ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

소프트웨어 테스트 모니터링 프레임워크 구축 방안☆

Construction Method of Software Test Monitoring Framework

서 용 진[1] 김 수 지[2] 김 현 수[1*]
Yongjin Seo Su Ji Kim Hyeon Soo Kim

요 약

테스팅은 시스템의 요구사항을 바탕으로 테스트 케이스를 생성하여 소프트웨어에 내재되어 있는 결함을 발견하는 활동이다. 테스팅을 효과적으로 수행하기 위해서는 충실한 테스트 계획, 잘 작성된 테스트 케이스 생성과 더불어 체계적인 테스트 모니터링 활동이 요구된다. 테스트 자동화 방법에 대한 대부분의 연구들은 테스트 케이스 생성에서 테스트 실행까지의 자동화 방법에 초점이 맞춰져 있다. 본 연구에서는 이와 달리 테스트 모니터링의 자동화 방안에 대하여 연구한다. 이를 위해 테스트 모니터링 자동화를 위해 해결해야 할 요소를 도출하고 이를 기반으로 테스트 모니터링 자동화 프레임워크의 구축 방안을 제시한다.

☞ 주제어 : 소프트웨어 테스팅, 소프트웨어 테스트 모니터링, 테스트 모니터링 프레임워크

ABSTRACT

Software testing is an activity to find defects included in software through creating test cases from the software system specification. In order to perform software testing effectively, it is required to prepare the full test plan, to create well-defined test cases, and to execute test monitoring activities systematically. Most existing researches for the test approaches focus on automating the activities from the test cases generation to the test execution. Contrary to those approaches, we study automatic approaches for test monitoring activities. For this, we identify the research issues that should be solved to automate test monitoring activities. Next, with those solutions, we suggest the construction method for an automatic framework for test monitoring.

☞ keyword : Software Testing, Software Test Monitoring, Test Monitoring Framework

1. 서 론

테스팅은 시스템의 요구사항을 바탕으로 테스트 케이스를 생성하여 소프트웨어에 내재되어 있는 결함을 발견하는 활동이다[1,2]. 테스팅을 효과적으로 수행하기 위해서는 첫 번째로 테스트 계획을 잘 세우고, 충실한 테스트 케이스를 생성하여야 한다. 즉 테스트 수행 목표, 수행 대상, 수행 방식, 수행 환경, 절차 등에 대한 명확한 계획과 이러한 계획을 충실하게 이행할 수 있는 테스트 케이스가 개발되어야 한다. 두 번째로, 작성한 테스트 계획에 맞게 테스트가 잘 수행되고 있는지 모니터링 하여야한다. 즉 테스트 계획에 명시된 요구사항을 잘 만족하는지, 테스트 계획의 예상 결과나 완료 시점과 실제 테스트 활동의 결과와 시점이 얼마나 일치하는지 확인하여야 한다[3,4].

기존의 테스팅과 관련된 많은 연구에서 테스트 자동화 방법을 제시하지만 대부분은 테스트 케이스 생성에서 테스트 실행까지의 자동화 방법에 초점이 맞춰져 있다. 하지만 본 연구에서는 관점을 바꾸어 테스트 모니터링의 자동화 방안에 대하여 고찰하고자 한다. 테스트 모니터링의 자동화를 위해서는 테스트 활동을 관찰할 수 있어야 한다. 사용자의 개입 없이 테스트 활동을 관찰하기 위해서는 테스트 활동에서 생성되는 여러 산출물을 테스트 모니터링 시스템에서 처리할 수 있어야 한다. 그렇지만 일반적으로 테스트 활동의 산출물들은 컴퓨터에서 읽어들여 처리할 수 있는 형태가 아니었기 때문에 테스트 활동에 참여하는 테스터나 개발자들이 수작업으로 입력한 테스트 활동의 진행 정보를 바탕으로 테스트 모니터링 작업이 수행되었다[5]. 본 연구에서는 테스트 모니터링

[1] Department of Computer Science & Engineering, Chungnam National University, Daejeon, 34134, Korea.
[2] Research Institute, DNI Cooperation, Daejeon, 35349, Korea.
* Corresponding author (hskim401@cnu.ac.kr)
[Received 23 August 2016, Reviewed 29 August 2016, Accepted 27 September 2016]
☆ 이 논문은 충남대학교 석사학위논문 '소프트웨어 테스트 활동 모델링 및 테스트 모니터링 도구 개발[5]'을 재구성 및 확장한 것임.
☆ 이 연구는 충남대학교 학술연구비에 의해 지원되었음

자동화를 위해 해결해야 할 요소와 이를 기반으로 테스트 모니터링 자동화 프레임워크의 구축 방안을 제시하고자 한다.

본 논문은 다음과 같이 구성된다. 2장에서는 테스트 모니터링에 대해 언급한 문헌과 기존 테스트 모니터링 시스템의 고찰을 통해 기존 시스템들의 문제점을 언급한다. 3장에서는 테스트 모니터링 프레임워크를 구축하기 위해 고려해야 할 이슈와 해결 방안으로 테스트 모니터링 자동화를 위한 구체적인 전략을 기술한다. 4장에서는 테스트 모니터링 자동화 프레임워크에 대한 구조 및 구현 내용을 설명한다. 5장에서는 기존 테스트 모니터링 시스템과의 정성적인 비교를 통해 논문에서 제시한 테스트 모니터링 프레임워크의 능력을 평가한다. 마지막으로 6장에서 결론을 맺는다.

