객체지향 산출물간의 일관성 유지를 위한 효율적인 역 추적 기법

한만집, 장치원, 박신영, 라현정, 김수동
숭실대학교 대학원 컴퓨터학과
{mjhan, cwchang, sypark, hjla}@object.ssu.ac.kr, sdkim@comp.ssu.ac.kr

An Effective Backtracking Technique for Maintaining Consistency among Object-Oriented Artifacts

Man Jib Han, Chee Won Chang, Shin Young Park, Hyun Jung La, Soo Dong Kim
Dept. of Computer Science, Soongsil University

요 약

소프트웨어 개발 과정의 여러 산출물 간의 일관성 유지는 최종 소프트웨어의 품질을 결정하는 중요한 요소가 되며, 운영단계의 유지보수 효율성에도 큰 영향을 미친다. 특히 구현 등 개발 후반부 활동을 진행 시 발견된 하나의 오류는 그 이전 단계의 다수의 산출물의 오류로 인한 파급 결과이므로, 발견된 오류의 원인(Cause)가 되는 이전 단계의 오류를 효과적으로 추적할 수 있는 체계가 요구된다. 본 논문에서는 한 산출물의 오류가 다른 어떤 산출물의 어떤 세부항목들의 어떤 오류로 인한 파급 결과 인지를 정의한 역 추적(Backtracking) 그래프를 제안하여, 개발자가 오류를 발견 시 관련 산출물과 항목을 신속히 파악할 수 있도록 한다. 이를 이용하여 산출물간의 일관성 유지 노력과 시간을 최소화하고 일관성 유지의 정확성을 높일 수 있다.

1. 서론

소프트웨어 개발 시 생성되는 산출물들은 서로 관련되어 있기 때문에 특정 산출물의 오류 수정 시 다른 산출물이 변경되는 파급 효과를 가져 온다. 이 경우 여러 산출물간의 일관성을 유지하기 위해서는 많은 시간과 비용이 소비된다. 특히 개발 후반부에 있는 산출물에서 오류가 발생하는 경우에는 여러 산출물을 수정해야 하는데, 이 경우 개발자들은 각 단계의 일관성을 유지하는데 어려움이 많다.

한 산출물 수정이 다른 산출물에 영향을 주는 정도는 변경되는 내용에 따라 차이가 있다. 본 논문에서는 각 산출물간의 파급 결과 인지를 정의한 역 추적 그래프를 제안하여, 개발자가 오류를 발견했을 경우 수정해야 할 산출물들과 항목들을 효율적으로 파악할 수 있게 하고자 한다. 뿐만 아니라 산출물간의 일관성을 유지하기 위한 노력과 비용을 최소화 하면서도 산출물간의 정확성은 높일 수 있는 방법을 제시하고자 한다.

본 논문의 구성은 다음과 같다. 2장의 기반연구에서는 본 논문을 작성하게 된 계기를 설명하고, 3장에서는 추적 대상 항목을 제시 하고자 한다. 4장에서는 3장에서 나열한 추적 대상 항목을 기반으로 개발 중 오류가 발생했을 때 다른 산출물의 변경을 요구하는 항목의 역 추적 그래프를 제시한다. 마지막으로 5장에서는 결론 및 향후 연구과제를 제시한다.

2. 관련 연구

객체지향 방법론의 OOSAD(Object-Oriented System Analysis and Design)을 이용하여 소프트웨어를 개발하는 사이클은 다음과 같은 같다.[1] 프로젝트 관리 및 계획단계, 시스템 분석단계, 시스템 구현단계 이다. 프로젝트 관리 및 계획단계에서는 새로운 시스템을 개발하기 위해 개발 범위, 시간과 시스템이 실행하기 위한 자원 등을 결정한다. 시스템의 분석단계에서는 요구사항에 명시된 것을 분석하여 비용 및 기술 수준을 확인하여 개발 프로세스에 언급한다. 시스템의 설계단계에서는 분석단계의 산출물을 이용하여 구현에 필요한 상세 로직과 물리적 시스템을 명세 한다. 마지막으로 구현단계에서는 각 단계에서 나온 산출물을 이용하여 코딩, 테스트 및 프로그램의 설치까지를 한다.

추적성은 한 단계의 결과가 다른 단계에 있는 결과로 명확한 규칙에 의해 자동 맵핑 될 때 가능해진다[2]. 소프트웨어 개발 산출물간에 추적이 가능해지면 소프트웨어는 일관성을 유지하는 것이 가능해 질 것이다. 하지만 기존 연구에서는 추적에 관한 필요성만 지적할 뿐 구체적으로 추적이 필요한 항목이나 항목간의 연관관계를 보여주지는 못했다.

