1999년 한국정보처리학회 추계 학술발표논문집 제6권 제2호

생명주기 단계별 테스트를 위한 체크리스트의 개발

○
이하용*, 양해술**
*한국소프트웨어품질연구소
**호서대학교 벤처전문대학원

e-mail : insq@unitel.co.kr

Development of Check-List for Test of Life-cycle Each Phase

Ha-Yong Lee*, Hae-Sool Yang**
*Institute of Software Quality(INSQ)
**Graduate School of Venture, Hoseo University

요 약

소프트웨어가 대형화되고 복잡해질수록 품질의 중요성은 높아지게 된다. 많은 소프트웨어 개발 프로젝트들이 충분한 품질관리가 이루어지지 못한 채 잠재적인 위험 요소들을 지니고 제품화되고 있다. 소프트웨어의 오류를 최소화하기 위한 방안으로 개발된 소프트웨어에 대해 시험 사례를 개발하여 적용하는 것이 일반적이다. 그러나 소프트웨어 개발 과정에서 발생하는 오류는 초기 단계의 오류일수록 최종 소프트웨어 제품에 미치는 영향이 크므로 초기 단계에 문제점을 점검하는 것이 더욱 중요한 문제일뿐 아니라 생명주기 전 단계에 걸쳐 각 단계에 적합한 시험 항목을 개발하여 적용하는 것이 고품질의 소프트웨어를 개발하는 최선의 방법이 될 수 있을 것이다. 본 연구에서는 생명주기 전 단계에 걸친 테스트 항목을 개발하였다.

1. 서 론

소프트웨어가 대형화되고 복잡해질수록 품질의 중요성은 더욱 높아지게 된다. 많은 소프트웨어 개발 프로젝트들이 충분한 시간을 할당하여 품질관리를 수행하거나 검토가 이루어지지 못한 채 잠재적인 결함 발생 위험 요소들을 지닌 상태에서 제품화되고 있다.

일반적으로 소프트웨어의 결함이나 오류를 최소화하여 고품질의 소프트웨어를 개발하기 위한 방안으로 적용하는 방법은 개발된 소프트웨어에 대해 결함이나 오류를 찾아낼 수 있을 것이라 사료되는 시험 사례를 개발하여 소프트웨어를 직접 실행해보면서 검토하는 것이 일반적이다.

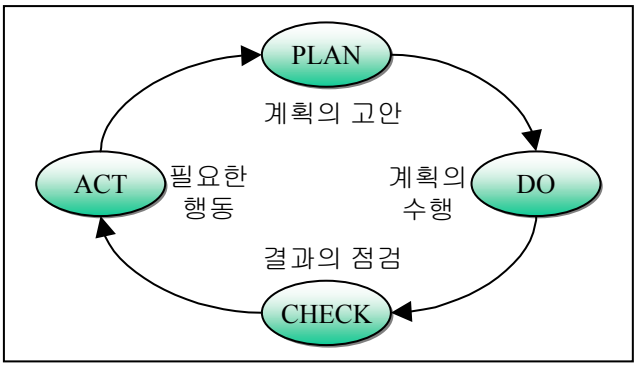
그러나 소프트웨어 개발 과정에서 발생하는 오류는 초기 단계의 오류일수록 최종 소프트웨어 제품에 미치는 영향이 크므로 초기 단계에 문제점을 점검하는 것이 더욱 중요한 문제일뿐 아니라 생명주기 전 단계에 걸쳐 각 단계에 적합한 시험 항목을 개발하여 적용하는 것이 고품질의 소프트웨어를 개발하는 최선의 방법이 될 수 있을 것이다.

따라서 본 연구에서는 생명주기 전 단계 중 분석 단계, 설계 단계, 프로그래밍 단계에 걸친 테스트 항목을 개발함으로써 소프트웨어를 품질 향상을 위한 기반을 구축하였다.

2. 소프트웨어의 테스트

소프트웨어의 개발은 공정의 순환으로 이루어진다. 공정의 순환은 다음의 4가지 요소로 구성된다.

· Plan : 계획의 고안
목적을 규정하고 그 목적을 이루기 위해 요구되는 전략과 지원 수단들을 결정한다. 그 계획은 현 상황의 평가를 기반으로 해야 하고, 전략은 개선 계획으로 유도하려는 전략상 독창성과 비결에 초점이 맞춰져야 한다.