2. 관련 연구

테스트 모니터링 활동은 실제 테스트 수행 과정과 내용이 테스트 계획에 비추어 일치하는지 관찰하는 과정이다[3, 4]. [3]에서는 테스트 모니터링 활동과 관련된 이슈들을 제시하고 있고, 테스트 모니터링 결과를 문서화할 때 문서에 포함해야 할 내용들을 언급하고 있다. 여기서는 모니터링 문서의 양식을 스프레드시트(spreadsheet) 형태로 제시하고 있는데 포함되는 내용에는 개발된 테스트 케이스의 수, 테스트 실행의 수, 테스트된 모듈의 비율, 테스트된 시스템 기능의 수, 각 기능의 테스트 커버리지 등이 있다. 이를 통해 알 수 있는 것은 여기서는 테스트 모니터링을 주로 테스트 수행의 성능 관점에서만 바라보고 있다는 것이다.

[4]에서는 테스트 모니터링 활동을 위해 사용되는 테스트 메트릭을 제시하고 있다. 이런 메트릭에는 계획 대비 준비된 테스트 케이스의 비율, 계획된 작업이나 노력 대비 준비된 테스트 환경의 비율, 실행된/실행되지 않은 테스트 케이스의 수, 성공한/실패한 테스트 케이스의 수, 결함 정보(결함 밀도, 실패 율 등), 테스트 마일스톤에서의 일정 등이 포함된다. 이를 통해 [4]에서는 [3]보다 좀 더 넓게 다양한 테스트 활동을 모니터링 대상으로 파악하고 이를 모니터링하기 위해 여러 종류의 테스트 메트릭을 사용하고 있다는 것을 알 수 있다.

[6]에서는 테스트 모니터링을 해야 하는 이유, 테스트 모니터링 대상, 테스트 모니터링의 절차 등을 간략하게 소개하고 있다.

대부분의 테스트 지원 도구들은 간단한 형태로나마 테스트 모니터링 기능을 지원하고 있다. QAComplete[7]와 같은 테스트 도구들은 대체로 테스트 실행의 자동화, 테스트 케이스 생성의 자동화에 초점이 맞춰져 있다 보니 지원되는 테스트 모니터링 기능은 상당히 적은 편이다.

RQM(Rational Quality Manager)은 웹 기반의 협업 도구로서 소프트웨어 개발 주기 동안 광범위한 테스트 계획, 테스트 생성 및 테스트 산출물 관리 기능을 제공한다[8]. 사용자가 입력한 테스트 계획을 기반으로 테스트 프로젝트의 모든 단계에서 테스트를 수행하는 팀의 활동을 지원한다. 그러나 RQM은 사용자가 직접 테스트 프로젝트를 생성하고 처음 테스트를 수행할 때 테스트의 시작, 종료 조건, 테스트 케이스, 테스트 자원 등을 모두 수동으로 입력해야하며 테스트 결과 또한 사용자가 입력해야한다.

NEXCORE Test Manager는 소프트웨어 프로젝트 전략에 맞게 테스트를 계획, 설계, 수행할 수 있는 환경을 제공한다[9]. 또한 테스트 수행 과정에서 발생한 결함을 개발자에게 할당하고, 결함의 원인을 분석 및 진척 사항을 상시 모니터링 할 수 있는 환경을 제공한다. 이 도구는 테스터가 도구에 테스트 시나리오와 케이스를 등록하고, 테스트 관리자가 등록된 테스트 케이스를 수행할 담당 테스터에게 할당한다. 또한 테스터는 테스트 수행 결과를 직접 기록해야하고 결함을 발견하여 등록해야 한다.

즉, 기존의 테스트 지원 도구들은 테스터나 개발자를 통해 테스트 모니터링을 위해 필요한 대부분의 정보를 수동으로 입력하여야 동작할 수 있다. 테스트 스케줄 및 테스트 자원과 관련된 테스트 계획뿐만 아니라 진척도 파악 및 테스트 성능 분석에 필요한 테스트 수행 결과 역시 수동으로 입력하여야 한다. 이와 같은 상황은 테스트 모니터링의 자동화를 달성하는데 걸림돌이 된다.

이에 본 논문에서 제안하는 테스트 모니터링 프레임워크는 사용자로부터 문서 형태의 테스트 계획을 수동으로 입력받지 않고, 사용자가 UI를 통해 테스트 계획을 작성하면 XML 형태의 테스트 계획으로 저장되어 컴퓨터가 자동으로 읽어 들인다. 읽어 들인 테스트 계획의 정보를 이용하여 테스트 스케줄 정의, 테스트 자원 할당 및 테스트 진행 상태 모니터링을 준비할 수 있다. 또한 자동으로 테스트 결과의 로그 정보를 분석하여 진척도 파악 및 테스트 성능 분석을 수행할 수 있다.

