DOI: 10.3745/KIPSTD.2010.17D.6.423

가중치를 이용한 소프트웨어 테스트케이스 동적 관리 기법

한 상 혁[†] · 정 정 수[††] · 진 성 일[†††] · 김 영 국[††††]

요 약

소프트웨어가 대형화 및 복잡화되고 품질보증 및 관리에 대한 요구가 한층 커짐에 따라 소프트웨어 테스트 활동이 중요시되고 있다. 소프트웨어 테스트 활동은 시스템에 존재하는 결함 발견과 처리가 주요 목적이지만, 현재 시스템의 위험요소를 관리하기 위한 요구 또한 존재한다. 하지만 일반적인 테스트 자동화 도구에서는 테스트케이스를 이용하여 동일한 순서로 테스트를 수행하며, 이러한 테스트 방식은 빈번하게 변경되는 소프트웨어를 테스트함에 있어 결함을 조기에 발견할 가능성이 낮아지고 처리되는 시간이 늦어지게 된다. 이에 따라 본 논문에서는 과거 테스트 이력을 이용하여 테스트케이스에 동적인 가중치를 부여함으로써 위험도가 높은 테스트케이스를 우선적으로 테스트 하여 결함을 빠르게 발견할 수 있도록 가중치를 이용한 테스트케이스 동적 관리 기법을 설계하였다.

키워드 : 테스트케이스, 가중치, 소프트웨어 테스트, 테스트 자동화 도구, 품질 보증

A Dynamic Management Technique for Weighted Testcases in Software Testing

Sang-Hyuck Han[†] · Jung-Su Jung[††] · Seung-Il Jin[†††] · Young-Kuk Kim[††††]

ABSTRACT

As software becomes large-scale and complicated, the need for Quality Assurance and management is increased and software testing is becoming more important. The main aims of software testing are not only detecting and handling the defects in the system but also investigating and managing the present system. But automatic testing tools require lots of time and efforts to detect and manage the risk in the system because test-cases used in the general automatic testing tools have the simply static information. In this thesis, the dynamic management technique for weighted testcases is designed to test the high-risk testcases preferentially by giving the testcases dynamic weight.

Keywords : Testcase, Weights, Software Test, Test Automation Tool, Quality Assurance

1. 서 론

소프트웨어가 대형화 및 복잡화됨에 따라 품질보증 및 관리에 대한 요구가 한층 커지고 있다. 소프트웨어 품질관리를 위한 다양한 방법들 중에서 가장 널리 사용되는 기법이 소프트웨어 테스트 기법이다. 과거의 소프트웨어 테스트는 대부분 개발자나 품질 관리자에 의해서 수작업으로 진행되었다. 그러나 소프트웨어의 규모가 방대해지고 입출력이 복잡해짐에 따라 소프트웨어 테스트에 소요되는 시간과 비용이 급격하게 증가하고 있다. 이를 완화하기 위해서 최근 소프트웨어 테스트 자동화에 대한 관심이 크게 높아지고 있다[1].

테스트 자동화의 필요성이 부각되면서 많은 테스트 자동화 도구들이 개발되어 사용되고 있다. 테스트 자동화 도구에서 사용되는 테스트케이스들의 데이터와 순서는 일반적으로 고정되어 있으며, 필요에 따라 데이터와 실행 순서가 변경될 수 있다.

테스트는 시스템에 존재하는 결함의 발견과 처리가 주요 목적이며, 현재 시스템의 위험요소를 파악하고 관리하기 위한 요구 또한 존재한다. 하지만 정적인 테스트 케이스를 이용하는 테스트 자동화 도구로는 시스템의 위험요소를 파악하고 관리하는데 추가적인 노력이 필요하다.

※ 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 육성·지원사업(NIPA-2010-C1090-1031-0002)의 연구결과로 수행되었음.
† 정 회 원 : (주)리얼타임테크 기술지원실장
†† 정 회 원 : (주)케이사인 DB보안팀 선임연구원
††† 종신회원 : (주)리얼타임테크 대표이사
†††† 종신회원 : 충남대학교 컴퓨터공학과 교수(교신저자)
논문접수 : 2010년 6월 3일
수 정 일 : 1차 2010년 7월 13일, 2차 2010년 7월 26일
심사완료 : 2010년 7월 26일

한편, 데이터베이스에 대한 활발한 연구로 국산 DBMS가 상용화되어 상당한 수준의 시장을 창출하고 있다. DBMS와 같은 시스템 소프트웨어는 간단한 소프트웨어처럼 테스트 가능한 모든 케이스를 생성하고 테스트하는 것이 거의 불가능하다. 하지만 최대한의 신뢰성을 확보하기 위해 수만 개 이상의 테스트케이스를 생성, 관리하며 전체 또는 일부에 대해 테스팅을 수행한다.

