테스트 케이스 정보를 공유하는 소프트웨어 테스팅 프레임워크

A TESTing Framewalk Sharing Test Case conformation

정창신[1], 김용성[2]

Chung Changsin[1], Kim Yongseong[2]*

요약

본 논문에서는 소프트웨어 테스팅에 소요되는 비용을 감소시키기 위하여 테스트 자동화 도구들의 통합화 대신에 테스트 과정에서 생성된 테스트 정보를 다른 테스트 도구에서도 접근, 공용할 수 있도록 하는 테스트 자동화 프레임워크를 제안하였다. 기존의 상용화된 테스트 자동화 도구들은 상호 호환성을 고려하지 않고 개발되었기 때문에 특히, 테스트 계획수립 단계에서 생성되는 테스트 케이스의 공유와 재사용이 거의 불가능하다. 현재까지 연구 결과에 의하면 자동 테스트인 경우는 테스트 계획수립 및 설계에 50-75%, 테스트 실행에 10-20%, 테스트 평가 및 시정 조치에 20-30%의 노력이 할당되고 있는 것이 일반적이다.. 따라서 자동테스트의 경우 테스트 설계 과정에서 생성된 테스트 케이스를 여러 테스트 도구들이 공유할 수 있다면 테스트에 소요되는 전체 비용을 감소시킬 수 있고, 본 논문은 이러한 점을 고려하며 소프트웨어 테스팅 프레임워크를 제안한다.

핵심어: 테스트케이스, 테스팅, 소프트웨어 테스트, 테스트 자동화 도구, 소프트웨어 플레임워크

Abstract

This paper suggests a software testing framework which makes it possible to approach and share the information of test cases earned in testing process with other test tools, rather than a unification of test automation tools. Thus, it seeks a way of reducing the cost of the software test. A test information repository was constructed on a testing framework in order for different test tools to share test case information earned at the planning and design stage. Three test tools and testing framework at the environment suggested in this paper were experimented. The results are as follows: first, treatment efficiency increased, showing 20 percent of reduction in CPU time; second, memory occupancy was reduced by an average of 10 percent. These results prove that test cases are reused in the testing automation framework in the unified environment and system functions, such as memory occupancy or treatment efficiency, have been enhanced.

Keyword: Softwaretesting, testcase, Framework, SW testing Framework

1 Department of Benchmarking Center, Telecommunication Technology Association 47, Bundang-Ro Bundang-Gu, Seongnam-city, Gyeonggi-do, 13591, Korea e-mail: csjung@tta.or.kr

2 Department of Computer science & Engreening, Cheonbuk Natational Univ, Bakjae-Ro, DuckJin-Gu, JeonJu-City, Cheonbuk-Do 54896, Korea e-mail: yskim@jbnu.ac.kr (Corresponding author)
Received(October 20.2014), Review (December 11.2014), Accepted(December 31.2014)

1. 서론

소프트웨어 개발에 소요되는 총 비용의 50%이상뿐만 아니라 총 기간의 50% 정도가 개발된 소프트웨어의 테스트 작업에 소요되고 있다. 이러한 테스트의 목적은 소프트웨어를 구성하는 요소들이 잘 조합되어 유효하게 작동되며, 요구된 성능을 충족시키는지를 점검하는데 있다[1].

그러나 개발된 소프트웨어를 충분히 테스트하지 못해 오류가 남아 있는 상태로 제품이 유통되고 활용하는 경우도 많다.

현재, 테스트 도구들은 업체에 따라 서로 다른 목적을 갖고 개발되어 있기 때문에, 도구들 간에 호환성이 결여될 뿐만 아니라 필요에 따라 도구를 선택하여 조합하여 활용할 수도 없다.

따라서, 소프트웨어 개발자나 테스트 엔지니어가 직접 테스트케이스를 설계하는데 많은 어려움이 있기 때문에, 테스트 계획 수립 과정에서 생성된 테스트케이스 정보의 축척으로, 이들은 재사용 할 수 있는 테스트 프레임워크가 필요한 시대가 되었다.

현재 상용화된 테스트 도구의 80% 정도가 테스트 수행의 자동화를 지원하고 있으며, 약 15% 정도는 테스트 관리의 자동화를 지원하고 약 5% 정도만 테스트 케이스를 자동 생성을 지원할 따름이다.

따라서, 소프트웨어 테스트 도구라 함은 주로 테스트 수행 도구를 의미한다고 볼 수가 있다[2].

본 논문에서는 소프트웨어 테스팅에 소요되는 비용을 감소시키기 위하여 테스트 자동차 도구들의 통합화하는 대신에 테스트 과정에서 생성된 테스트 케이스 정보를 다른 테스트 도구에서도 상호접근, 공유 할 수 있도록 하는 테스트 프레임워크를 제안하는데 그 목적이 있다.