3. 테스트 모니터링 프레임워크 구축

테스트 모니터링 작업을 자동으로 수행하기 위해서는 자동화된 프레임워크를 구축하여야 한다. 자동화된 테스트 모니터링 프레임워크를 구축하기 위해서는 다음과 같은 이슈를 고려해야 한다.

1) **모니터링 대상**: 다양하게 수행되는 테스트 활동 중에서 테스터나 관리자와 같이 테스트 활동에 대한 이해관계자가 어떤 정보나 내용에 관심 있는지 파악하여야 한다. 그래야만 그들에게 의미 있는 정보를 제공해 줄 수 있기 때문이다. 따라서 어떤 정보를 모니터링 할 것인가에 대해 파악하여야 한다.
2) **모니터링의 자동화**: 간단한 테스트 활동은 테스터가 수작업으로 자료를 정리하여 테스트 활동의 진행 상황을 파악할 수 있다. 그러나 대규모 프로젝트의 경우에는 테스트 활동이 복잡해지고 다수의 테스터가 관여하는 상황이 자연스럽게 나타나게 된다. 이런 상황에서 수작업으로 자료를 수집하여 테스트 활동을 모니터링 한다는 것은 매우 어렵다. 이런 상황을 해결하기 위해서는 모니터링 활동을 자동화하는 것이 절대적으로 필요하다. 모니터링을 자동으로 수행하기 위해서는 각 모니터링 활동의 세부적인 절차나 방법이 정의되어야 하고, 그 방법이 컴퓨터에서 실행 가능하여야 한다.

3.1 테스트 모니터링 대상 파악

테스트 모니터링 활동은 테스트 계획에 비추어 실제 테스트 수행 과정과 내용을 관찰하는 과정이다. 의미 있는 테스트 모니터링 수행하기 위해서는 사용자가 필요로 하는 테스트 모니터링 대상을 파악하여야 한다. 각 조직마다 테스트의 진행 상황에 대해 필요로 하는 정보가 다를 수 있기 때문에 본 연구에서는 보편적으로 대부분의 조직에서 필요로 하는 정보를 파악하려 하였다. 이를 위해 테스트 모니터링을 언급한 문헌[3,4,6]과 기존 테스트 모니터링 도구[8,9]뿐만 아니라 TMMI (Test Maturity Model Integration)[10,11]의 내용도 분석하였다. 이런 과정을 통해 도출된 테스트 모니터링 대상은 다음과 같다[12].

1) **테스트 진행 상황**: 테스트 스케줄 관점에서 테스트 진행 상황을 모니터링하기 위해 필요하다. 테스트 계획에 기술된 스케줄 대비 실제 수행된 스케줄을 비교하여 테스트 작업의 진행 상황을 파악할 수 있다.
2) **테스트 환경 자원 제공 및 사용 상황**: 제공된 테스트 환경 및 자원 관점에서 테스트 계획에서 제공하기로 하였던 테스트 환경 및 자원이 올바르게 제공되고 사용되는지 확인하기 위함이다.
3) **이해관계자의 참여 상황**: 테스트에 참여하는 이해관계자에 대한 정보를 나타낸다. 이를 통해 테스트 계획에 기술된 테스트 참여자가 실제 테스트 수행에 참여하는지 확인할 수 있다.
4) **테스트 단계 진척 상황**: 현재 진행 중인 테스트의 단계를 나타낸다. 테스트 계획에 기술된 테스트 단계 중 현재 진행 중인 테스트 단계가 어느 단계에 위치하는지 확인이 가능하다.
5) **테스트 시작/종료 시점 확인**: 수행하는 테스트의 시작 시점과 종료 시점을 파악하기 위한 목적으로, 테스트 계획에 기술된 테스트 스케줄에 따라 테스터에게 테스트 수행 시작 시점과 종료 시점을 알려준다.
6) **테스트 성능 기록**: 수행중인 테스트에 대해 어느 정도의 테스트 케이스를 커버했는지, 성공 및 실패한 테스트 케이스에 대한 정보 등을 나타내어 테스트 성능을 확인할 수 있다.

3.2 테스트 모니터링 자동화

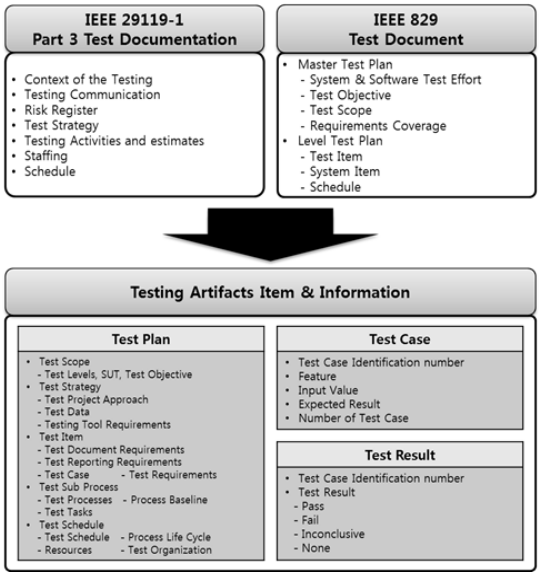
자동화 기반의 테스트 모니터링 프레임워크를 구축하기 위해서는 무엇보다도 테스트 활동의 여러 산출물들이 사용자의 개입 없이 모니터링 시스템에 의해 처리될 수 있어야 한다. 기존의 테스트 계획은 대부분 자연어로 기술되어 컴퓨터가 테스트 계획의 정보를 자동으로 인식할 수 없었다. 또한 테스트 진행 상황에 대한 정보도 대부분 테스터가 입력한 정보에 의존할 수밖에 없었다. 이런 상황에서는 테스트 모니터링을 수행하기 위해 사용자의 개입이 많이 요구된다는 문제가 있다. 이 절에서는 테스트 모니터링 자동화를 위해 해결해야 할 요소와 테스트 모니터링 자동화 프레임워크 구축 방안을 제시한다.