현존하는 테스트 자동화 도구는 테스트케이스의 일부 또는 전체를 임의의 순서로 테스트하는데, 환경적, 시간적 제약으로 전체 테스트케이스 중 일부만을 시험하거나, 결함을 발견할 확률이 높은 테스트케이스를 시험하고자 할 때는 부적합하다. 일부만을 시험하고자 할 때는 테스트케이스에 서브시스템 분류정보를 두거나, 요구사항의 중요도 정보를 활용하게 되는데, 정적으로 관리되어 테스트를 수행하면서 축적되는 테스트케이스의 특성을 미반영하게 된다.

임의적 테스트케이스 선정방식을 보완하고, 테스트 과정에서 발생하는 각 테스트케이스의 결함 수, 결함 재 발생등을 반영하여 우선순위 및 중요도를 부여하고, 테스트 자동화 도구에서 중요도를 관리하는 방법을 고려할 수 있다.

본 논문에서는 복잡한 시스템 소프트웨어의 시스템 테스트 과정에서 결함 재 발생, 결함 수 등 과거 테스트 이력을 이용하여 테스트케이스에 동적인 가중치를 부여하고 테스트케이스 간 우선순위를 평가하여, 시스템의 위험요소를 관리하고 위험도가 높은 테스트케이스가 우선적으로 테스트 되도록 가중치를 이용한 테스트케이스 동적 관리 기법을 설계하고, 이를 기존의 임의적 테스트케이스 선택하여 시험하는 방식과 비교한다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 대표적인 테스트 자동화 도구와 DBMS 테스트 도구에 대해서 알아본다. 3장에서는 테스트케이스 동적 관리 기법 개요, 가중치 선정, 동적 관리 알고리즘과 우선순위 변화에 대해 기술한다. 4장에서는 3장에서 제안한 테스트케이스 동적 관리 기법을 적용하였을 때 기존 방식과 비교하여 효율성을 알아본다. 마지막으로 5장에서는 결론 및 향후 연구방향에 대해 기술한다.

2. 관련연구

2.1 테스트케이스

테스트케이스는 특정 프로그램 경로를 실행해 보거나 특정 요구사항의 준수여부를 확인하기 위해 개발된 입력값, 실행 조건, 예산된 결과값의 집합이다[10]. 테스트케이스는 SW의 개발, 운영, 유지보수에 따라 지속적으로 추가, 삭제, 변경된다. 테스트케이스 추가는 개발단계의 품질 활동 과정, 운영 및 유지보수 단계에서 신규 기능의 도출시 수행된다. 테스트케이스 삭제는 기존 테스트케이스가 더 이상 결함을 발견치 못하거나, 신규로 등록된 테스트케이스가 기존 테스트케이스의 내용을 포함할 때 수행된다. 테스트케이스 변경은 결함을 효과적으로 발견하거나, 소프트웨어의 기능에 변경사항이 발생했을 때 이를 반영하기 위해 수행된다.

SW가 기능이 많고, 복잡할수록 테스트케이스의 수도 급격히 증가하는데, 효과적인 SW 테스트를 위해서는 최적화된 테스트케이스의 도출과 각 테스트케이스간의 관계를 식별하는 것이 중요하다. 특히, SW 테스트 시점에 전체 테스트케이스를 시험하는데 시간적인 제약이 있는 경우, 중요도가 높은 테스트케이스를 대상으로 SW 테스트를 수행함으로써 SW의 품질 저하 징후를 판단할 수 있다.

2.2 대표적인 테스트 자동화 도구

<표 1>에서 살펴 본 테스트 도구들은 정적인 정보를 가지는 테스트케이스를 사용한다. 이 밖에 다양한 종류의 테스트 도구들이 사용된다. 현재 테스트 도구 중 테스트케이스의 위험도를 수동으로 파악하여 특정한 순서로 수행되는 경우는 있었지만, 테스트케이스를 동적으로 평가하여 관리하는 도구는 찾아볼 수 없었다.

〈표 1〉 대표적인 테스트 자동화 도구

도구이름	용도	설명	테스트케이스 타입
NIST SQL Test Suite[12]	기능 테스트	DBMS의 SQL2 적합성 평가	정적
C++Test[1]	단위 테스트	클래스의 메소드 단위 테스트를 위한 테스트 드라이버 생성	정적
QADirector[1]	프로세스 관리	테스트 계획, 실행, 분석, 결함 추적	정적
WinRunner[1]	기능 테스트	테스트케이스 자동 생성, 점검 및 수행	정적

2.3 DBMS 테스트 도구

데이터베이스 시스템의 테스트 도구는 크게 성능 테스트, 기능 테스트로 구분할 수 있다. 현재 쓰이고 있는 성능 테스트로는 트랜잭션 처리협회에서 지정한 TPC 벤치마크, 위스콘신 벤치마크, AS3AP 등이 있는데 가장 널리 쓰이고 있는 것은 TPC-A, TPC-B, TPC-C TPC-H로 구분되는 TPC 벤치마크 테스트이다[2-5]. TPC 벤치마크 테스트 중 TPC-H는 응용분야에서의 복잡질의 처리에 대한 성능평가를 위한 테스트로 이용되고, TPC-B는 변경 오퍼레이션이 많은 데이터 서비스를 모델링 한 벤치마크 기법으로 은행업무를 기반으로 하여, 초당 트랜잭션 처리 성능 및 서버의 동시 수행 능력 검증을 위한 테스트 도구이다.

