을 제시하고 있다. 즉 상호참조에 의해서 빼놓은 객체, 트랜지션과 메서드를 찾을 수 있다. 또한 중복된 객체들, 트랜지션과 메서드도 발견할 수 있다. (그림 4)는 상호참조의 예를 보여주고 있다. 즉, (그림 4)에서 'X'표시는 트랜지션과 메서드가 같은 의미로 사용되는 것을 의미하고 'X'의 표시가 1개도 없는 부분은 해당항목이 빠져있다는 것을 알 수 있다. 또한 1개의 항목에 'X'의 표시가 2개 이상인 경우는 중복이 되었다는 것을 알 수 있다.

ObjectMethod / Transition	Admission.RegisterPatient	Admission.AssignPatientID	Chart.CreateChart	Chart.UpdateChart	Chart.ReportOnPatient	Patient.CreatePatient	Patient.UpdatePatient	Patient.GetPatientInfo	Patient.PreviousPatient?
CheckHistoryOfPatient									X
AssignID		X							X
CreateAdmissionRecord	X		X			X			
ObtainMed.Record								X	
PassCaseToServiceLocation									
CheckIfMedicarePatient				X					

(그림 4) 상호참조 표
(Fig. 4) Cross-Reference Table

3.5 데이터 사전

본 논문의 데이터 사전은 객체모델, 객체모델에서의 릴레이션쉽, STD로 구성된다. 데이터 사전의 정보는 완전성과 일관성을 체크하기 위해 일정한 형식의 데이터 구조를 갖는다. 따라서 정형화된 명세서를 위한 자료로 이용될 수 있다[15].

3.5.1 데이터 사전-객체

항 목	설명 / 예
Object Name	유일해야 한다.
ObjectDescription	객체의 설명부
Number of Instance	예상되는 객체의 최대수
Object creation rate	객체의 생성빈도 (number/week, number/day)
Reference to source	사용자가 제공한 서류의 참조위치 (chapter, paragraph)

3.5.2 데이터사전-속성

항 목	설명 / 예
Attribute Name	유일해야 한다.

Existence type	필수적/선택적
Uniqueness	유일/비유일
Attribute Description	속성의 설명부
Data type	date,char,string
Value Range	합당한 값의 범위
Reference to source document	서류의 참조 위치(Inpatient admitting,paragrph)

3.5.3 데이터사전-메서드

항 목	설명 / 예
Method Name	주로 사용자의 용어를 사용한다.
Method Description	메서드의 설명부
Attributes used	메서드가 사용한 속성

3.5.4 데이터사전- 릴레이션쉽

항 목	설명 / 예
First object name	첫번째 객체이름
Second object name	두번째 객체이름
Type of Relationship	관계의 유형(Data, Char, String)
Dependency	의존 / 비의존
Cardinality	최대값, 최소값
Specific use information	객체사이의 처리과정의 설명부분
Relationship Description	객체사이의 설명부분
Reference	Document, pgh6

3.5.5 데이터사전-STD(상태)

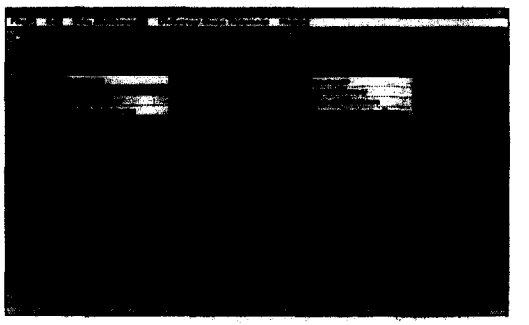
항 목	설명 / 예
State Name	사용자의 사용법 따름, 부모 STD를 따름
State Description	설명부
Reference	Document, pgh4

3.5.6 데이터사전-STD(트랜지션)

항 목	설명 / 예
Transition Name	유일해야 한다.
Transition Called	상위 트랜지션에서 세분화된 이름
Transition Description	설명부분
Reference	참조위치

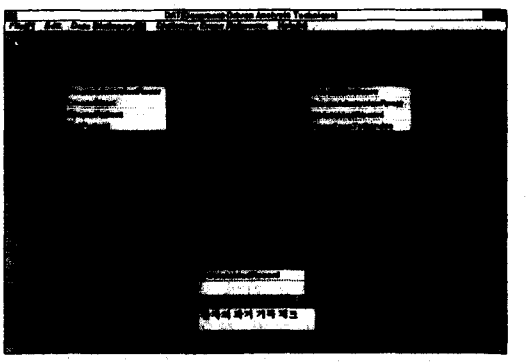
4. 실행 예

개발하고자 하는 문제에서 객체모델과 STD 사이에 자동적인 상호참조가 이루어질 수 있도록 객체모델의 메서드와 STD의 트랜지션에 대한 속성을 정의하고 데이터베이스화하여 완전성과 일관성을 체크한다. 다음의 그림은 상호참조의 실행예를 보인 것이다.



(그림 5) 메서드와 트랜지션의 선택
(Fig. 5) The selection of a method and a transition

위 (그림 5)에서 보듯이 왼쪽의 메서드와 오른쪽 트랜지션사이의 상호 참조가 이루어지는 과정으로 왼쪽 메서드의 항목을 크릭하면 왼쪽의 메서드와 동일한 이름의 트랜지션이 오른쪽에 나타난다. 만약 같은 이름의 항목이 나타나지 않으면 메서드와 동일한 이름의 트랜지션은 존재하지 않는다는 것을 의미한다. 또한 분석가는 메서드와 트랜지션의 항목을 비교하



(그림 6) 상호참조 체크 결과
(Fig. 6) A result of cross-reference checking

여 같은 의미이면서 다른 이름이 있는 경우 이를 수정할 수 있고 이와 같은 사항도 없으면 해당 항목이 빠진 것을 식별할 수 있다.