3.2.1 테스트 계획 기술 언어

기존의 테스트 계획은 자연어 형태로 기술되어 컴퓨터가 읽어 들일 수 없어서 테스트 모니터링 과정에 필요

한 테스트 계획 정보는 사용자가 일일이 입력해야 하는 단점이 있었다. 본 논문에서는 먼저 테스트 계획을 컴퓨터가 인식할 수 있도록 테스트 계획 기술 언어를 구축한다.

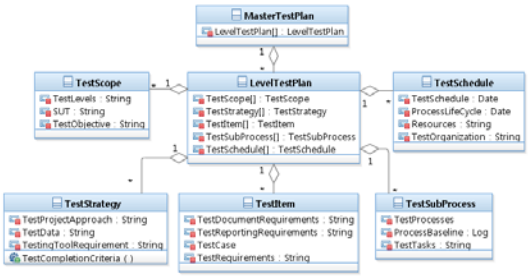
테스트 계획 기술 언어를 구축하기 위해서는 먼저, 테스트 관련 표준문서를 분석하여 테스트 계획 문서를 정형화하기 위한 항목을 도출하여야 한다. 테스트 관련 표준 문서를 분석한 결과 테스트 계획에는 테스트 목적, 실제 적용되는 테스트 범위, 테스트 전략, 테스트 대상, 테스트 스케줄, 테스트 자원, 테스트 프로세스 등의 내용이 적어도 포함되어야 한다.



(그림 1) 테스트 활동 산출물 항목 및 포함되는 정보 도출
(Figure 1) Induction of Testing Artifacts Item & Information

그림 1은 소프트웨어 테스트와 관련된 표준문서 IEEE Std 29119-3-2013[13]과 IEEE Std 829-2008[14]에서 나타나는 정보들을 바탕으로 테스트 활동 산출물의 항목과 정보들을 도출하는 과정을 보여주고 있다. IEEE Std 29119-3-2013 표준문서에서는 소프트웨어 테스트 문서에 포함되는 테스트 계획, 테스트 수행, 테스트 결과가 어떠한 정보를 갖고 있어야 하는지 나열하고 있고, IEEE Std 829-2008 표준문서에서는 테스트 계획을 중심으로, 크게 MTP (Master Test Plan)와 LTP (Level Test Plan)로 나누어 각 항목에 기재되어야 하는 정보를 나타내고 있다. 이 두 가지 표준문서의 정보를 바탕으로 본 논문에서는 테스트 활동 산출물의 항목과 포함되어야 할 정보를 도출하였다. 이렇게 도출된 항목과 정보를 바탕으로 테스트 계획을 기술하기 위한 정보 모델은 그림 2와 같이 구축할 수 있다.

테스트 계획을 정형화한 다음, 테스트 계획을 입력받기 위하여 사용자 인터페이스를 제공하는 방식을 택하였다. 이렇게 함으로써 자연어로 기술해야 했던 테스트 계획을 UI를 통해 간편하게 입력할 수 있으며, 또한 입력된 테스트 계획을 XML로 변환하여 저장함으로써 컴퓨터가 읽을 수 있는 형태의 테스트 계획을 생성할 수 있다. XML 형태로 생성된 테스트 계획은 컴퓨터가 인식할 수 있어서 테스트 모니터링 활동에 필요한 정보를 자동으로 추출할 수 있게 된다[15].



(그림 2) 테스트 계획 정보 모델
(Figure 2) Information Model for Test Plan

3.2.2 테스트 모니터링을 위한 정보 추출

테스트 모니터링을 자동화하기 위해서는 테스트 모니터링 활동을 자동으로 수행하기 위한 방법과 이를 지원하기 위한 정보가 필요하다.

먼저 테스트 모니터링 활동을 자동으로 수행하기 위해 필요한 정보는 표 1과 같이 정리할 수 있다. 이런 정보는 표 1의 Test Plan, Test Cases, Test Results 항목 등으로부터 얻을 수 있다.