대표적인 기능테스트 도구로는 NIST에서 만든 ANSI SQL의 적합성 테스트인 NIST SQL Test Suite를 들을 수 있다. NIST SQL Test Suite는 ANSI X3.135-1992 및 ISO/IEC9075:1992에 근거한 Federal Information Processing

Standards(FIPS) FUB 127-2에서 정의한 필수적인 요구사항에 대한 SQL 구현의 적합성을 평가하는 것을 목적으로 하고 있다. 그러나 NIST SQL Test Suite는 표준화된 SQL2의 완전 수준이나 확장된 기능을 테스트하여 주지 못하는 한계성을 지니고 있다[6].

3. 테스트케이스 동적 관리 기법 설계

테스트케이스의 동적 관리 기법은 테스트 수행에 앞서 이전 테스트 상태를 파악하여 도구에 등록되어 있는 테스트케이스의 우선순위를 선정하고, 그 결과를 이용하여 현재 중요도와 긴급도가 높은 테스트케이스를 우선적으로 테스트함으로써 시스템의 결함을 빠르게 발견하고 처리하는 것을 목표로 한다.

3.1 테스트케이스 동적 관리 기법 개요

본 논문에서 제안하는 테스트케이스 동적 관리 기법은 테스트 수행에 필요한 테스트케이스 간의 우선순위를 과거 테스트 결과를 반영한 동적인 평가 항목을 이용하여 테스트케이스를 정량적으로 표현, 관리하고 테스트를 수행하는데 있어 위험도가 높은 테스트케이스를 우선적으로 수행하는 테스트 기법을 말한다.

테스트케이스의 정량적인 표현은 테스트케이스의 정성적 평가 항목을 선정하고 각 속성 값으로서 점수를 산출한 후 가중치를 부여하는 방식을 이용한다. 하지만, 정성적인 평가 값으로만 평가를 하게 되면 정밀한 평가를 할 수 없는 문제점이 있다. 따라서 테스트케이스 그룹을 정량적으로 평가하여 얻어질 수 있는 평가 항목을 선정하고 두 평가 값을 조합하여 테스트케이스 우선순위를 선정한다.

테스트케이스 동적 관리 기법에서 테스트케이스를 정량적으로 관리하는 방법은 테스트케이스 평가 값과 테스트케이스 그룹 평가 값을 산출하는 두 부분으로 분류된다. 또한 평가 값을 산출하기 위한 사전작업으로, 테스트케이스를 분류하기 위한 그룹 선정과 테스트케이스를 정성적으로 평가하기 위한 평가속성 선정 작업을 수행한다.

3.2 테스트케이스 평가 항목 선정

테스트케이스 동적 관리 기법은 위험도가 높은 테스트케이스를 우선적으로 처리하도록 하며, 테스트케이스의 위험도를 선정하기 위한 방법으로 프로젝트 위험분석 고려 사항 중 테스트 과거 이력과 테스트 수행으로 평가될 수 있는 것을 평가항목으로 선정하였다.

소프트웨어 개발 중 위험 분석 고려 사항은 아래와 같다[7].

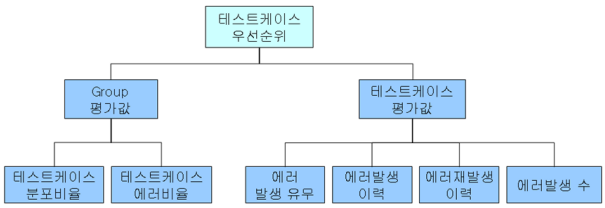
- 프로젝트가 의도하는 목적에서 가장 중요한 기능은 무엇인가?
- 어떤 기능이 안전성에 가장 많이 영향을 많이 주는가?
- 어떤 기능이 사용자에게 금전적인 영향을 가장 많이 주는가?
- 제품의 어떤 면이 고객에게 중요한가?
- 제품의 어떤 면이 제품 개발과정 중에서 먼저 테스트될 수 있나?
- 어떤 부분의 코드가 가장 복잡한가?
- 가장 결함 유발 경향이 많은 코드는?
- 어플리케이션의 어떤 부분이 급하고 정신없이 작성 되었나?