또한, 본 논문에서는 테스트 설계 과정에서 생성된 테스트 케이스 정보를 다른 테스트 도구들도 상호 공유할 수 있도록 하기 위한 테스트 레파토리를 테스트 프레임워크 상에서 구축하였다. 기존의 테스트 도구와 본 논문에서 제안한 통합 환경의 테스트 프레임워크를 실험평가한 결과 CPU시간은 20%이상 처리 속도가 줄어들어 처리 효율이 증가했고, 메모리 점유율도 평균10%정도 줄어든 것으로 나타났다.

이러한 실험결과는 통합 환경의 테스트 프레임 워크 상에서 테스트케이스를 재사용함으로써 시스템전체가 개선되어 효율성이 향상된 것이라고 말할 수 있다.

2. 관련연구

2.1 소프트웨어 테스팅

초창기 1950년대에는 테스트를 프로그램을 작성하는 작업의 일부러 보았기 때문에, 프로그램을 작성한 후 테스트하는 것을 디버깅 작업이라고 생각하였다[3].

따라서 테스트는 오류를 발견하고, 이를 바로 잡거나 제어하는 작업이라고 간주 하였으나, 1980년 초 IEEE에서 "소프트웨어 테스트는 수동이나 자동으로 시스템을 시험 작동시키고 평가하는 작업으로 명시된 요구를 잘 만족여부, 즉 예상된 결과와의 차이를 인식하기 위한 것으로 정의가 실제화 되어 소프트웨어의 발전상을 알 수가 있었다[4].

이후, Glen Myers는 대부분 사람들이 테스트가 무엇인지에 대해 정확한 견해를 갖고 있지 않아, 테스트 작업이 부실해졌다고 주장하며, 테스트는 프로그램이 잘 작동되는 것을 보이는 것이라기보다는 프로그램에 포함된 오류를 찾아내려는 의도로 프로그램을 수행해보는 프로세스라고 정의[5]하였다. 이는, 테스트를 프로그래밍 이후에만 이루어 질 수 있는 작업으로 한정시켜 범위가 한정적이고 제한적이다.

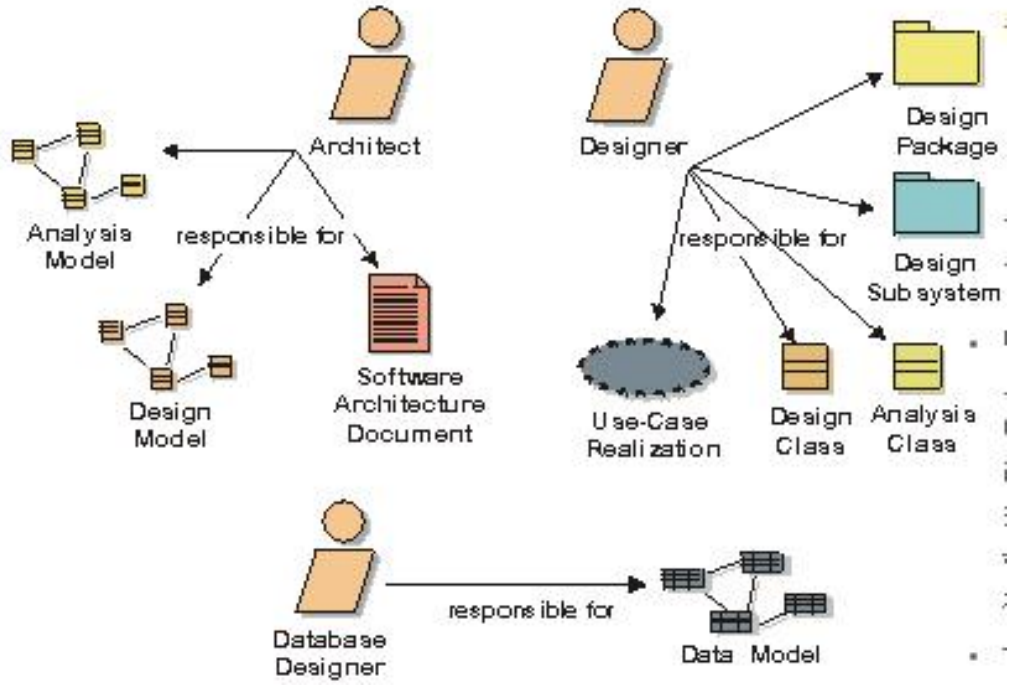
오늘날은 일반적인 테스트 개념은 잘못을 발견하는 작업이라기보다 측정하고 평가하는 측면이 더욱 커지고, 테스트는 개발과정 전체에 걸쳐서 발생하는 광범위한 작업, 즉 워크스루, 인스펙션, 각종리뷰 작업도 내포하고 있다.