Test Plan의 Test Scope의 항목들을 통해 Test Target에 대한 정보를 도출한다. 이를 통해 수행되는 테스트 대상이 무엇인지, 사용자가 올바른 테스트 대상을 테스트하고 있는지 확인할 수 있다. Test Item의 Test Requirements를 통해 테스트 계획에 기술된 요구사항을 현재 수행하는 테스트가 만족하는지 확인할 수 있다. Test Strategy의 항목들을 통해 Test Approach와 Test Coverage에 대한 정보를 파악하여 어떤 테스트 방법이 적용되고, 수행하는 테스트에 대한 커버리지 측정에 대한 정보를 확인할 수

(표 1) 테스트 모니터링에 요구되는 정보
(Table 1) Required Information for Test Monitoring

Source		Item	Information for Test Monitoring
Test Plan	Test Scope	• Test Levels • SUT • Test Objective	• Test Target
	Test Item	• Test Requirements	• Test Requirements
	Test Strategy	• Test Project Approach • Test Data • Testing Tool Requirement	• Test Approach • Test Coverage
	Schedule	• Test Schedule • Process Life-Cycle • Resources • Test Organization	• Test Schedule • Process Life-Cycle • Tester • Test Organization
	Environment	• Test Environment • Resources	• Test Environment • Resources
Test Cases		• Test Case	• Test Cases List
Test Results		• Test Result (Pass/Fail/Inconclusive)	• Test Result (Pass/Fail/Inconclusive)

있다. Schedule 항목을 통해 테스트를 언제부터 언제까지 수행하는지에 대한 정보를 확인할 수 있으며 어떤 이해관계자 및 조직이 테스트에 참여하는지 나타낼 수 있다. Environment 항목은 테스트를 수행할 때 제공되는 환경 및 자원을 나타내고 있으며 이를 통해 사용자는 테스트 계획에 기술된 테스트 환경에 따라 테스트를 수행하고 있는지, 자원이 올바르게 제공되고 있는지 확인할 수 있다. Test Cases의 생성된 Test cases list 정보와 Test Results의 Pass/Fail/Inconclusive 정보를 통해 테스트 단계 진척도와 테스트 성능을 파악할 수 있다.

이런 모든 정보들은 XML 형태로 저장되므로 테스트 모니터링 프레임워크는 이 정보들을 읽어 들여 테스트 활동의 모니터링에 필요한 기능을 수행한 다음, 결과를 시각화하여 사용자에게 제공한다.

3.2.3 테스트 모니터링 기능 실현

테스트 모니터링의 자동화를 실현하기 위해서는 테스트 활동을 모니터링하기 위한 기능을 구현하여야 한다. 3.1절에서 언급한 6가지의 테스트 모니터링 대상에 대하여 그것을 관찰할 수 있도록 각각 기능으로 구현한다. 본 연구에서는 6개의 모니터링 대상에 대한 기능을 구현하였으나 지면 관계상 여기서는 테스트 성능의 모니터링에 대한 기능 구현을 기술한다. 테스트 성능 모니터링은 다음과 같은 절차로 수행되고 그 결과가 사용자에게 시각적으로 제공된다.

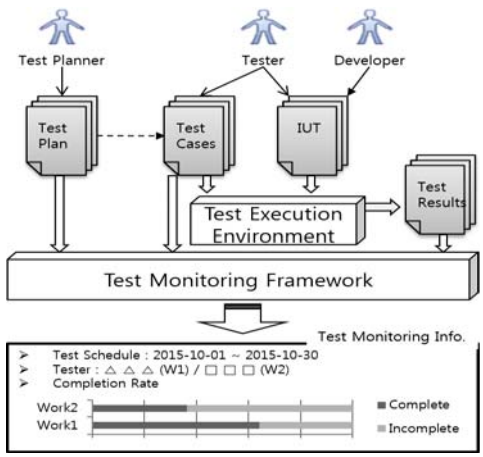
① 테스트 케이스 정보로부터 전체 테스트 케이스의 리스트 정보를 획득한다.
② 테스트 케이스 리스트 정보로부터 전체 테스트 케이스 수를 파악한다.
③ 테스트 결과 파일로부터 테스트 결과가 없는 것(아직 수행하지 않은 테스트 케이스)을 파악한다.
④ 전체 테스트 케이스 수에서 아직 수행하지 않은 테스트 케이스의 수를 차감하여 테스트 수행을 완료한 테스트 케이스 수를 획득한다.
⑤ 테스트 결과 파일로부터 테스트 결과가 성공/실패/보류 (Pass/Fail/Inconclusive)인 테스트 케이스와 그 수를 파악한다.
⑥ 테스트 성공률 = (테스트 결과가 성공인 테스트 케이스 개수)/(테스트 완료 테스트 케이스의 개수)
⑦ 테스트 실패율 = (테스트 결과가 실패인 테스트 케이스 개수)/(테스트 완료 테스트 케이스의 개수)
⑧ 테스트 보류율 = (테스트 결과가 보류인 테스트 케이스 개수)/(테스트 완료 테스트 케이스의 개수)

⑨ ② ~ ⑧의 결과를 이용하여 텍스트 정보와 함께 파이 차트를 생성한 다음 사용자에게 제공한다.