위의 고려 사항 중 대부분은 평가자의 의견이 반영되어 주관적 평가가 이루어지는데 객관적인 평가가 이루어지도록 복잡도와 결함 관련 고려사항인 “어떤 부분의 코드가 가장 복잡한가?”와 “가장 결함 유발 경향이 많은 코드는?”을 통해 결함 발생 유무, 결함 발생 이력, 결함 재 발생 이력, 결함 발생 수와 같은 결함 관련항목을 테스트케이스 평가 항목으로 선별하였다.

Bohem과 Basili는 [11]에서 20%의 소프트웨어 모듈에서 대략 80%결함이 나오고, 절반의 모듈에서는 결함이 없는 결과를 얻었고, 결함 발생 가능성이 높은 20%에 테스트를 집중하는 것이 효율적임을 나타낸다. 이는, 위에서 선택한 각 테스트케이스의 평가 항목이 적절함을 의미한다.

테스트케이스 그룹의 평가 항목은 복잡도와 결함관련 항목으로서 그룹의 테스트케이스 분포도와 결함 비율을 선정하였다. 분포도와 결함 비율을 평가항목으로 선정한 이유는 테스트케이스를 소프트웨어가 제공하는 기능들로 분류하였을 경우 복잡도가 높은 기능을 테스트하기 위한 그룹이 다른 그룹에 비해 많은 테스트케이스가 존재하게 되고, 결함 비율이 높은 그룹은 결함 비율이 낮은 그룹에 비해 높은 위험도를 가지게 되므로 두 평가항목을 이용한다.

(그림 1)은 테스트케이스 관리 기법에서 테스트케이스 우선순위를 평가하기 위한 속성이다.



(그림 1) 테스트케이스 우선순위 평가 속성

3.3 테스트케이스 가중치 산정

가중치의 결정에는, 각 항목의 가중치 점수를 준 후 원하는 총점을 전체항목의 점수 합으로 나눈 값을 각 항목의 점수에 곱하여 가중치를 결정하는 방법과, 사전에 총점을 정해주고 평가자가 총점을 고려하여 각 항목에 대한 가중치를

부여하는 상수 합 척도 방법이 있다[8].

위의 가중치 결정 방식은 평가자의 주관적인 판단으로 가중치를 판단하며 가중치가 결정되면 가중치 값이 고정되어 처리되므로 테스트는 항상 동일한 순서로 진행되는 정적인 테스트가 진행된다. 이러한 정적인 테스트 수행 방식은 지속적으로 변경되는 소프트웨어를 테스트하는데 결함을 조기에 발견 할 가능성이 낮아지며, 결함이 처리되는 시간 또한 늦어진다. 이에 따라 빈번하게 변경되는 소프트웨어의 테스트에서 테스트 과거 이력을 반영하여 3.2장에서 선정된 그룹의 테스트케이스 분포도와 결함비율, 테스트케이스의 정성적인 평가 항목에 가중치 값을 동적으로 부여하는 방식을 선정하였다.

3.4 테스트케이스 동적 관리 알고리즘

본 절에서는 3.2장에서 선정된 평가 항목 값을 산정하기 위한 테스트케이스 동적 관리 알고리즘에 대해서 기술한다.

테스트케이스 그룹은 (그림 2)와 같이 트리 구조 형태를 가지며 단말 그룹에 테스트케이스가 존재하게 된다. 아래 식은 테스트케이스 동적 관리 알고리즘을 보인다.

$$IP(TC) = E(TC) + E(G)$$

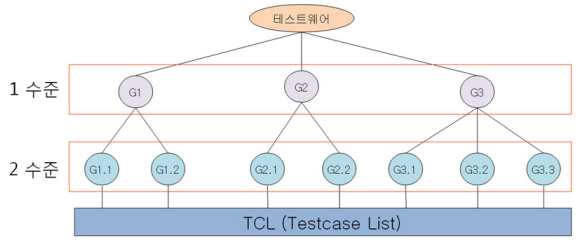
테스트케이스 중요도 점수(Impact Point)는 테스트케이스 평가 값 E(TC)와 그룹의 평가 값 E(G)의 합으로 계산된다.

$$E(TC) = W(TC) * R_{TC}$$

테스트케이스 평가 값 E(TC)는 테스트케이스 평가항목 가중치 합인 W(TC)의 보정값(R_{TC})을 반영하여 계산한다. W(TC) 값은 <표 2>의 평가항목을 이용하여 값이 부여된다.