일관성(Consistency)이란 시스템과 컴포넌트의 부분이나 문서간에 균일성(Uniformity)이나 표준성(Standardization)을 이루고 있고, 문서간 모순성이 없는 것을 의미한다. 시스템을 다양한 관점에서 보게 되면 하나의 시스템이 각 산출물에서 다르게 표현되어 일관성 유지하기 힘들다. 그리고 소프트웨어 산출물은 안정적이지 않고 요구된 정보에서 다른 대안을 반영하거나 시스템 개발과정에서 변경될 수 있다[3]. 개발 과정 산출물간 일관성을 유지하지 못한다면 개발 후반부 과정에서 산출물이 수정 됐을 경우 그와 관련된 산출물을 모두 수정하는 과정에 어려움이 생긴다. 특히 개발완성 된 소프트웨어를 향상시킬 때는 산출물간의 일관성 유지 개념이 더욱더 필요하다[4].

3. 객체지향 개발 산출물 및 항목

본 절에서는 객체지향 개발 시 중요한 산출물을 나열하고, 각 산출물에 대하여 일관성 유지를 위하여 추적해야 할 항목들을 정의한다.

3.1. 객체지향 개발 산출물

RUP, OMT등 객체지향 프로세스나 방법론에 따라 사용되는 산출물의 종류와 이름이 달리 사용되므로, 본 절에서는 객체지향 과제 시 공통적이며 중요한 산출물들을 가능한 범용적이거나 표준적 용어를 사용하여 나열한다. 계획단계의 산출물로 과제 계획서와 요구사항명세서(SRS)가 있으며, SRS는 기능적 요구사항과 비기능적 요구사항으로 구성된다.

분석단계의 산출물로는 Use Case 모델, 개념적 클래스 다이어그램(Class Diagram, CD), 개념적 순차도(Sequence Diagram, SD)가 있다. Use Case 모델은 SRS를 기반으로 액터와 고객이 원하는 기능을 식별하여 도식화한 Use Case 다이어그램과 각 Use Case에 대한 명세로 구성된다. 개념적 CD는 클래스와 클래스간의 관계를 표현하며, 개념적 SD는 Use Case 명세를 기반으로 객체 간에 메시지 호출 순서를 나타낸다.

설계단계의 산출물로는 사용자 인터페이스 설계(User Interface, UI), 상세 CD와 상세 SD가 있다. UI는 사용자가 사용할 환경은 문서화한 것이고, 상세 CD는 개념적 CD를 기반으로 프로그래밍 언어, DBMS 등 개발 환경에 종속적인 설계이다. 상세 SD는 개념적 SD와 상세 CD를 기반으로 객체 메시지 교환을 지원하는 EJB등의 미들웨어와 기타 개발 환경에 종속적인 메시지 교류를 정의한다.

구현단계의 산출물로는 UI를 구현한 클라이언트 프로그램 코드 (Client Program), 서버 영역에 사용되는 클래스를 구현한 코드 (Server Program) 및 데이터베이스로 구성된다. 시험단계 산출물로는 프로그램의 시험 결과를 포함하는 시험 보고서와 최종 산출물인 어플리케이션이 있다.

3.2. 산출물의 추적 대상 항목

본 절에서는 각 산출물별 추적 대상 항목을 정의한다. 표 1은 단계별, 산출물이 포함하고 있는 구성요소 중에서, 추적 대상 항목들만 나타내고 있다.

표 1. 산출물 별 추적 대상 항목 표

단계	산출물	추적대상 항목
계획단계	SRS	- 기능적 요구사항
	Use Case 다이어그램	- 액터 - Use Case, Use Case간의 관계
	Use Case 명세	- Flow - 전제 조건
분석단계	개념적 CD	- 클래스, 속성, 메소드 시그네쳐 - 상속, 포함, Association 관계
	개념적 SD	- 메시지 이름, Object - Object
설계단계	UI	- 네비게이션 트리 - UI 컴포넌트
	상세 CD	- 클래스, 속성, 메소드 시그네쳐 - 상속, 포함, Association 관계
	상세 SD	- 메시지, Object - 객체간 메시지 순서
구현단계	코드	- 클래스 - 상수, 변수, 자료구조 - 메소드의 시그네쳐, 코드 - 상속, 포함 관계
시험단계	테스트	- 버그

4. 산출물간 역 추적 그래프

본 절에서는 3.2에서 나타낸 표1을 바탕으로 역 추적 그래프를 나타낸다.

역 추적 그래프는 어떠한 하위 추적 대상 항목의 오류가 상위 추적 대상 항목의 오류인지를 쉽게 확인하고, 수정 시 개발자가 효율적으로 산출물간의 일관성을 유지하기 위해 제안한 그래프로써 그림 1과 같이 표현될 수 있다.