2.2 테스팅 프로세스와 테스트문서

일반적으로 테스트 문서는 테스트 계획문서, 테스트 규격문서, 테스트 보고문서로 구성되며, 테스트 계획문서는 범위, 접근방법, 자원, 테스팅 활동을 기술하고, 테스트 규격문서는 테스트 설계・규격・절차 규격문서로 구성되고, 테스트 보고서는 테스트 항목 전달보고서, 테스트로그, 테스트 사건보고서 및 요약서 등으로 구성된다[6-7].

소프트웨어 분석과 설계 웍플로우의 작업자와 산출물의 관계는 <그림2-1>과 같이 테스트 계획문서는 프로젝트 문서를 참조하고, 테스트 케이스 규격문서와 절차규격문서는 테스트 설계 규격문서를 참조하여 작성한다.

ISSN: 2508—9099 JDAEM



[그림 2-1] 소프트웨어 분석 & 설계 웍플로우의 작업자와 산출물의 관계

[Fig. 2-1] The Relation of software analysis & design wookflow's worker and product

소프트웨어 테스트를 위한 매트릭스는 표2-1과 같이 테스트케이스와 관련된 매트릭스, 그리고 테스트 수행기록과 관련된 매트릭스로 구분한다.

[표 2-1] 소프트웨어 테스트를 위한 매트릭스

[Table 2-1] Matrix for software Test

테스트 정보 / 테스트 요소	테스트 케이스	테스트 수행 기록
시도한 테스트 케이스의 퍼센트	√	
수행한 테스트 케이스당 결함 수	√	
실패한 테스트 케이스 수	√	
성공률(SR)		√
불변의 실패 비율(PFR)		√
결함 주입 비율(DIR)		√
코드 완전성(CC)		√

3. 테스트 정보의 공유구조

3.1 테스트 정보의 공유

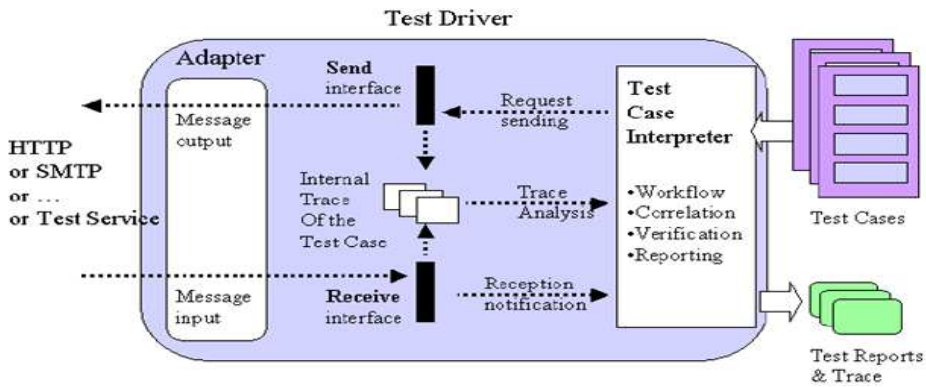
테스트 정보를 공유하기 위한 테스트 프레임워크 구성요소들을 정의하고, 공유하는 테스트 정보와 공유하지 않는 테스트 정보에 대해서 알아본다

1) 테스트 프레임워크 구성

테스트 프로임워크는 다른 테스트 정보를 결함될 수 있도록 컴포넌트타입으로 구성하며, 구성요소는 테스트 드라이브, 테스트 서비스, 테스트 케이스실행의 3가지 컴퍼넌트로 구성된다.

○ 텍스트 드라이버

텍스트 드라이버는 텍스트 케이스의 각 스템을 구동하는 컴포넌트로 주요기능은 텍스트 프레임워크 마크업언어로 작성된 테스트 케이스 정의를 좌싱하고 해석하는 역할을 한다.