<표 2>에서 평가항목은 고정적으로 유지되는 항목과 테스트 결과에 따라 동적으로 유지되는 항목을 가진다. 동적/정적인 유지 방식은 결함이 발생 되었던 기능은 결함을 수정하는 과정에서 발생할 수 있는 개발자의 실수로 발생할 수 있는 side effect 가능성을 고려하여 점수를 누적 시키는 방식을 이용하였으며, 동적으로 유지되는 속성은 테스트 수행 테스트 결과에 따라 달리 부여된다.

$$E(G) = E(G_{DR}) + E(G_{ER})$$



(그림 2) 테스트케이스 저장 구조

〈표 2〉 테스트케이스 가중치 척도

평가항목	설명	가중치	유지 방식
결함발생 유무	테스트 수행 시 결함 발생 유무	2	동적
결함발생 이력	이전 테스트 수행 시 결함 발생 이력	1	정적
결함 재발생 이력	결함 재 발생 이력	1	정적
결함 발생 수	총 결함 발생 수	1	정적

그룹 평가 값 E(G)는 그룹의 테스트케이스 분포 비율평가 값 $E(G_{DR})$과 에러 비율 평가 값 $E(G_{ER})$의 합으로 산출된다.

$$E(G) = W(G_{DR})*R_{DR} + W(G_{ER})*R_{ER}$$

그룹 평가값 E(G)는 테스트케이스 그룹에 포함되는 테스트케이스의 분포도 가중치 $W(G_{DR})$에 분포 비율 보정 값(R_{DR})을 곱한 평가값 $E(G_{DR})$과 테스트케이스의 결함 비율 가중치 $W(G_{ER})$에 에러 비율 보정값(R_{ER})을 곱한값 $E(G_{ER})$를 합산하여 산출된다.

그룹이 (그림 2)와 같이 구성되어 있는 경우, 각 수준에 n개의 그룹이 존재 할 때 동일한 수준의 G_{DR} 값은 다음의 성질을 만족한다.

$$\sum_{i=1}^{n} G_{DR_i} = 1$$

위의 성질은 G_{ER}에도 동일하게 적용된다.

$W(G_{DR})$ = (그룹 G에 포함되는 테스트케이스 수/전체 그룹의 테스트케이스 수)의 가중치 값

$$W(G_{DRk}) = \frac{G_{DR_k}}{\sum_{i=1}^{n} G_{DR_i}} \text{의 가중치 값}$$

$W(G_{ER})$ = (그룹 G에 포함되는 테스트케이스 결함비율/전체 그룹의 테스트케이스 결함비율)의 가중치 값

$$W(G_{ERk}) = \frac{G_{ER_k}}{\sum_{i=1}^{n} G_{ER_i}} \text{의 가중치 값}$$

<표 5>는 그룹 수준이 증가하였을 때 평가항목의 누적 가중치 허용 범위를 보인다.

<표 5>에서 $W(G_{DR})$ 가중치의 허용 범위는 단계가 높아질수록 최대값이 $W(G_{ER})$에 비해 낮게 부여되는 것을 볼 수 있다. 실제로 $W(G_{DR})$의 값은 $W(G_{ER})$과 동일한 분포를 가질

〈표 3〉 테스트케이스 분포 비율(DR)의 가중치 척도

평가등급	G_{DR}	가중치
없음	0	1
낮음	0 이상	2
보통	0.2 이상	3
높음	0.4 이상	4
매우높음	0.6 이상	5

〈표 4〉 테스트케이스 결함 비율(ER)의 가중치 척도

평가등급	G_{ER}	가중치
없음	0	1
낮음	0 이상	2
보통	0.3 이상	3
높음	0.6 이상	4
매우높음	0.9 이상	5

〈표 5〉 테스트케이스 그룹 수준별 가중치 허용 범위

그룹	$W(G_{DR})$	$W(G_{ER})$	W(TC)
1 단계	2~5	2~5	2~5
2 단계	4~9	4~10	2~5
3 단계	6~12	6~15	2~5

수 있지만, $W(G_{DR})$이 낮은 수치로 증가되는 이유는 테스트케이스를 그룹화 할 경우 그룹의 수준이 높아질수록 그룹의 수는 증가하고 각 그룹에 포함되는 테스트케이스 수는 적어지기 때문이다.

테스트케이스 그룹이 1 수준일 경우 W(TC), $W(G_{DR})$, $W(G_{ER})$ 값은 가중치 척도에 의해 1에서 최대 5의 값을 가지게 되며, 그룹의 수준이 높아짐에 따라 $W(G_{DR})$, $W(G_{ER})$의 최대값을 변경한다. 이러한 방식은 그룹의 수준에 따라 최대값이 변경되고 우선순위 선정에서 $W(G_{DR})$ 값에 의해 좌우될 수 있기 때문에 테스트케이스 중요도 점수(Impact Point)의 최대값을 100으로 정하였으며, 그룹 수준이 변경되더라도 E(TC), $E(G_{DR})$, $E(G_{ER})$의 합이 100으로 환산 되도록 보정 값(Revision, R)을 달리한다.

