제22회 한국정보처리학회 추계학술발표대회 논문집 제11권 제2호 (2004. 11)

요구사항명세와 테스트케이스 간의 일관성 유지 기법

박상훈, 최진명, 류성열

숭실대학교 컴퓨터공학과
e-mail:se_hoon@selab.ssu.ac.kr

A Techniques of Consistency Preservation between Requirements Specifications and Testcase

Sang-Hoon Park, Jin-Myung Choi, Sung-Yul Rhew
Dept of Computer at Graduate School, Soong-sil University

요 약

고품질의 소프트웨어 시스템을 개발하기 위해서 소프트웨어 테스트는 중요하다. 소프트웨어 개발 시에 발견되는 심각한 결함과 오류들은 소프트웨어 시스템이 사용자의 요구사항을 정확히 충족시키지 못하는데 원인이 있다. 테스트 수행 시에 요구사항을 정확히 반영하지 못한다면 반드시 테스트되어야 할 조건들이 누락될 수 있고 테스트의 성취도는 감소한다. 따라서 모든 소프트웨어 테스트는 사용자의 요구사항을 추적할 수 있어야 한다. 이를 위해 본 논문에서는 객체지향 접근방법을 사용하여 사용자의 요구사항을 테스트에 반영하고 추적하기 위한 일관성 유지 기법을 제안한다. 이를 기반으로 요구사항을 만족하는 일관성 있는 테스트케이스를 생성한다.

1. 서론

고품질의 소프트웨어 시스템을 개발하기 위해서 소프트웨어 테스트는 중요하다. 소프트웨어를 개발하는 과정에서 테스트는 시간과 비용이 가장 많이 드는 작업이다. 소요 기간의 50% 이상, 소요 비용의 50% 이상을 테스트에 사용하는 높은 품질을 요구하는 소프트웨어 프로젝트도 많이 찾아 볼 수 있으며 테스트의 비중은 점점 높아지고 있고 테스트에 소요되는 노력도 증가하고 있다[1].

소프트웨어 테스트의 목적은 오류를 일찍 발견하고 예방하기 위함이다. 대부분의 소프트웨어에서 발견되는 심각한 결함과 오류들은 소프트웨어 시스템이 사용자의 요구사항을 정확히 충족시키지 못하는데 원인이 있다. 그러므로 모든 소프트웨어 테스트는 사용자의 요구사항을 추적할 수 있어야 한다[2]

테스트를 실행하는 과정에서 부딪히게 되는 많은 문제들은 테스트케이스 설계의 미흡함에서 발생한다. 현재에도 실제 소프트웨어의 문제점들을 찾아내기 위한 다양한 테스트케이스 설계 기법이 연구되고 있으나 미흡한 실정이다. 요구사항의 정확한 반영이나 테스트기술 없이 직관만으로 테스트의 대상을 분석한다면 너무 많은 시간과 과다한 인력을 낭비하게 된다. 반드시 테스트되어야 하는 조건들이 누락될 수 있고 시스템의 복잡도로 인하여 테스트케이스는 많아지고 따라서 제한된 시간과 인력으로 테스트 성취도는 감소하게 된다. 따라서 테스트케이스 설계는 테스트 수행 전에 이루어져야 하며 이상적으로는 시스템 개발 초기에 작성되어 요구사항을 기반으로 시스템 설계과정 시에 생성되는 산출물도 테스트케이스 설계에 반영되어야 한다.

시스템 개발에서의 주된 문제점은 부정확하고 불완전한 요구사항이다. 이러한 요구사항은 비용의 증가, 납기 지연 등으로 결국 고객의 불만족으로 이어진다. 이것은 요구사항 도출 활동인 고객의 요구사항 수집, 분석, 관리, 검증 등의 활동이 제대로 이루어지지 않아 발생된다.

테스트 작업의 유형과 프로세스는 소프트웨어 개발 단계와 밀접히 관련되어 있다. 테스트 작업이 프로그램에 포함된 단순한 코딩의 오류만을 찾는 작업이 아니라 요구 분석에서의 오류, 설계 또는 모듈 인터페이스의 오류 등 개발단계의 작업들에 대한 테스트를 포함하므로 개발 프로세스와 매핑시킬 필요가 있다[1].