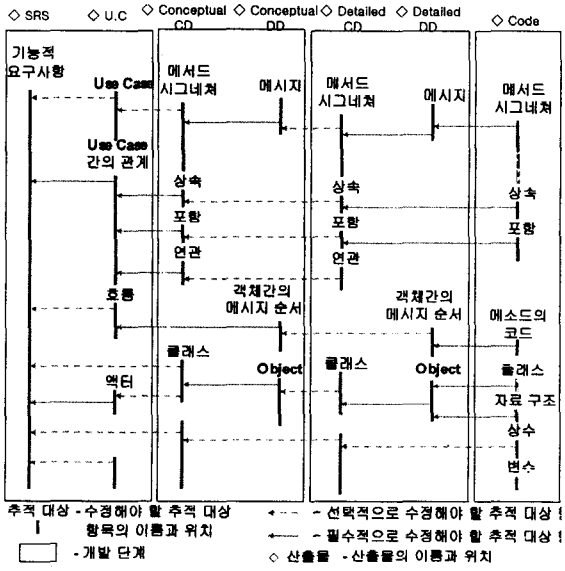

그림 1 역 추적 그래프

역 추적을 하기 위해 수정된 추적 대상 항목을 찾고 그 항목과 연결되어 있는 추적 대상 항목을 찾는다. 만약 찾은 항목간의 관계가 실선 화살표로 되어있는 경우 필수적으로 찾은 항목을 수정해야 하고 점선 화살표인 경우 선택적으로 수정하게 된다.

코드단계의 메서드 시그네쳐가 변경된 경우 상세 DD의 메시지를 사용한 것이므로 같이 변경한다. 그리고 상세 DD의 메시지는 상세 CD의 메서드 시그네쳐를 사용하므로 같이 수정이 되고, 상세 CD의 메서드 시그네쳐가 개념적 DD에서 메시지로 사용되지 않을 경우 역 추적이 끝난다. 그러나 메시지가 사용되는 경우 개념적 CD의 메서드 시그네쳐가 사용되므로 개념적 CD도 수정이 되고, 이 메서드 시그네쳐가 Use Case의 기능적인 이름이 변경되지 않은 경우 여기서 역 추적이 끝나고 기능적인 이름이 변경된 경우 Use Case도 수정된다. 마지막으로 Use Case가 기능적 요구사항과 틀리게 되거나 빠진 경우 기능적 요구사항까지 수정하고 아닌 경우 Use Case에서 역 추적이 끝난다.

또한 상속, 포함, 연관에서의 관계에서 코드가 변경 되었을 경우 상세 CD의 관계가 틀려지므로 수정하고 상세 CD의 관계가 개념적 CD의 관계에 사용되지 않은 경우 역 추적이 끝나고 관계가 사용된 경우 개념적 CD도 수정된다. 개념적 CD의 관계는 Use Case 간의 관계도 변해야 하므로 Use Case간의 관계도 수정한다. 마지막으로 Use Case간의 관계가 변경되면 기능적 요구사항과 틀려지므로 기능적 요구사항도 수정된다.

코드 단계의 메서드의 코드가 변경된 경우 상세 DD의 객체간의 메시지 순서가 변한다. 상세 DD의 변경된 객체간의 메시지 순서가 개념적 DD의 순서에 영향을 주지 않은 경우 역 추적이 끝나고, 영향을 주는 경우 개념적 DD의 객체간의 메시지 순서도 변한다. 개념적 DD의 객체간의 메시지 순서가 변하게 되면 Use Case 모델의 흐름에도 영향을 주므로 같이 변한다. 마지막으로 Use Case 모델의 흐름이 기능적인 요구사항에 명시된 경우 기능적 요구사항을 수정하고, 명시되지 않은 경우 역 추적이 끝난다.

코드 단계의 클래스 이름이나 자료구조 이름이 변할 경우 상세 DD의 오브젝트 이름도 변한다. 상세 DD의 오브젝트 이름이 변하면 상세 CD의 클래스 이름도 변한다. 상세 CD의 클래스 이름이 개념적 DD의 오브젝트 이름에 사용되지 않는 경우 역 추적이 끝나고 오브젝트 이름이 사용되는 경우 개념적 CD의 클래스 이름도 변한다. 개념적 CD의 클래스 이름이 기능적 요구사항에 명시된 경우 기능적 요구사항을 수정하고 명시되지 않은 경우 역 추적이 끝난다.

표 2에서는 역 추적 그래프에서 사용되는 표식 중 점선 화살표에 의하여 역 추적되는 경우를 설명한다.

표 2의 '현재단계'는 현재 오류가 발생된 산출물의 항목을 나타내며 '이전단계'는 추적이 필요한 항목을 나타낸다. '현재단계'의 [] 표시는 or를 대신한다.