(그림 1) 소프트웨어 개발 공정의 순환

· Do : 계획의 수행
조건들을 만들고 계획을 실행시키기 위한 필요한 훈련을 수행하고, 모든 구성원들이 철저히 목적과 계획을 이해하도록 한다. 작업자들에게 계획을 실행하고 작업을 이해하는데 필요한 절차와 기술을 가르치며, 그 절차에 관련된 작업을 수행한다.

· Check : 결과의 점검
작업이 계획에 따라 잘 진행되고 있는지, 예상되는 결과를 얻을 수 있는지를 결정하기 위해 점검한다. 단위별 절차의 수행, 조건들의 변화, 발생할 수 있는 비정상적인 것들을 점검하고 가능한 한 자주, 목적과 관계된 작업의 결과를 비교한다.

· Action : 필요한 행동
점검표에 작업이 계획대로 수행되지 않거나 결과가 예상대로 되지 않으면, 적절한 행동으로 조치를 취한다.

테스트는 오직 plan-do-check-action(PDCA) 순환의 점검(check) 요소에 관련되어 있다. 소프트웨어 개발팀은 나머지 세 요소에 대한 책임이 있다. 개발팀은 프로젝트 계획을 세우고, 소프트웨어를 개발하며 테스트하는 사람은 소프트웨어가 수요자와 사용자의 필요에 부합하는지를 결정하기 위한 점검을 수행한다. 만약 부합되지 않는다면, 테스트하는 사람은 결점들을 개발팀에게 보고하며, 개발팀은 처리되지 않은 결점들을 바로잡는다. 테스트가 갖는 역할은 테스트하는 사람에게 부여된 점검 책임이 달성되는 것이다.

3. 소프트웨어의 결함

소프트웨어의 결함이란 원하는 제품의 속성과 불일치를 보이는 것이다. 결함에는 제품 명세의 결함과 고객이나 사용자의 기대와 불일치하는 경우와 같이 두 가지 종류가 있다.

3.1 제품 명세의 결함(defect)

제품 명세의 결함이란 만들어진 제품이 제품에 대한 명세와 다른 경우를 말한다. 즉, 제품을 만드는 알고리즘이 명세와 다르다면 결함이 있는 것으로 생각할 수 있다.

3.2 고객이나 사용자의 기대와 불일치

이러한 불일치는 만들어진 제품에 사용자가 원하는 것이 없다든가 만들어진 제품에 포함될 수 있도록 명세되지 않았다는 것이다. 이러한 잘못은 명세서를 작성하는 과정에서 또는 요구 과정에서 발생하거나 구현된 요구가 만족스럽지 못한 경우에 발생한다.

결함은 일반적으로 다음과 같은 세 가지 부류 중 하나의 형태로 나타난다.

① 잘못(wrong)
명세가 부정확하게 구현된 경우에 해당한다. 이 결함은 고객이나 사용자의 명세와 불일치하는 것이다.

② 누락(missing)
명세되거나 요구된 것이 만들어진 제품에 구현되어 있지 않은 경우에 해당한다. 이것은 명세대로 구현되지 않은 경우이거나 고객의 요구사항이 제품이 만들어지는 동안 또는 만들어진 이후에 확인된 경우이다.

③ 여분(extra)
명세되지 않았던 요구가 제품에 포함된 경우이다. 이것은 항상 명세와 불일치하지만 제품의 사용자가 원하는 속성일 수도 있다. 그러나 이것은 결함으로 간주된다.

3.3 결함과 실패

결함은 소프트웨어 시스템에 포함된다. 결함은 잘못(wrong), 누락(missing), 여분(extra)으로 분류된다. 결함은 소프트웨어 자체에서 발견되거나 매뉴얼과 문서화를 지원할 때 발견될 수 있다. 결함이 소프트웨어 시스템에 포함된 문제점이긴 해도, 그것이 사용자, 고객, 운영 시스템에 영향을 미치기 전까지는 충격이 되지 않는다.

동작 중에 발생하는 오류나 사용자, 구매자에게 부정적인 영향을 미치는 결함을 실패(failure)라 한

다. 결함이 갖는 주된 관심사는 결함이 실패로 바뀔 수 있다는 것이다. 조직에 손해를 끼치는 것은 실패(failure)이다.

어떤 결함은 전혀 실패로 나타나지 않을 수도 있다. 다시 말해서, 하나의 결함이 많은 실패를 야기할 수도 있다는 것이다.

3.4 공정과 결함 비율

대부분의 결함은 작업을 적절히 수행하지 않은 공정에서 발생된다. 만일 요구 공정에 결함이 있다면 적절한 정보를 모을 수 없을 것이다. W. Edwards Deming은 적어도 90%의 결함은 공정 문제에 의해 발생한다고 했으며 많은 연구들이 증거를 제공했다. 테스트하는 사람은 이점을 중시해야 한다.