3.2.4 테스트 모니터링 프레임워크의 동작

본 논문에서 제안하는 테스트 모니터링 프레임워크는 그림 3과 같이 표현된다. 먼저 테스트 설계자 (Test Planner)가 UI를 통해 테스트 계획을 채우면 (항목에 따라

버튼으로 해당 내용을 선택하는 경우, 텍스트 형태로 입력하는 경우 등이 있음) 시스템은 이것을 XML 형태의 테스트 계획으로 저장한다. 테스트 모니터링 프레임워크는 XML로 저장된 테스트 계획을 파싱하여 테스트 대상, 요구사항, 테스트 커버리지, 테스트 스케줄, 테스터, 테스트 환경 및 자원 등의 정보를 읽어 들인다. 이를 통해 테스트 모니터링 프레임워크의 기능 중 테스트 환경 자원 제공 및 사용에 대한 모니터링, 이해관계자의 참여에 대한 모니터링, 테스트 시작/종료 시점 확인이 가능하다. 다음으로 테스터에 의해 개발된 테스트 케이스에 대한 정보를 읽어 들인다. 계속해서, 테스터 또는 개발자가 준비된 테스트 케이스를 이용하여 테스트를 수행하게 되면 테스트 실행 환경은 테스트 결과 파일을 생성하게 된다. 테스트 케이스에 대한 정보로부터 적용하는 테스트 케이스 목록과 테스트 케이스의 총 개수, 테스트 결과 파일로부터 성공/실패/보류 등 테스트 수행 결과 정보를 획득하게 되면 테스트 단계 진척도, 진행 상황, 성능 등의 분석을 수행한다.

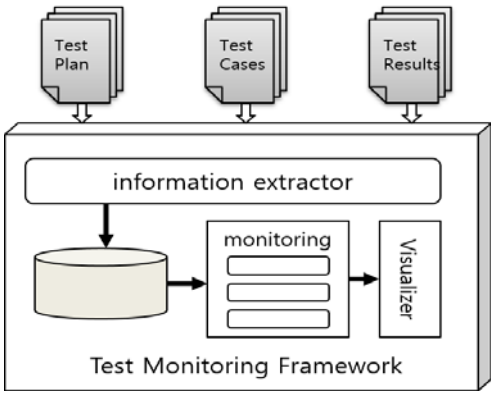


(그림 3) 테스트 모니터링 프레임워크 개념도
(Figure 3) Schematic of Test Monitoring Framework

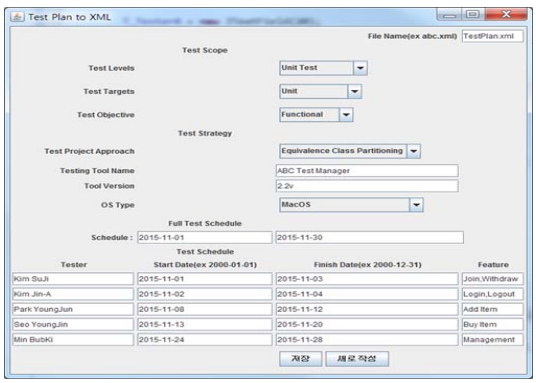
4. 테스트 모니터링 프레임워크의 설계 및 구현

4.1 테스트 모니터링 프레임워크의 구조

테스트 모니터링을 자동으로 수행하기 위한 프레임워크의 구조는 그림 4와 같다. 그림에서 정보추출기는 각각의 입력 문서로부터 모니터링을 수행하기 위해 필요한 정보를 추출하는 기능을 수행한다. 예를 들어, 테스트 계획으로부터는 전체 테스트 스케줄, 테스트 활동에 참여하는 테스터, 그들의 일정 및 담당 역할 등의 정보를 추출하고 테스트 케이스로부터는 전체 테스트 케이스 목록의 정보를, 테스트 결과로부터는 성공 또는 실패한 테스트 결과의 정보 등을 추출한다. 이렇게 추출된 정보는 파일로 저장된다. 모니터링 기능은 3.1 절에서 언급한 모니터링 대상을 관찰하기 위해 각각을 기능으로 구현한 모듈들을 의미한다. 각 기능들은 기능마다 필요한 정보를 파일로부터 읽어 와서 기능의 내부 절차에 따라 적절한 처리 및 연산을 수행한 다음 그 결과를 시각화 모듈에게 넘겨준다. 시각화 모듈은 사용자들이 테스트 모니터링 결과를 쉽게 파악할 수 있게 해당 정보를 시각화하여 준다.



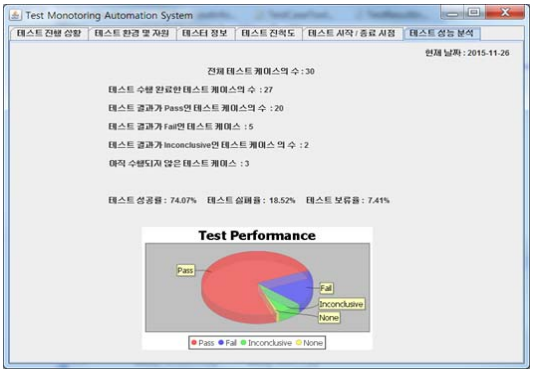
(그림 4) 테스트 모니터링 프레임워크 구조
(Figure 4) Structure of Test Monitoring Framework



(그림 5) 테스트 계획을 입력하기 위한 사용자 인터페이스
(Figure 5) User Interface for Inputting Test Plan