본 논문에서는 소프트웨어 개발단계에서 생성되는 산출물을 기반으로 요구사항과 테스트케이스 간의 일관성을 유지하기 위한 기법을 제시한다. 구성은 2장에서 관련연구로서 테스트 프로세스와 요구사항 기반의 테스트 방법, 명세기반 테스트와 코드기반 테스트에 대해 기술한다. 3장은 요구사항과 테스트 케이스 간의 추적 기법을 제안하고 4장에서는 추적성의 이점을 살펴본다. 마지막으로 결론 및 향후연구로 구성된다.

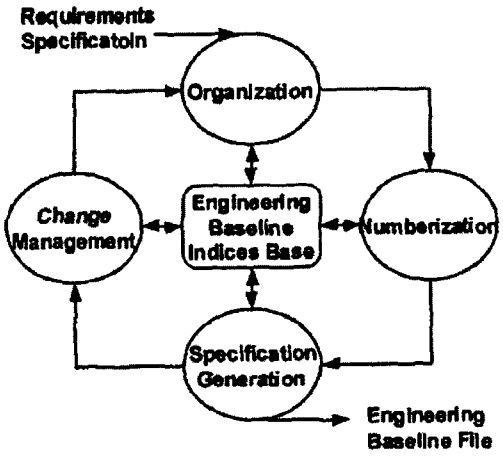
2. 관련 연구

2.1 객체지향 방법론을 이용한 소프트웨어 개발

객체지향이란 객체를 중심으로 모든 개발 활동이 수행되는 것을 말한다. 객체지향 프로그래밍은 객체를 정의하고 객체를 이용한 프로그래밍 방식을 말하고, 객체지향 분석 및 설계는 객체에 바탕을 둔 시스템 분석 및 설계 기법을 말한다. 객체지향 방법론의 주된 활용은 현실세계를 정보 시스템 상에서 정확하게 묘사할 수 있어야 하며 사용자의 요구, 정의, 방법 변경에 유연하게 대응하여야 하며 소프트웨어 개발의 산출물을 재사용 할 수 있어야 한다. 객체지향방법론에 따라 작성되는 산출물은 크게 분석, 설계, 개발, 구현단계로 구분하여 작성된다. 분석과 설계 단계의 산출물은 표준 객체지향 모델링 방법인 UML에 의하여 표현된다.

테스트작업의 유형과 프로세스는 소프트웨어 개발 단계와 밀접히 관련되어 있다. 테스트 작업이 프로그램에 포함된 단순한 코딩의 오류만을 찾는 작업이 아니라 요구사항분석에서의 오류, 설계 또는 모듈 인터페이스의 오류 등 개발단계의 작업들에 대한 테스트를 포함하므로 개발 프로세스와 매핑시킬 필요가 있다. 소프트웨어 오류에 대한 통계를 보면 구현 단계보다 분석과 설계 단계에 더 많은 오류가 발생하므로 개발과정 중 분석과 설계 단계에서부터 테스트 작업과 통합하여 조기에 오류를 찾아내는 방법이 필요하다.

2.2. 요구사항의 일관성 유지를 위한 "Engineering Baseline Process"



[그림 2] Engineering Baseline Process

기존의 요구사항의 추적기법이 너무 광범위하고 고수준 요구사항에서 저수준의 요구사항을 추적하는 것이 애매모호했다. 이러한 문제점을 해결하기 위해 "Engineering Baseline Process"에 대한 연구가 진행되었다[5].

1. Organization
요구사항 명세안의 개별 요구사항을 ASCII코드로 변환한다.
2. Numberization
3가지의 Index 넘버를 생성한다. 시스템 넘버, 시스템 넘버와 연관된 engineering 넘버, 변경 넘버 (xxx.yyy.zzz)
3. Specification Generation
변경이력을 가진 요구사항 문장과 추적 태그를 가진 요구사항 문장을 생성한다.
4. Change Management
시스템 넘버의 이력을 유지하기 위한 프로세스이다.

2.3 명세기반의 테스트케이스 생성

테스트를 자동으로 생성하는 연구는 크게 코드기반 방법과 명세기반 방법으로 나눌 수 있다. 코드기반 방법은 소스코드를 기반으로 테스트데이터를 생성하는 방법이고, 명세기반 방법은 프로그램에 대한 명세로부터 테스트데이터를 유도하는 방법이다.