4. 생명주기 단계별 체크리스트

생명주기 단계별로 시험을 수행하기 위한 체크리스트를 개발하였다.

4.1 요구분석 단계의 시험을 위한 체크리스트

만일, 요구분석 단계에서 에러가 있다면 전체 어플리케이션은 에러를 가지게 된다. 요구분석 단계의 테스팅은 좋은 요구가 될 가능성을 증가시킬 수 있다.

<표 1> 요구분석 단계의 체크리스트

#	테스트기준	평가				제안된 테스트
		매우 적합	적합	부 적합	N/A	
1	이용할 수 있는 조직의 정책과 절차는 명시되어 있는가?					정책과 절차를 개발할 책임이 있는 사람과 모든 적용가능한정책들이 확인되었는가를 확인하라.
2	요구사항들은 이런 정책과 절차에 따르는가?					정책과 절차에 따르는 가를 보증하기 위해 요구를 재검토하라.
3	요구사항들은 요구방법론에 따라 문서화되어 있는가?					모든 필요한 문서가 완전히 갖추어져 있는지 확인하기 위해 요구를 검토하라.
4	비용/이익 분석은 적절한 절차에 따라 준비되어 있는가?					절차에 따라 준비되었는가를 확인하기 위해 비용/이익 분석을 검토하라.
5	요구단계는 요구방법론의 취지를 만족시키는가?					요구사항으로부터 전달사항을 살피고, 방법론의 내용과 일치하는지 평가하라.

요구분석 단계의 테스팅을 위한 체크리스트를 개발하기에 앞서 테스팅 요소를 유형별로 분류할 필요가 있으며 본 연구에서는 방법론 테스트 요소, 정확성 테스트 요소, 사용용이성 테스트 요소, 유지보수성 테스트 요소 등 15가지로 분류하여 개발하였다.

<표 1>에 방법론 테스트 요소에 관한 체크리스트를 나타내었다.

4.2 설계 단계의 시험을 위한 체크리스트

설계 단계는 컴퓨터화된 어플리케이션의 구조를 테스트할 기회를 준다. 이런 점에서 요구사항들은 컴퓨터에 이용될 시스템 구조로 변환된다. 프로젝트 팀의 구조가 효율적일수록 프로젝트의 성공확률은 커진다.

설계 단계의 테스팅을 위한 체크리스트를 개발하기 위해 테스팅 요소를 분류할 수 있다. 본 연구에서는 데이터 응집성 조정, 파일 응집성 조정, 우발사건 계획, 방법론 준수, 유지관리 용이성 등 15가지 관점에서 체크리스트를 개발하였다. <표 2>는 설계 단계의 시험을 위한 체크리스트의 예를 나타내었다.

<표 2> 설계 단계의 체크리스트

#	테스트기준	평가				제안된 테스트
		매우 적합	적합	부 적합	N/A	
1	트랜잭션이 설정되는 동안 정확하고 완전한 제어가 이루어지는가?					정확하고 완전한 제어가 가능한 트랜잭션 설정의 타당성을 점검하라.
2	모든 트랜잭션이 시작되었음을 확인하기 위해 연속적인 숫자들의 입력을 가지고 입력 트랜잭션의 제어가 이루어지는가?					모든 입력값이 입력되었음을 확인하는 입력제어의 타당성을 점검하라.
3	데이터의 정확하고 완전한 전송을 위해 통신의 제어가 이루어지는가?					정확하고 완전한 제어가 가능한 전송의 타당성을 조사하라.
4	현금 영수증과 같은 주요 항목의 트랜잭션을 위해 일괄 제어가 제공되는가?					모든 프로시저에 대한 일괄적인 제어의 타당성을 조사하라.
5	구매 주문과 같은 주요 항목의 입력 트랜잭션을 위한 일괄 숫자가 제공되는가?					순차적인 일괄 프로시저의 타당성을 조사하라.

4.3 프로그래밍 단계의 시험을 위한 체크리스트

프로그래밍 단계 수행의 복잡도는 설계 단계의 완전함에 달려 있다. 잘 정의되고 측정할 수 있는 설계 명세들은 프로그래밍 작업을 매우 간단하게 할 수 있다. 반면에 초기 단계 동안 결정을 하지 못한 것은 프로그래밍 단계에서 결정될 필요가 있다.