본 논문에서는 그룹 수준에 따라 평가 항목의 점수를 <표 7>의 평가 값에 만족하도록 하였다.

(그림 3)은 2 수준의 그룹으로 구성되어 있으며, 테스트케이스 개수와 결함 개수 정보를 가지는 1수준의 3개의 그룹과 2 수준의 13의 그룹을 표현 한 것이다. (그림 3)에서 첫 번째 표는 그룹에 포함되는 테스트케이스의 개수와 결함 개수이며, 이 값들을 이용하여 두 번째 표와 같이 그룹의 테스트케이스 분포 비율(G_{DR})과 그룹의 테스트케이스 에러 비율(G_{ER})의 값을 구한 후, G_{DR}과 G_{ER}값은 가중치 척도에 의해 세 번째 표와 같이 가중치가 구해진다. 가중치 값들은 보정 값이 반영되어 (그림 4)와 같이 계산된다.

〈표 6〉 테스트케이스 그룹 수준별 보정 값

그룹	R_{DR}	R_{ER}	R_{TC}
1 단계	2	8	10
2 단계	1	6	6
3 단계	0.8	6	2

〈표 7〉 테스트케이스 그룹 수준별 평가값

그룹	$E(G_{DR})$	$E(G_{ER})$	E(TC)	총점
1 단계	10	40	50	100
2 단계	10	60	30	100
3 단계	10	80	10	100

수준	그룹 1		그룹 2		그룹 3	
	TC수	에러수	TC수	에러수	TC수	에러수
1수준	5200	100	900	13	65	3
2수준	3000	100	100	0	30	0
	2000	0	100	3	20	1
	100	0	500	10	10	1
	100	0	100	0	5	1
			100	0		

수준	그룹 1		그룹 2		그룹 3	
	분포율	에러율	분포율	에러율	분포율	에러율
1수준	0.843	0.240	0.146	0.180	0.010	0.578
2수준	0.486	0.076	0.016	0.000	0.003	0.000
	0.016	0.000	0.081	0.046	0.001	0.230
	0.016	0.000	0.016	0.000	0.000	0.461
			0.016	0.000		

수준	그룹 1		그룹 2		그룹 3	
	분포도 가중치	에러율 가중치	분포도 가중치	에러율 가중치	분포도 가중치	에러율 가중치
1수준	5	2	2	2	2	3
2수준	4	2	2	1	2	1
	3	1	2	2	2	2
	2	1	2	2	2	2
	2	1	2	1	2	3
			2	1		

(그림 3) 그룹 가중치 산정

다음은 그룹이 2 수준인 경우, 테스트케이스의 우선순위 점수(Impact Point, IP)를 구하는 식을 보여준다.

$$IP(TC) = [W(1\text{수준의 } G_{DR}) + W(2\text{수준의 } G_{DR})] * R_{DR} + [W(1\text{수준의 } G_{ER}) + W(2\text{수준의 } G_{ER})] * R_{ER} + W(TC) * R_{TC}$$

(그림 4)의 그룹 평가 값 중 밑줄로 표시 된 값은 위 식

수준	그룹 1		그룹 2		그룹 3	
	그룹 평가값	TC 평가값	그룹 평가값	TC 평가값	그룹 평가값	TC 평가값
1수준						
2수준	**33**	0	22	0	28	0
	26	0	28	0	34	0
	31	0	28	0	34	0
	25	0	22	0	**40**	0
			22	0		

(그림 4) 그룹 평가 값 산정

을 이용하여 다음과 같이 계산된다.

$$(5+4)*1 + (2+2)*6 = 33$$
$$(2+2)*1 + (3+3)*6 = 40$$

위에서 산출 된 그룹평가 값 E(G)는 테스트케이스 평가 값 E(TC) 값과 합산하여 테스트케이스 중요도 점수(Impact Point)가 산정된다.

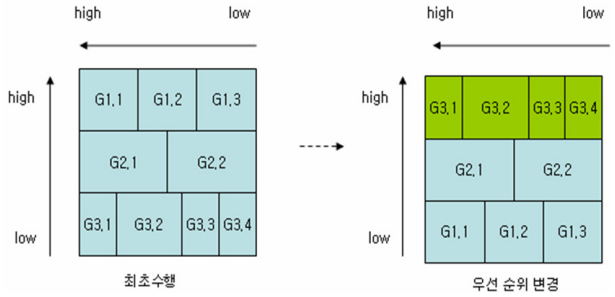
3.5 테스트케이스 우선순위 변화

테스트케이스 동적 관리 알고리즘은 최초 테스트를 수행할 경우 과거의 테스트 이력을 알 수 없기 때문에, 분포도가 높은 그룹에 포함되는 테스트케이스를 우선적으로 수행하게 된다. 최초 수행과정에서 결함이 발견된다면, 결함이 발견된 테스트케이스의 그룹과 테스트케이스에 가중치가 부여되고, 우선순위가 반영되어 테스트 수행 순서가 변경된다.