명세기반 테스트데이터 생성 방법은 모듈이 요구사항에 맞게 잘 작동하는가에 초점을 맞춘 것으로 프로그램의 원시코드를 분석하지 않고 기능중심으로

테스트를 수행하는 블랙박스 테스트라고 할 수 있다. 요구사항을 기반으로 UML의 유즈케이스다이어그램, 시퀀스다이어그램, 상태차트다이어그램 등을 이용하여 프로그램의 이벤트를 시나리오로 작성하여 시나리오를 기반으로 테스트케이스를 작성하는 기법 등이 있다[6]. 명세기반의 테스팅의 장점은 테스팅 관련 정보를 집중화시켜 쉽게 테스팅 정보를 변경할 수 있고 다양한 테스트케이스를 생성할 수 있으며 테스트케이스의 복잡도를 제어하고 관리가 용이하다.

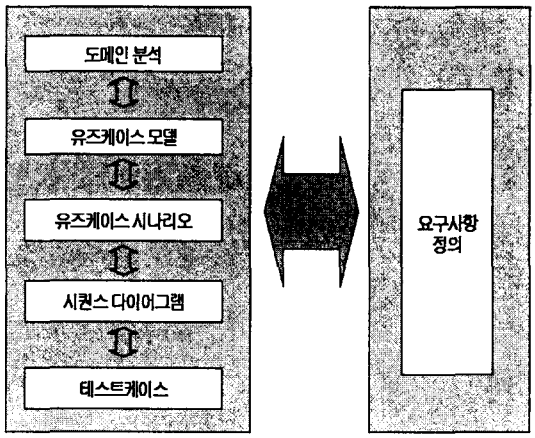
3. 요구사항과 테스트케이스 간의 일관성 유지

실제 소프트웨어를 개발하는 과정에서 단계별로 작성되는 산출물 간에는 많은 불일치성이 존재하므로 향후 테스트에 이용할 때 오류를 발생할 수 있다. 본 논문에서는 객체지향 소프트웨어 개발 단계 초기에 정형화된 요구사항 분석을 통하여 요구사항의 추적성을 유지할 수 있는 기법을 제시하여 향후 테스트케이스와의 일관성을 유지하여 개발 초기에 결함을 식별, 수정하고 변경 및 추가 시에 빠르게 적용할 수 있는 효율적인 테스트케이스를 생성하는 데 목적이 있다. 본 논문은 객체지향 개발방법론을 통해 생성되는 산출물들을 대상으로 하며 분석 및 설계 단계에 중점을 둔다.

3.1 객체지향 접근방법을 이용한 요구사항 추적

본 논문에서 중점을 두는 추적대상은 요구사항 명세와 UML의 유즈케이스다이어그램, 시퀀스 다이어그램, 클래스다이어그램등의 객체지향 개발방법을 통해 산출되는 분석, 설계 산출물이다. 시스템을 개발하는 단계별로 선정된 추적대상을 연결할 수 있는 요소를 분석하여 이를 연결하고 추적하기 위한 절차는 다음과 같다. [그림 2]는 요구사항 추적 절차를 나타낸다.

① 개발하려고하는 도메인을 명확히 이해하고 요구사항을 상세화하기 위해 여러 가지 다양한 기법을 적용하여 요구사항을 도출한다.
② 요구사항을 도출하기 위한 적용기법은 워크샵, 인터뷰, 설문조사, 브레인스토밍, 스토리보드, 프로토타이핑, 시나리오 등을 이용한다.
③. ②의 기법들은 요구사항을 역추적하기 위한 대상으로 선정한 후 요구사항정의서에 추가한다.
④ 도출된 요구사항을 기반으로 액터와 유즈케이스를 식별한 후 유즈케이스모델과 유즈케이스시나리오를 작성한다.
⑤ ④에서 작성된 산출물은 요구사항을 순방향으로 추적하기 위한 대상으로 선정한 후 요구사항정의서에 추가한다.
⑥ 유즈케이스시나리오를 실현한 시퀀스다이어그램을 작성한다.
⑦ 하나의 유즈케이스시나리오와 시퀀스다이어그램을 연결한다.
⑧ 모든 정보를 기반으로 요구사항정의서를 작성한다.
⑨ ⑧을 기반으로 테스트케이스를 작성한다.
⑩ ⑨를 포함하여 요구사항정의서를 정제한다.