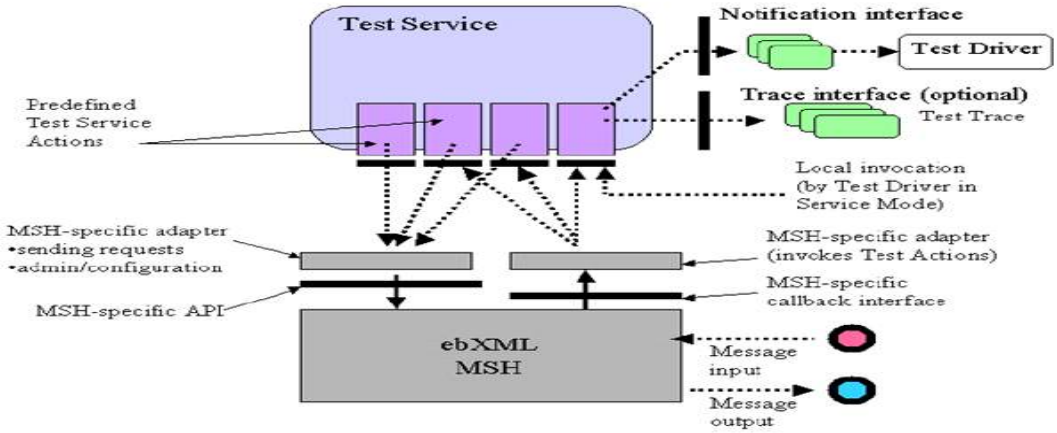


[그림 3-1] 테스트 드라이버 기능과 데이터 흐름

[Fig. 3-1] Test Driver function and Dataflow

○ 테스트 서비스

테스트 서비스는 테스트 케이스 수행을 위한 행위의 집합이며, 메시지 조정을 위함을 표현하고, MSH(Message Service Hondlder)로부터 메시지 콘텐츠와 에러 통보 메시지를 수신하고, 처리된 메시지를 다시 송신하는 역할을 수행한다.

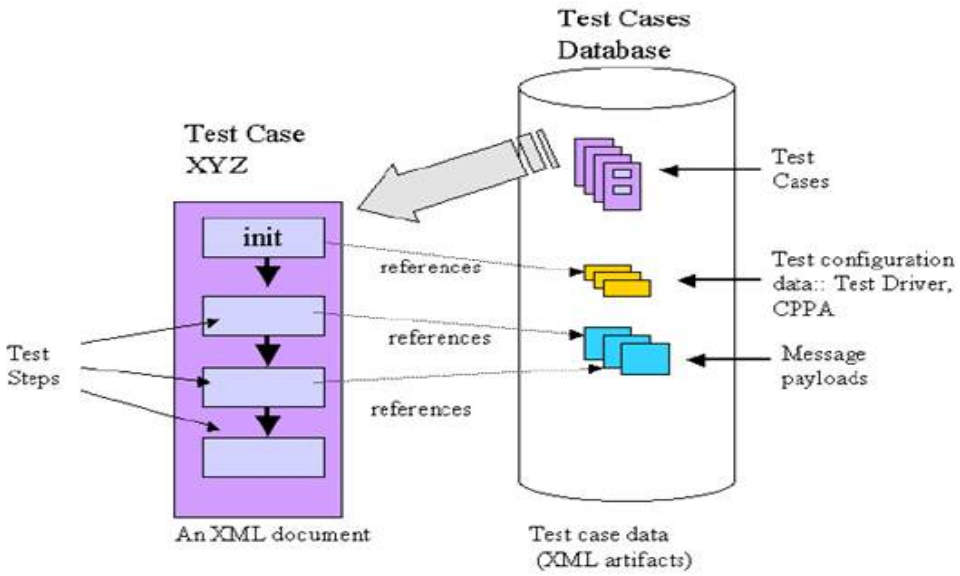


[그림 3-2] 테스트 서비스와 인터페이스

[Fig. 3-2] Test Service and Interface

○ 테스트 케이스 실행

테스트 케이스는 테스트 스텝으로 구성되고, 테스트 스텝은 하나의 컴포넌트가 수행되는 한 개 이상의 오퍼레이션 집합이며, 그림3-3은 테스트 케이스가 메시지 데이터를 참조하는 절차를 표시한 것이다.



[그림 3-3] 테스트 케이스 문서와 데이터베이스

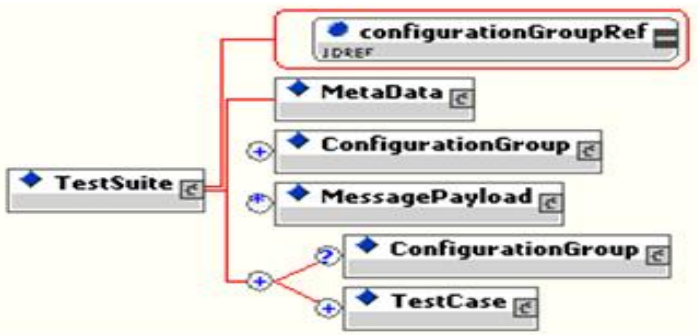
[Fig. 3-3] Testcase Document and Database

3.2 테스트 정보 통합

XML기반으로 테스트 정보를 통합하기 위해서 테스트 슈트문서, 테스트 케이스 구조, 테스트 스텝구조를 정의해야한다.