(그림 6)은 메서드와 트랜지션의 상호 참조의 체크로 메서드와 트랜지션의 이름이 동일하지는 않지만 같은 의미임을 알 수 있고 또한 메서드의 이름을 변경할 수 있는 예를 보여주고 있다. 이와 같은 방법으로 객체모델의 메서드와 STD의 트랜지션을 상호 참조하여 일관성과 완결성을 체크할 수 가 있다. 또한 메서드와 트랜지션이 독립적으로 개발되기 때문에 불일치 되는 이름과 의미를 확인 할 수 있으며 상위레벨의 트랜지션은 명세서의 일부분으로 설계단계로 연결시킬 수 있다.

5. 결 론

기존의 객체지향분석이 상위레벨에서의 제어흐름 측면을 소홀이 하고 확인하기가 어려운 문제점을 가지고 있다. 본 논문에서는 기존의 문제점을 해결하기 위한 TOVERC를 설계 하였다. 이툴은 사용자가 제공한 서류에 기반을 두고 분석가들은 컴포넌트 리스트를 나열한다. 컴포넌트 리스트는 서류와 분석 결과 사이에서 색인(기준)과 같은 역활을 한다. 또한 객체모델과 STD 모델이 독립적으로 생성되고 STD는 문제에 대한 전체적 상위레벨의 관점에서 제어흐름 측면을 나타낸다. 객체모델의 메서드와 STD의 트랜지션 사이에 상호 참조를 통하여 사용자의 질문, 전문가의 면담, 오류의 발견, 서류의 수정 등이 이루어진다. 또한 이방법은 사용자가 복잡한 연습과정 없이도 분석결과를 이해할 수 있도록 사용자가 이용하고 있는 용어를 따르도록 하는 단순성을 제공하며 데이터 사전에 의한 문법적, 어의적인 완전성과 일관성을 체크하여 주는 툴을 설계하였다.

향후 연구 과제로는 문세의 분석단계에서 제일 어려움을 겪는 객체 식별을 최종사용자와 분석가 사이에서 자동화된 객체 식별 기법 개발의 연구가 진행되어야 할 것이다.

참 고 문 헌

[1] A.M Davis, "Software Requirements Analysis and Topdown Software Development," Report HPL-91-21, Feb. 1991.

[2] S.C. Bailin, "An Object-Oriented Requirements Specification Method," Comm. of the ACM, Vol.32, No.5, May 1989, pp.608-623.

[3] S.Shlaer and S.J.Mellor, Object-Oriented Systems Analysis, Yourdon Press, Englewood Cliffs, NJ, 1988.

[4] P.Coad and E,Yourdon, Object-Oriented Analysis, Second Edition. Yourdon Press, Prentice Hall Englewood Cliffs NJ, 1991.

[5] A. Kawaguchi, H.Motoda, and R.Mizoguchi, "Interview-Based Knowledge Acquisition Using Dynamic Analysis," IEEE Expert, Vol. 6, Oct. 1991, pp. 47-60.

[6] E. Colbert, "The Object-Oriented Software Development Method:A Practical Approach to Object-Oriented Development," TRI-Ada 89 Proceedings, Oct. 1989.

[7] D. de Champeaux, "A Comparative Study of Object-Oriented Analysis Methods," Journal of Object-Oriented Programming, Vol.5, No.1, 1992. pp. 21-33.

[8] J.C.S.P. Leite, and P.A. Freeman, "Requirements Validation Through Viewpoint Resolution," IEEE Trans. on Software Engineering, 1993.

[9] D.W. Embly, B.D.Kurtz, and S.N. Woodfield, Object-Oriented Systems Analysis, Model-Driven Approach, Yourdon Press, Englewood Cliffs, NJ, 1992.

[10] P. Chen, "The Entity-Relationship Model-Toward a Unified View of Data," ACM Trans. on Database Systems, Vol.1, No.1, March 1976.

[11] A.Snyder, "Inheritance and the Development of Encapsulated Software Components," in Research Directions in Object-Oriented Programming, edited by b. Shriver and P. Wegner, MIT Press Cambridge, MA, 1987,PP. 411.

[12] G. Booch, Object Oriented Design with Applications, Benjamin/Cummings, Redwood City, CA, 1991.

[13] N.D. Birrell and M.A.Ould, A practical Handbook for Software Development, Cambridge University press, New York, NY, 1985.

[14] C.V.Ramamoorthy and H.So, "Software Requirements and Specifications," UCB/ERL M78/44, Electronic Research Laboratory, University of California, Berkeley, CA, 1978.

[15] B.W. Boehm, "A Spiral Model of Software Development and Enhancement," IEEE Computer, Vol.21, No.5 May 1988, pp.61-72.

김 치 수

1984년 중앙대학교 전자계산학과(학사)
1986년 중앙대학교 전자계산학과(석사)
1990년 중앙대학교 전자계산학과(박사)
1990년~1992년 8월 공주교육대학교 전임강사
1992년 9월~현재 공주대학교 전자계산학과 부교수
관심분야:소프트웨어공학, 객체지향분석·설계

진 영 진

1988년 공주대학교 상업교육과(학사)
1997년 공주대학교 전자계산학과(석사)
1993년~현재 충남 당진여고 교사
관심분야:소프트웨어공학, 객체지향분석·설계