[그림 2] 요구사항 추적 절차

3.2 추적가능한 요구사항 명세의 작성

위의 절차를 따라서 요구사항 명세를 작성한다. 분석, 설계, 테스트를 연결하는 식별자를 부여하여 요구사항 명세에 추가한다. 다음과 같은 식별자들이 사용되어 요구사항의 순방향 및 역방향 추적을 가능하게 한다. 도출된 식별자는 다음과 같다.

■요구사항도출기법
■요구사항 ID
■유즈케이스 ID
■유즈케이스시나리오 ID
■시퀀스 ID
■테스트케이스 ID
■관련요구사항 ID

이와 같은 식별자를 통하여 분석 및 설계 단계의 산출물과의 일관성을 유지하고 추적이 가능하다. 또

한 테스트케이스와의 추적성도 고려하여 향후 테스트 시에 발생되는 원인을 찾아내고 결함을 제거 할 수 있다.

3.3 테스트케이스 생성

4. 요구사항 명세의 검증 및 효율성

명세기반테스트 시에 테스트케이스의 중복을 제거하고 요구사항을 충분히 검토하였는지 확인가능하며 요구사항의 변경을 효과적으로 수행할 수 있다. 이를 기반으로 자동화가 가능하다.

요구사항 간의 추적성은 연관 있는 요구사항끼리의 링크 정보를 관리하여 요구사항이 변경되었을 경우 관련 있는 요구사항이 영향을 받는지를 파악하여 요구사항 변경 커버리지를 평가하기위해 주로 사용된다.

역방향 추적성은 각각의 요구사항의 원천에 해당하는 이전 단계의 문서정보를 관리하여 프로젝트 범위를 명확히 하는데 활용하도록 한다.

순방향 추적성은 각각의 요구사항의 구체화에 해당되는 주요 결과물에 대한 추적으로 일반적으로 모델링 결과물, 설계 결과물, 구현된 프로그램 결과물, 테스트케이스를 대상 주요 산출물로 선정한다. 이를 통해 요구사항이 변경되었을 경우 분석, 설계, 구현에 미치는 변경의 커버리지를 평가하는데 활용한다.

상위수준의 요구사항으로 도출된 모든 요구사항을 만족하는지를 검증하기 위한 커버리지분석에 용이하며 요구사항 추척을 통해 더 좋은 설계를 생성하고 또한 개발단계 초기에 잠재적인 오류를 검출할 수 있다.

4. 결론 및 향후연구

본 논문에서는 객체지향 개발 방법론을 이용한 명세기반의 소프트웨어 테스트를 위한 요구사항 추적 기법을 제안하였다. 일관성 있는 요구사항도출을 통하여 산출물간의 추적성을 제공하고 이를 통해 테스트케이스를 생성하여 모든 요구사항을 만족하는 테스트케이스를 생성할 수 있다. 또한 요구사항의 순방향, 역방향 추적이 가능하여 요구사항 변경을 쉽게 해주고 초기에 소프트웨어의 오류를 발견하고 이를 수정할 수 있다. 향후 연구로는 좀더 정형화된 추적을 위한 기법을 연구하고 각 단계마다 산출되는 산출물을 효율적으로 테스트하기 위한 기법을 추가하여 설계와 분석단계에 테스트를 고려한 테스트 요소를 고려할 수 있는 요소를 적용하여 이를 자동화 하고자 한다. 또한 ISO9126을 기반으로 하는 품질특성을 요구사항에 고려하여 비 기능적인 속성도 고려하는 중이다.

참고문헌

[1] William E.Lewis, Software Testing and Continuous Quality Improvement, CRC Press LLC, 2000

[2] Roger S. Pressman, Software Engineering A Practitioner's Approach 5Th Ed. McGrawHill

[3] D. Kung, "Testing Object-Oriented Software", IEEE Computer Society, 1998

[4] Davis, Alan. Software Requirements: Objects, Functions, and States. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[5] Sooyong Park, "Requirements Management in Large Software System Development", Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on , Volume: 3 , 11-14 Oct. 1998

[6] Verlin Kelly, "Requirement Testability and Test Automation", Software Technology Conference 2002, 2002

[7] Daniel R. Windle, Software Requirements Using the Unified Process: A Practical Approach, Prentice Hall. August 2002

[8] Hutcheson, Software Testing Fundamentals: Methods and Metrics, Willey, 2003

[9] Pinheiro F.A.C., "An Object-Oriented Tool for Tracing Requirements", Software, IEEE , Volume: 13 , Issue: 2 Pages:52-64, March 1996