○ 테스트 슈트문서

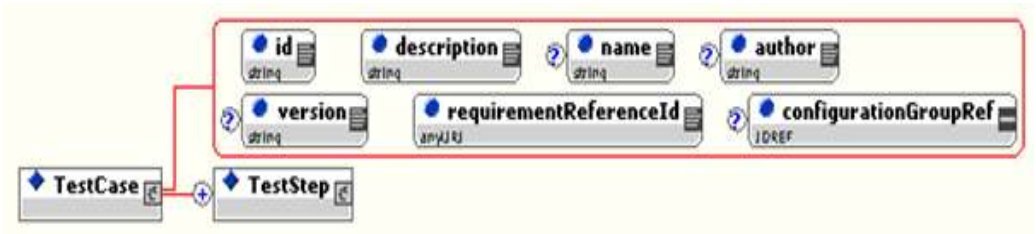
테스트 슈트문서는 테스트 데이터 드라이브의 메타데이터와 형상데이터, 문서 및 테스트드라이브에서 사용되는 문서로 구성되며, 자료구조는 다음 그림과 같다.



[그림 3-4] 테스트 슈트 문서의 데이터 구조도
[Fig. 3-4] Data Structure of Test suit Document

○ 테스트 케이스

테스트 케이스는 테스트 케이스에 관한 명세를 기술하는 영역으로 시스템을 유지보수하고, 테스팅하는데 중요한 키가된다.



[그림 3-5] 테스트 케이스의 데이터 구조도
[Fig. 3-5] Data Structure of Test Case

○ 테스트 스텝

테스트 스텝은 테스트에 관여되는 상황데이터를 관리·운영 하는데 정보를 수록하는 자료구조로, 구조도는 다음과 같다.

[그림 3-6] 테스트 스텝의 데이터 구조도

[Fig. 3-6] Data Structure of Test step

3.3 테스트 프레임 워크 구조

테스트 프레임워크는 테스팅을 하기 위한 3.2절에서 정의한데이터 구조를 기반으로 테스트 수행엔진, 테스트 라이브러리, 테스트 통합도구, 테스트 정보 저장소등으로 구성된다.

○ 테스트 수행엔진

테스트 케이스를 수행하고 수행결과를 모니터링 하는 역할을 수행한다.

○ 테스트 라이브러리

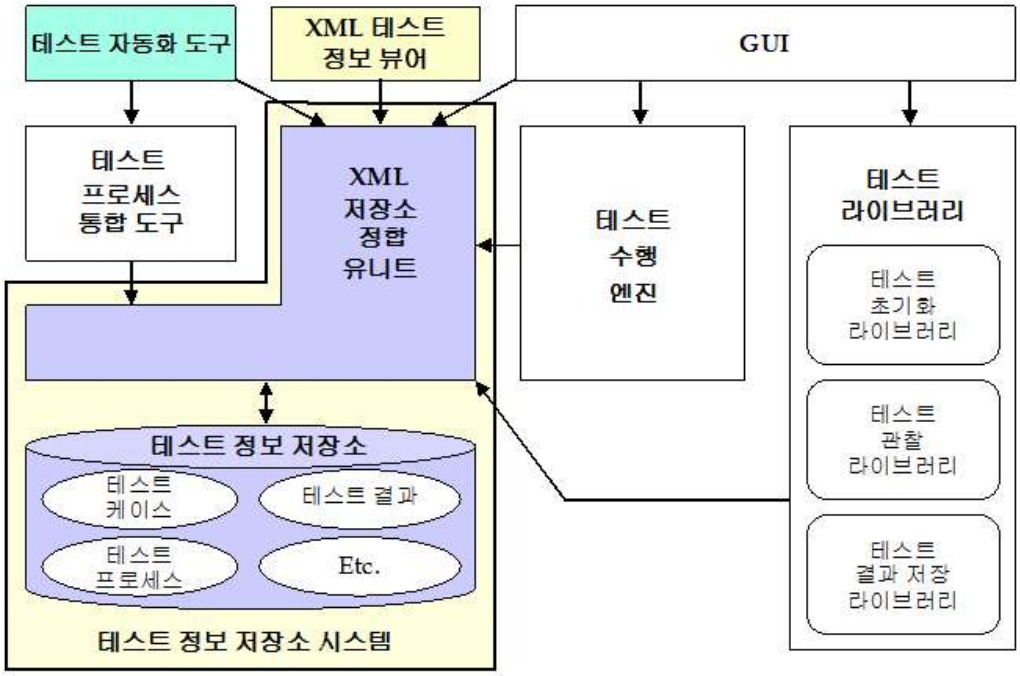
테스트 설계와 개발을 쉽게하기위한 내용을 제공한다.