이 논문에서는 제안한 테스트 모니터링 프레임워크의 타당성을 보이기 위하여 논문에서 제시한 방법을 구현하였다. 그림 5는 테스트 계획을 입력하기 위한 사용자 인터페이스 모습이다. 사용자 인터페이스를 통하여 테스트 계획의 항목에 대한 정보를 입력할 수 있다. 어떤 항목은 텍스트 형태로 사용자가 직접 입력해야 하는 것도 있고, 다른 항목은 주어진 메뉴에서 선택해야 하는 것도 있다. 이렇게 사용자 인터페이스를 통해 테스트 계획을 입력함으로써 자연어로 기술하던 방식에 비해 간편하게 입력할 수 있으며, 또한 입력된 테스트 계획은 내부적으로 XML 형태로 변환되어 저장됨으로써 컴퓨터가 읽을 수 있다는 장점이 있다. XML 형태로 저장된 테스트 계획은 컴퓨터가 읽을 수 있어서 테스트 모니터링 활동에 필요한 정보를 쉽게 추출할 수 있다.

그림 6은 테스트 모니터링의 6가지 대상 중 3.2.3 절에서 언급한 테스트 성능을 모니터링한 결과를 시각적으로 보여주는 사례이다.



(그림 6) 테스트 성능 분석 결과
(Figure 6) Test Performance Analysis Result

5. 테스트 모니터링 프레임워크 평가

(표 2) 테스트 모니터링 시스템 평가
(Table 2) Evaluation of Test Monitoring Systems

테스트 모니터링 대상	테스트 모니터링 시스템			
	RQM	NEXCORE	QAComplete	본 논문 시스템
테스트 진행 상황	◐	●	○	●
테스트 환경 자원 사용	○	○	◐	●
테스터 참여 상황	◐	◐	○	●
테스트 단계 진척 상황	●	●	○	●
테스트 시작/종료 시점	◐	◐	◐	●
테스트 성능 분석	●	●	◐	●

(지원: ● 조건부 지원: ◐ 미지원: ○)

본 논문에서 제시하는 테스트 모니터링 프레임워크의 능력을 평가하기 위해 기존의 테스트 모니터링 시스템인 RQM, NEXCORE와 테스트 관리 시스템인 QAComplete를 대상으로 비교하였다. 비교 결과는 표 2와 같다. 단, 다른 도구의 경우 테스트 모니터링 기능으로 한정하여 비교하였다.

RQM[8]은 자동으로 테스트 모니터링을 수행하기 위해서 Junit, RAT, RFT, Robot 등 IBM의 여러 도구와 연동해야 하며, 사용자가 직접 테스트 프로젝트를 생성하고 처음 테스트를 수행할 때 테스트의 시작, 종료 조건, 테스트 케이스, 테스트 자원 등을 모두 수동으로 입력하고 설정해야 한다. 이런 정보를 입력해야만 테스트 진행 상황, 테스터 정보, 테스트 시작/종료 시점의 확인이 가능하다. 테스트 진척도와 테스트 성능 분석은 생성된 프로젝트에서 테스트를 수행한 결과를 받아오므로 사용자가 별도로 입력하지 않아도 모니터링이 가능하다.

NEXCORE[9]는 테스터가 도구에 테스트 시나리오 및 테스트 케이스를 등록하고, 테스트 관리자가 등록된 테스트를 수행할 담당 테스터에게 할당하는 작업을 일일이 수행해야 한다는 점에서 번거롭다. 또한 테스트 환경 자원과 테스터 정보는 관리자 입장에서만 확인이 가능하며 별도의 입력 없이 모든 참여자가 회원가입을 하여 정보를 입력해야만 확인할 수 있어 누락될 가능성이 있다. 다만 로그 분석을 이용한 테스트 결과가 사용자에게 제공된다는 점과 테스트에 대한 여러 응용 제어 기능을 지원한다는 장점이 있다.

QAComplete[7]는 많은 테스트 도구들이 그러하듯이 테스트 모니터링을 위한 전용 기능을 제공하지는 않는다. 다만 테스트 수행 결과를 보여주는 Test Run History 기능을 통하여 테스트 수행의 결과를 알 수 있다. 테스트 수행 결과에 초점이 맞춰져 있어서 각 테스트 실행의 시

작 시간, 테스트 대상, 테스트 수행 결과, 테스트 실행 환경, 테스트 수행 소요 시간 등을 표 형태로 하나의 화면에 제공한다.

본 논문에서 제안하는 테스트 모니터링 프레임워크는 이러한 테스트 모니터링 시스템과는 달리 테스트 계획, 테스트 케이스, 테스트 결과를 자동으로 읽어 테스트 진행 상황 분석, 테스트 환경 자원 제공 및 사용 분석, 테스터 정보 모니터링, 테스트 단계 진척도 모니터링, 테스트 시작/종료 시점 확인, 테스트 성능 분석 등 테스트 모니터링을 위한 특화된 기능을 사용자에게 제공한다. 이렇게 함으로써 테스트 이해관계자들에게 그들에게 반드시 필요한 정보를 정확하게 제공할 수 있다는 장점이 있다. 아울러 테스트 모니터링 과정의 여러 부분을 자동화함으로써 테스터들이 모니터링 활동을 위해 별도의 노력을 하지 않아도 되므로 그들의 업무 노력이 줄어들 수 있다는 장점도 있다.