매 테스트 수행 시 결함 발생 횟수, 결함 발생 일, 발생 후 경과 일에 따라 테스트케이스의 가중치를 변경하고 테스트케이스 그룹의 결함 분포비율과 에러비율로 그룹 우선순위를 동적으로 조정한다.

결함 발생 횟수가 높고 최근에 결함이 발생된 테스트케이스가 높은 가중치를 얻게 되며, 결함 발생 없이 시간이 경과할 경우 가중치는 낮아지게 된다.

이처럼 테스트케이스의 평가치를 통해 현재의 문제점을 파악, 이전 시험에서 문제가 발생되었던 테스트케이스를 우선적으로 수행하여 결함을 빠르게 발견하고 현재 시스템의 위험 요소를 관리함으로써 결함 발생 가능성이 높은 테스트케이스를 우선적으로 수행하여 결함을 빠르게 검출하도록 한다.



(그림 5) 테스트케이스 우선순위 변경

4. 시험 및 결과

본 장에서는 DBT_SQL[9]에서 사용하는 전체 테스트케이스 일부를 대상으로 테스트케이스 동적 관리 기법을 적용한 제안한 시험 방식과 순차적 범위에 의한 테스트케이스를 적용한 기존 시험 방식을 수행하고 결과를 보인다.

기존 시험 방식은 각 테스트케이스에 대해 식별된 ID가 부여가 되고, 임의의 범위가 정해지면 ID가 작은순에서 임의의 개수만큼 테스트케이스를 선택하고 이를 시험하는 방식이다.

테스트 대상은 주기억장치 상주형 DBMS인 Kairos의 SQL 기능부분이며, 6개월간의 일일 시험의 결함 결과를 임의 선택 방식과 동적 시험 방식을 비교하였다.

평가에 사용된 테스트케이스는 SQL 92와의 준수성을 평가하기 위해 대상 SW의 매뉴얼을 각 요소들에 대해 기본적으로 점검해야할 사항들 중심으로 도출하고, 관련 기능에서 결함 발생시 추가, 수정된다. 테스트케이스는 <표 9>와 같이 2수준의 그룹으로 구성되며, 1수준의 그룹은 SQL, 내장함수로 구성되고, 2수준은 SQL에 포함되는 DDL, DML, DCL, DQL과 내장함수에 포함되는 SP와 UDF로 구성된다.

테스트 결과는 테스트케이스를 100개 단위로 분류하고 동적 관리 기법의 테스트 방식과 현재의 테스트 방식으로 테스트하였을 때 각 범위의 결함 검출 확률을 계산하였다.

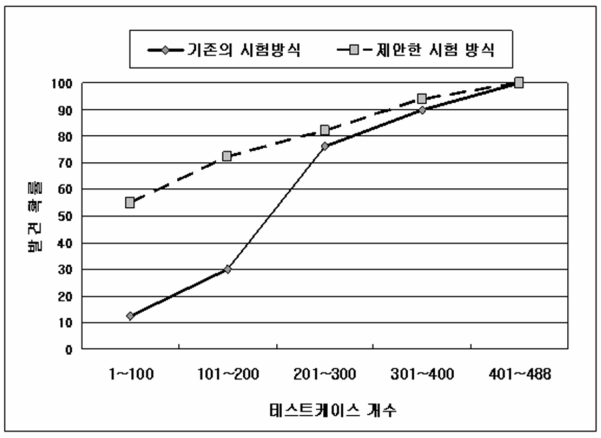
(그림 6)은 일일 단위로 6개월간의 각 테스트케이스의 시험 결과 중 임의 선택 방식과 동적 관리 기법간 수행한 결

〈표 8〉 시험 데이터

시험 데이터	
대상 시스템	Kairos[13]
테스트 기간	6개월
평가 기간	30일
테스트케이스 개수	488개
시험도구	EXCEL

〈표 9〉 테스트케이스 구성

1수준	2수준	테스트케이스 수
SQL	DDL	79
SQL	DML	47
SQL	DCL	3
SQL	DQL	238
내장함수	SP	114
내장함수	UDF	7



(그림 6) 기존 방식과 제안된 방식의 결함 발견 확률

과의 평균값으로 비교한 결과를 보여준다. 제안된 기법을 적용 하였을 경우 약 500개의 테스트케이스 테스트 중 100개 내에서 50%이상의 결함이 발견된 것을 볼 수 있다. 하지만 시스템에 존재하는 결함을 과거 이력으로 평가하므로 우선순위가 낮은 그룹에서 새로운 결함이 발생하는 경우 결함을 발견하는 시점이 늦어지는 문제가 발생 할 수 있다.