○ 테스트 통합도구

품질평가내용과 단계 등 다양한 평가 프로세스를 수용할 수 있는 방법을 제공한다.

○ 테스트 정보 저장시스템

소프트웨어 테스트 과정에서 산출되는 결과물을 저장할 뿐만 아니라 향후 제작될 테스트 도구에서 생성된 결과물도 저장할 수 있는 유연성도 갖고 있다.



[그림 3-8] 테스트 자동화 프레임워크 구조

[Fig. 3-8] Structure of Test Automatic Framework

4. 실험 및 결과분석

본 논문에서는 제안된 테스팅 프레임워크의 유효성과 효과성, 그리고 성능과 재사용성을 분석하며, 실험 환경은 LG-IBMMultiNet 시스템이며 실험의 주요내용은 테스트요소와 리스크요소 매트릭스, 실험을 위한 접근제어와 권한부여에 관한 자료제공, 그리고 본 논문에서 중요한 정확성과 무결성 테스트 데이터와 성능테스트 데이터 등이다.

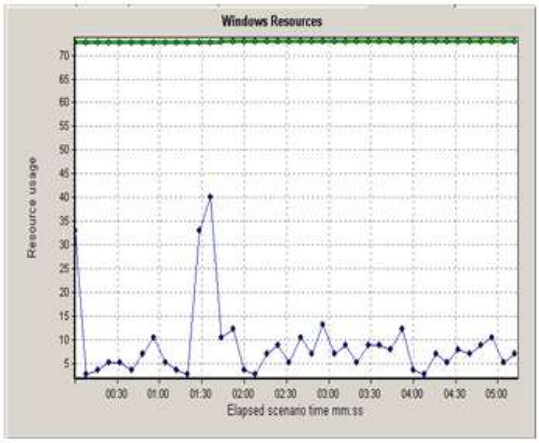
4.1 평가결과분석

실험용 웹소프트웨어를 3가지 평가도구와 제안된 방안을 평가한 결과 몇 가지를 지적할 수가 있다.

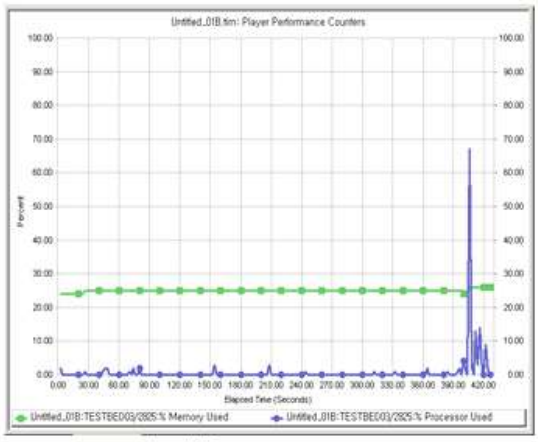
첫째, 접근제어와 권한부여의 측면에서 테스트 한 결과 안정성이 있다고 볼 수 있으며

둘째, 정확성과 무결성, 그리고 보안성은 별견해차이가 없고 잘 진행되었음을 알 수가 있다.

셋째, 성능평가 결과 본 논문에 제안한 테스팅방식이 cpu시간 20%이상 처리속도가 줄어들어 효율성이 증가되었고, 메모리 점유율도 평균 10%이상 상승하였으며, 이를 종합하면 다음과 같다.

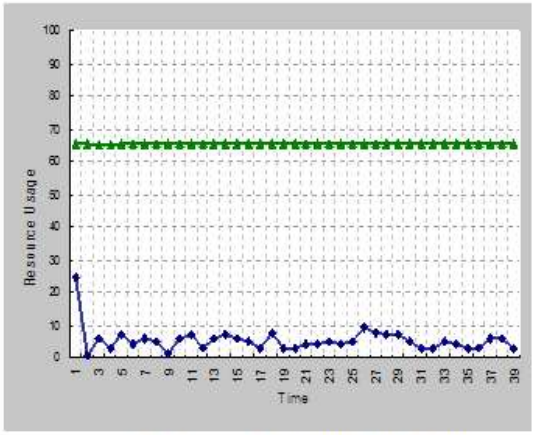


I 사 R 제품 : S/W1 (CPU 0.92, MEM 7.28)



C 사 A 제품 : S/W1 (CPU 0.67, MEM25)

B 사 C 제품 : S/W1 (CPU 9.5, MEM 29)