6. 결 론

본 논문에서는 자동화된 테스트 모니터링 프레임워크 구축 방안을 제안하였다. 이를 위해 테스트 활동 산출물을 자동으로 처리할 수 있도록 소프트웨어 테스트 표준 문서를 참조하여 테스트 계획, 테스트 수행, 테스트 결과에 기본적으로 들어가야 할 항목과 포함되는 정보를 범주화하였고 테스트 활동 산출물의 정보가 XML로 저장되어 컴퓨터가 읽을 수 있게 하였다. 또한 테스트 모니터링 프레임워크를 구축하기 위한 테스트 모니터링 기능을 도출하였고 테스트 모니터링 기능을 실현하기 위해 필요한 정보 추출 방법 등을 제시하였다. 이를 통해 기존의 사용자들이 테스트 활동 산출물을 기술하고 이 정보를 이용하여 테스트 모니터링을 스스로 수행하여야 하는 데에 드는 부담을 줄였으며 아울러 테스트 활동 산출물의 정보를 컴퓨터가 인식할 수 있으므로 이 정보를 이용하여 자동화된 테스트 모니터링 프레임워크를 구축하였다.

참 고 문 헌 (Reference)

[1] A. P. Mathur, “Foundations of Software Testing, Person Education”, 2008.

[2] IEEE Std 29119-1-2013, “Software and systems engineering-Software testing-Part 1: Concepts and definitions”, 2013. http://dx.doi.org/10.1109/IEEESTD.2013.6588537

[3] P. Farrell-Vinay, “Manage Software testing”, Auerbach Publications, 2008.

[4] M.G. Limaye, Software Testing: Principles, Techniques and Tools, Tata McGraw-Hill Publishing Company Ltd, 2009.

[5] S.J. Kim, Software Testing Activity Modeling and Test Monitoring Tools Development, M.S. Thesis, Chungnam National University, 2016. http://www.riss.kr/link?id=T14055512

[6] Guru99, “Achieve your Project Goals through Test Monitoring & Control”, http://www.guru99.com/how-you-can-achieve-project-goals-through-test-monitoring-control.html

[7] SMARTBEAR, QAComplete, http://support.smartbear.com/qacomplete/docs/default.htm

[8] IBM, Rational Quality Manager, http://www-03.ibm.com/software/products/ko/ratiqualmana

[9] SK C&C, NEXCORE, http://nexcore.skcc.com

[10] TMMi Foundation, Test Maturity Model integration, Release 1.0, http://www.tmmi.org/pdf/TMMi.Framework.pdf

[11] S.H. Choi, H.S. Kim, G.Y. Lee, “TMMi Level 5 Quality Control Process Implementation Strategy”, Journal of KIISE:Software and Applications, Vol.41, No.8, pp.533-544, 2014.

[12] S.J. Kim, J.A. Kim, Y.J. Seo, H.S. Kim, “Construction of Test Monitoring Framework”, Proc. of KIISE 2015 Winter Conference, pp.531-533, 2015. http://www.dbpia.co.kr/Article/NODE06228661

[13] IEEE Std 29119-3-2013, “Software and systems engineering-Software testing-Part 3: Test Documentation”, 2013. http://dx.doi.org/10.1109/IEEESTD.2013.6588540

[14] IEEE Std 829-2008, “IEEE Standard for Software and System Test Documentation”, 2008. http://dx.doi.org/10.1109/IEEESTD.2008.4578383

[15] S.J. Kim, J.A. Kim, Y.J. Seo, E.Y. Cheon, H.S. Kim, “Test Plan Language Implementation for Model-based Testing”, Proc. of Korea Computer Congress 2015, pp.591-593, 2015. http://www.dbpia.co.kr/Article/NODE06394154

◐ 저 자 소 개 ◑

서 용 진 (Yongjin Seo)
2011년 충남대학교 컴퓨터공학과 졸업(학사)
2011년~현 재 충남대학교 컴퓨터공학과 (박사 과정, 석박사통합)
관심분야 : 소프트웨어 공학, 소프트웨어 테스팅, 소프트웨어 분석
E-mail : yjseo082@cnu.ac.kr

김 수 지 (Su Ji Kim)
2014년 충남대학교 컴퓨터공학과 졸업(학사)
2014년~2016년 충남대학교 컴퓨터공학과 졸업(석사)
2016년~현 재 ㈜ 디엔아이 연구원
관심분야 : 소프트웨어 공학, 소프트웨어 테스팅
E-mail : suji5090@gmail.com

김 현 수 (Hyeon Soo Kim)
1988년 서울대학교 계산통계학과(학사)
1991년 한국과학기술원 전산학과(공학석사)
1995년 한국과학기술원 전산학과(공학박사)
1995년~1995년 한국전자통신연구원 Post Doc.
1996년~2001년 금오공과대학교 조교수
1999년~2000년 Colorado State University 방문교수
2007년~2008년 Purdue University 방문연구교수
2001년~현 재 충남대학교 컴퓨터공학과 교수
관심분야 : 소프트웨어 공학, 소프트웨어 테스팅, 소프트웨어 아키텍처
E-mail : hskim401@cnu.ac.kr