5. 결론 및 향후 연구 방향

본 논문에서는 복잡한 시스템 소프트웨어의 시스템 테스트 과정에서 테스트 과거 이력을 이용하여 테스트케이스에 동적인 가중치를 부여하고 테스트케이스 간 우선순위를 평가하여, 시스템의 위험요소를 관리하고 위험도가 높은 테스트케이스가 우선적으로 테스트 되도록 가중치를 이용한 테스트케이스 동적 관리 기법을 설계하였으며, 테스트케이스 동적 관리 기법의 효율성을 알아보기 위해 DBT_SQL의 과거 테스트 결과 중 일부를 이용하여 테스트를 수행하였다.

그 결과, 본 논문에서 제시한 알고리즘을 적용한 시험 방식과 기존의 시험 방식을 비교하였을 때 제안된 시험 방식이 테스트 수행 범위 40%이내에서 결함 발견 확률이 30% 이상 높은 수치를 보임으로써, 테스트케이스 동적 관리 기법에서 선별된 평가항목과 가중치 부여의 적합성 및 효율성에 대해 확인하였다.

향후 과제는 실 테스트 환경에 적용과 테스트 우선순위 선정 과정에서 기능 사이의 유사도를 평가하고 적용하여 본 논문에서 제안한 우선순위 선정 방법을 확장시켜 위험 분석의 효율을 높이기 위한 연구를 수행할 예정이다.

참 고 문 헌

[1] 배현섭, “소프트웨어 시험 단계별 자동화 지원 도구” 정보과학회지, 제23권 제3호, 2005. 3.

[2] Transaction Processing Performance Council, “TPC Benchmark A standard Specification”, 1989.

[3] Transaction Processing Performance Council, “TPC Benchmark B standard Specification”, 1990.

[4] Transaction Processing Performance Council, “TPC Benchmark C standard Specification”, 1992.

[5] Transaction Processing Performance Council, “TPC Benchmark H standard Specification”, Revision 2.6.1, 2007.

[6] 이상호, 이상구, 심준호, “SQL2 적합성 테스트도구의 설계 및 구현”, 한국정보과학회, 제2권 제3호, pp.266-275, 1996. 10.

[7] Rick Hower, “Software QA and Testing Consulting Services”, http://www.softwareqatest.com

[8] 김영복, 류성열, 정기원, “실시간 소프트웨어 개발방법론의 평가방안”, 한국정보과학회, 제19권 제2호, pp. 729-732, 1992. 10.

[9] 한상혁, 정정수, 유명호, 김영국, 진성일 “DBMS 기능테스트 도구 설계 및 구현”, KDBC2008, pp.176-180, 2008. 5.

[10] IEEE Std 610.12-1990, “IEEE Standard Glossary of Software Engineering Terminology”, IEEE, 1990. 12.

[11] Barry Boehm, Victor R. Basili, “Software Defect Reduction Top 10 List,” IEEE Computer, 34(1), pp.135-137, 2001. 1.

[12] NIST SQL Test Suite, http://www.itl.nist.gov/div897/ctg/sql_form.htm, NIST

[13] (주)리얼타임테크, “Kairos 4.8 사용자 매뉴얼”, (주)리얼타임테크, 2006.

한 상 혁

e-mail : han.sanghyuck@gmail.com
1998년 충남대학교 컴퓨터과학과(학사)
2000년 충남대학교 컴퓨터과학과(이학석사)
2000년~현 재 (주)리얼타임테크 기술지원실장
관심분야 : SW 품질 보증, SW 테스트, 실시간 DBMS

정 정 수

e-mail : circletea@naver.com
2005년 목원대학교 정보통신공학과(학사)
2008년 충남대학교 컴퓨터공학과(공학석사)
2010년~현 재 (주)케이사인 DB보안팀 선임연구원
관심분야 : DBMS, 분산처리 시스템, DB 암호화

진 성 일

e-mail : sijin@realtimetech.co.kr
1978년 서울대학교 계산통계학과(학사)
1980년 KAIST 전산학(공학석사)
1994년 KAIST 전산학(공학박사)
1981년~현 재 충남대학교 컴퓨터공학과 교수
2000년~현 재 (주)리얼타임테크 대표이사
관심분야 : 실시간 DBMS, 시공간 DBMS, Embedded DBMS

김 영 국

e-mail : ykim@cnu.ac.kr
1985년 서울대학교 계산통계학과(학사)
1987년 서울대학교 계산통계학과(이학석사)
1995년 버지니아대학교 컴퓨터과학과(공학박사)
1995년~1996년 핀란드 VIT, 노르웨이 SINTEF DELAB 방문연구원
2002년~2003년 미국 UC Davis 방문교수
1996년~현 재 충남대학교 컴퓨터공학과 교수
관심분야 : 실시간 및 모바일정보시스템, 센서 데이터베이스, 상황인지 개인화 시스템