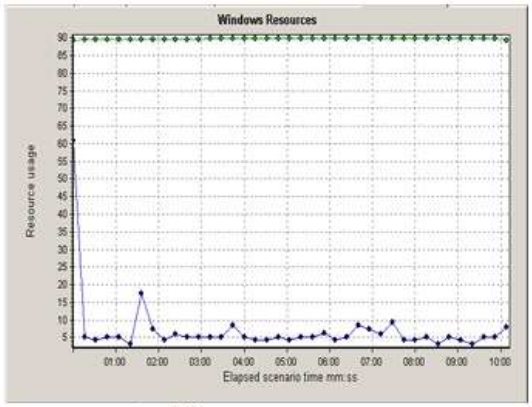
통합 환경 : S/W1 (CPU 0.54, MEM 6.55)

[그림 4-1] 실험용 소프트웨어 S/W1 성능 평가 데이터

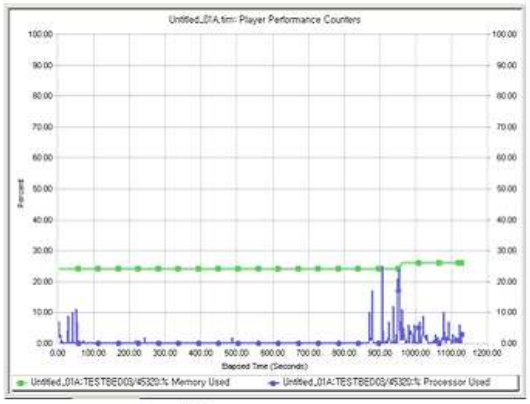
[Fig. 4-1] Efficiency Evolution Data of Testing software S/W1

2) 실험용 소프트웨어 2개를 조합한 성능 평가 데이터

세 개의 실험도구 및 통합 환경 프레임워크를 사용하여 실험용 소프트웨어 S/W1과 S/W2를 조합하여 동시에 성능을 평가한 데이터는 그림 4-2와 같다.

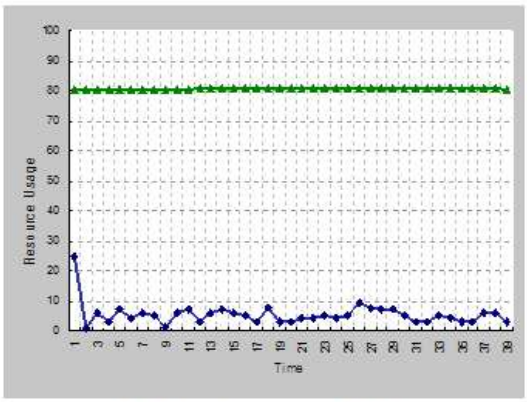


I 사 R 제품 : S/W2 (CPU 0.72, MEM 8.97)



C 사 A 제품 : S/W2 (CPU 0.67, MEM24.3)

B 사 C 제품 : S/W2 (CPU 14.5, MEM 29)

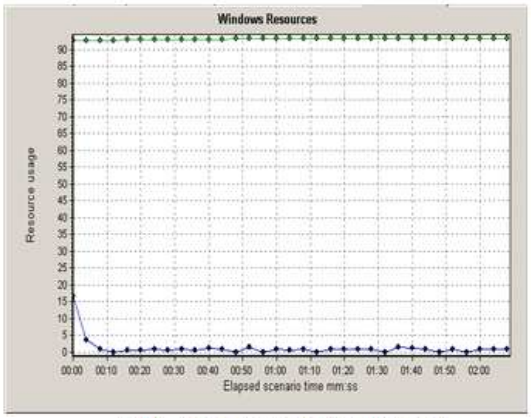


통합 환경 : S/W2 (CPU 0.54, MEM 8.07)

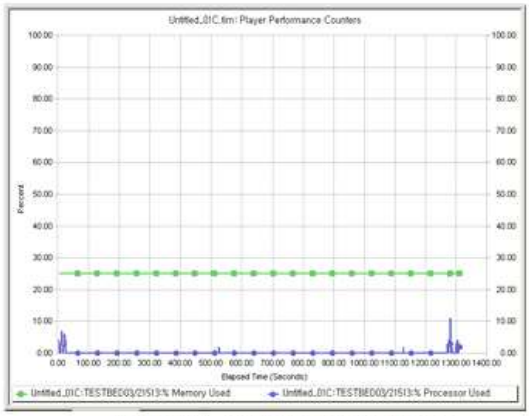
[그림 4-2] 실험용 소프트웨어 S/W2 성능 평가 데이터
[Fig. 4-2] Efficiency Evolution Data of Testing software S/W2

3) 실험용 소프트웨어 3개를 동시 수행한 성능 평가 데이터

세 개의 실험도구 및 통합 환경 프레임워크를 사용하여 실험용 소프트웨어 S/W1, S/W2, S/W3 모두를 동시에 성능 평가한 데이터는 그림 4-3과 같다.