프로그래밍 단계 동안의 테스팅은 정적 또는 동적이다. 프로그래밍 단계에서 결과 코드는 실행 가능하지 않을 수도 있으므로 다른 테스트 툴들이 요구될 수 있다.

프로그래밍 단계의 테스팅을 위한 체크리스트를 개발하기 위해 테스팅 요소를 분류할 수 있다. 프로그래밍 단계에 관해서는 데이터 무결성 조정, 파일 무결성 조정, 보안 절차, 프로그램 유지보수성 등 15가지 관점에서 체크리스트를 개발하였다. <표 3>은 프로그래밍 단계의 시험을 위한 체크리스트의 예를 나타내었다.

<표 3> 프로그래밍 단계의 체크리스트

#	테스트기준	평가				제안된 테스트
		매우 적합	적합	부 적합	N/A	
1	다음에 뒤따르는 행동이 수행될 수 있도록 하기 위해 에러들이 정확하게 확인되고 설명되었는가?					데이터 엔트리 프로시저들의 완전성을 검사한다.
2	프로시저들이 데이터 에러들을 바로잡는 행동을 취하도록 수립되었는가?					확인된 에러들을 바로잡는 행동을 취하기 위한 프로시저들의 타당성을 검사한다.
3	에러들이 적시에 교정되는 것을 보장하도록 프로시저들이 설립되었는가?					에러들이 적시에 교정되는 것을 보장할 프로시저들을 검증한다.
4	트랜잭션들이 시스템 내의 한 포인트에서 다른 포인트로 이동할 때 그들의 완전성과 정확성을 보장하도록 조정장치들이 설치되었는가?					트랜잭션들이 시스템을 통해 흐를 때 그들의 정확성과 오나전성을 보장하는 프로시저들의 타당성을 검사한다.
5	자동화된 인증 프로시저들의 위반을 확인하고 대처하기 위한 프로시저들이 설립되었는가?					자동화된 인증 위반 프로시저들의 적절함을 검사한다.

5. 결 론

소프트웨어를 개발하면서 오류가 없는 완전 무결한 소프트웨어를 개발하는 것은 불가능하다. 다만, 오류를 최소화함으로써 오류로 인한 손실 발생 가능성을 최소화할 수밖에 없다.

오류는 소프트웨어 개발 과정에서 초기 단계의 오류일수록 최종 소프트웨어 제품에 미치는 영향이 크므로 초기 단계부터 생명주기 전 단계에 걸쳐 시험을 통해 문제를 최소화할 필요가 있다.

본 연구에서는 고품질의 소프트웨어 제품을 개발하기 위한 방안으로 생명주기 각 단계별로 적용할 수 있는 테스트 항목을 개발하였다.

최종적으로 개발이 완료된 소프트웨어를 수행할 때 나타날 수 있는 각종 오류들은 프로그래밍 단계에 원인이 국한되는 것이 아니라 그 이전 단계에서 발생된 문제로 인한 것일 수도 있으며 이러한 오류일수록 더욱 중대한 문제를 야기할 수 있다.

따라서 본 연구를 통해 개발된 생명주기 단계별 시험을 위한 체크리스트를 활용하여 생명주기 전단계에 걸친 효과적인 시험 공정이 이루어질 수 있을 것이다.

향후, 테스트 항목을 추가하고 구체적인 테스트 방법에 대한 보완이 이루어질 수 있도록 지속적인 연구가 진행되어야 할 것이다.

참 고 문 헌

[1] Moller, K. H. and Paulish, D. J., "Software Metrics", Chapmen & Hall(IEEE Press), 1993.

[2] Wallmuller, E., "Software Quality Assurance A practical approach", Prentice Hall, 1994.

[3] ISO/IEC 12119, "Information Technology - Software Package - Quality requirement and testing".

[4] 吉澤. 東. 片山, “ソフトウェアの 品質管理と生産技術”, 日本規格協會, 1990. 5.

[5] 水野幸男, “ソフトウエアの綜合的品質管理“, 日科技連出版, 1993.

[6] 양해술, 이하용, “설계단계에서의 품질평가 툴킷(ESCORT-D)의 설계 및 구현“, 한국정보과학회 논문지(C), Vol. 3, No. 3, 1997. 6.

[7] 양해술, “한진해운 신정보(영업 및 물류)시스템의 품질보증과 품질평가”, 한진해운(주) 구현단계 확인평가, 1998. 9. 7.

[8] 양해술, “소프트웨어 제품 평가 지원도구의 개발“, ETRI 컴퓨터 · 소프트웨어 기술연구소 용역과제, 3차년도최종보고서, 1999. 10.