표 2 선택적으로 수정해야 할 추적 대상 항목

현재 단계	이전 단계	추적 대상항목을 변경할 경우
[Use Case, 흐름,액터,전제조건,개념적 CD(클래스),개념적 CD(속성)]	기능적 요구사항	SRS에 명시되어야 하는 중요한 요소가 빠져있거나, 산출물의 변경된 요소가 SRS에 명시된 경우
개념적 CD (메소드 시그네처)	U.C (Use Case)	메소드의 기능에 따른 이름이 바뀌어 Use Case명이 변경되어야 할 경우
상세 CD(메소드 시그네처)	개념적 DD (메시지)	상세 CD와 개념적 DD에 동일하게 있는 메소드 이름이 변경될 경우
상세 CD(관계) [상속,포함,연관]	개념적 DD (관계)[상속,포함,연관]	상세 CD와 개념적 DD에 동일하게 있는 관계가 변경될 경우
상세 CD(속성)	개념적 CD(속성)	상세 CD와 개념적 DD에 동일하게 있는 속성이 변경될 경우
상세 CD (클래스)	개념적 DD(Object)	상세 CD의 클래스 이름과 개념적 DD에서의 Object명이 동일할 경우
상세 DD(객체간의 메시지 순서)	개념적 DD (객체간의 메시지 순서)	상세 DD와 개념적 DD 둘 다 명시된 메시지의 흐름이 변경인 경우
코드(상수)	상세 CD (속성)	코드와 상세 CD에 모두 명시된 상수가 변경된 경우

표 2에 대한 예로 우리가 하나의 도서관리 시스템을 구현하였다고 가정하자. 구현이 이루어져 있는 코드상에서 엔티티 클래스 내에 속하여 있는 checkOut(String barcode,String memberID) 메소드가 checkOutItem(String barcode,String memberID) 메소드로 이름이 변경 되었다. 산출물간의 일관성을 유지하기 위해 다음과 같이 수정할 수 있다. 메소드의 이름이 변경될 경우 상세 DD의 메시지와 상세 OM의 메소드 이름이 반드시 변경 된다. 설계단계의 메소드 이름과 분석단계의 메소드 이름이 동일하므로 개념적 DD의 메시지 와 개념적 OM의 메소드 이름을 수정 한다. 메소드의 기능에 따른 이름이 바뀌지 않았으므로 Use Case명이 변경될 필요 없이 역 추적이 종료 된다. 위 예와 같이 메소드 시그네처가 수정될 경우 역 추적 그래프를 사용 하여 효율적으로 산출물간의 일관성을 유지할 수 있다.

5. 결론 및 향후 연구과제

본 논문에서는 산출물간의 파급 결과를 정의한 역 추적 그래프를 제안하여, 산출물의 오류 발견 시 관련되어 변경 되는 산출물과 항목을 신속히 파악 하고자 했다. 지금까지 여러 연구를 통해 산출물간의 일관성에 관해서 논의한바 있으나 본 논문에서 제시한 역 추적 그래프를 사용하면 한 산출물의 특정 부분 수정 시 관련 있는 다른 산출물을 쉽게 찾을 수 있다. 또한 효율적으로 객체지향 산출물간의 일관성을 유지할 수 있을 뿐만 아니라 일관성 유지 노력과 시간을 최소화하고 정확성도 높일 수 있을 것으로 기대한다.

앞으로는 이 기법을 통해 산출물간의 일관성을 유지 하는 과정에서 더욱더 상세하고 규칙화 되며, 좀더 높은 수준의 일관성을 유지할 수 있게 해야 할 것이다. 그러기 위해서 선택적으로 역 추적이 이루어지는 추적대상 항목을 더욱 구체화해야 할 것이다.

마지막으로 객체지향 산출물간의 일관성 유지를 역 추적 그래프를 통해서가 아니라 자동적으로 추적할 수 있는 자동화 도구나 소프트웨어 개발이 필요할 것이다.

6. 참고문헌

[1] George, J., Batra, D., Joseph S. Valacich, Jeffrey A. Hoffer, *Object-Oriented Systems Analysis and Design* , Prentice Hall 2004.

[2] Bari, M., Gabrini, P., Rolland, C.,Zeroual, K., Active Information Systems,From Object-Oriented Design to Ada 95, Proceedings of the conference on TRI-Ada '96: disciplined software development with Ada, 1996

[3] Kuzniarz L., Reggio G., Sourrouille J. L., Huzar Z., Workshop on Consistency Problems in UML-based software development, Workshop Materials, Research Report 2002:06, Blekinge Institute of Technology, Ronneby 2002

[4] K. Narayanaswamy., Neil Goldman., "Lazy" consistency: a basis for cooperative software development, Proceedings of the ACM conference on Computer-supported cooperative work table 1992