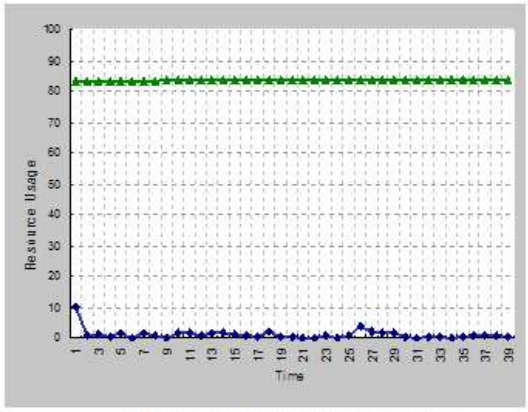


I 사 R 제품 : S/W3 (CPU 1.3, MEM 9.31)



C 사 A 제품 : S/W3 (CPU 0.12, MEM 25)

B 사 C 제품 : S/W3 (CPU 3, MEM 29)



통합 환경 : S/W3 (CPU 0.1, MEM 8.38)

[그림 4-3] 실험용 소프트웨어 3개를 동시 수행한 성능평가 데이터

[Fig. 4-3] Efficiency Evolution Data of concurrent Excution for 3 Test software

종합하면, 그림4-3에 의하면, 통합 환경의 프레임워크 상에서 CPU 시간은 20%이상 처리 속도가 줄어들어 처리 효율이 증가되었고, 메모리 점유율도 평균10%로 줄어든 것을 알 수가 있다.

5. 결론

소프트웨어 테스트 자동화 도구를 이용하여 테스트 프로세스의 전부 또는 일부를 자동화 처리함으로써 테스트 시간의 단축과 테스트 비용을 줄일 수 있다. 그러나 하나의 테스트 자동화 도구를 이용하여 이질적인 컴퓨팅 환경에서 다양한 종류의 테스트 요구 사항을 모두 충족시킨다는

것은 현실적으로 거의 불가능하다.

본 논문에서는 소프트웨어 테스팅에 소요되는 비용을 감소시키기 위하여 테스트 자동화 도구들의 통합화 대신에 테스트 과정에서 생성된 테스트 정보를 다른 테스트 도구에서도 접근, 공용할 수 있도록 하는 테스트 프레임워크를 제안하였다. 현재 사용화 된 테스트 자동화 도구들은 상호 호환성을 고려하지 않고 개발되었기 때문에 특히, 테스트 계획수립 단계에서 생성되는 테스트 케이스의 공유와 재사용이 거의 불가능하다.

본 논문에서는 테스트 설계 과정에서 생성된 테스트 케이스 정보를 다른 테스트 도구들도 공유할 수 있도록 테스트 정보 저장소를 테스트 자동화 프레임워크 상에서 구축하였다. 세 종류의 테스트 도구와 본 논문에서 제안한 통합 환경의 테스트 자동화 프레임워크를 실험한 결과 CPU 시간은 20% 이상 처리 속도가 줄어들어 처리 효율이 증가되었고, 메모리 점유율도 평균 10%로 줄어든 것을 확인하였다. 이는 통합 환경의 테스트 자동화 프레임워크 상에서 테스트 케이스를 재사용함으로써 시스템 성능인 시스템 메모리 점유율과 처리 효율을 향상시켰다고 볼 수가 있다. 향후 연구로는, 테스트 프레임워크를 일반화 시켜 여러 장르에 적용하고 평가할 수 있는 도구를 설계 구현하는데 있다.

ISSN: 2508—9099 JDAEM

References

[1] G. J. Myers, The Art of Software Testing, John Willy & Sons, (1979).

[2] IEEE Standard Glossary of Software Engineering Terms, IEEE Society Press, Addison Wesley, (1993).

[3] B. Beizer, Software Testing Techniques, 2nd Ed., International Thomson Computer Press, (1990).

[4] E. Dustin, J. Rashka, and J. Rashka, and J. Paul, Automated Software Testing, Addison Wesley, (1999).

[5] M. Fewster, D. Graham, Software Test Automation: Effective use of test execution tools, ACM Press, Addison Wesley, (1999).

[6] ISO/IEC 9126, Software engineering – Product quality – Part 1: Quality model, ISO, (2001).

[7] The Open Group, TETware white Paper, March (2003).