2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

소프트웨어 구현단계의 테스트를 위한 시험 측정 항목의 개발

○
이하용*, 황석형** 양해술***
*한국소프트웨어품질연구소
**선문대학교 자연과학대학 정보과학부
***호서대학교 벤처전문대학원

e-mail : insq@unitel.co.kr

Development of Test Measurement Items for Test of Software Implementation Phase

Ha-Yong Lee*, Hae-Sool Yang**
*Institute of Software Quality(INSQ)
**Sun-Moon Univ. Div. of Information & Computer Science
***Graduate School of Venture, Hoseo University

요 약

소프트웨어의 사용자층이 확대되면서 소프트웨어는 갈수록 많은 기능을 가진 복잡한 시스템이 되고 있다. 사용자의 요구 사항에 부응할 수 있도록 소프트웨어가 대형화되고 복잡해졌으며 이로 인해 품질의 중요성이 더욱 높아지게 되었다. 그러나 다수의 소프트웨어들이 품질관리의 미흡으로 인해 발견되지 못한 오류들을 지닌 채 상품화되고 있는 것이 현실이다. 소프트웨어의 오류를 최소화하기 위해 일반적으로 소프트웨어에 대한 시험 사례를 통해 검증하게 되며 구현 이후, 시험 단계에서 이루어지는 경우가 대부분이다. 그러나 소프트웨어의 오류는 생명주기 전단계에 걸쳐 발생될 수 있으므로 생명주기 전단계에 걸친 적절한 시험을 통해 문제점을 점검하는 것이 고품질의 소프트웨어를 개발하는 최선의 방법이 될 수 있을 것이다. 본 연구에서는 생명주기 단계 중 소프트웨어 구현 단계의 테스트를 위한 시험 측정 항목을 개발하였다.

1. 서 론

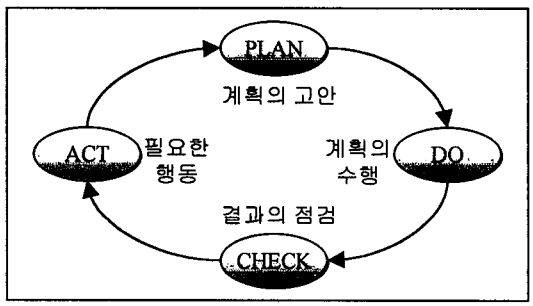
소프트웨어의 사용자층이 확대되면서 소프트웨어는 갈수록 많은 기능을 가진 복잡한 시스템이 되고 있다. 사용자의 요구 사항에 부응할 수 있도록 소프트웨어가 대형화되고 복잡해졌으며 이로 인해 품질의 중요성이 더욱 높아지게 되었다.

그러나 다수의 소프트웨어들이 품질관리의 미흡으로 인해 발견되지 못한 오류들을 지닌 채 상품화되고 있는 것이 현실이다. 소프트웨어의 오류를 최소화하기 위해 일반적으로 소프트웨어에 대한 시험 사례를 통해 검증하게 되며 구현 이후, 시험 단계에서 이루어지는 경우가 대부분이다.

그러나 소프트웨어의 오류는 생명주기 전단계에 걸쳐 발생될 수 있으므로 생명주기 전단계에 걸친 적절한 시험을 통해 문제점을 점검하는 것이 고품질의 소프트웨어를 개발하는 최선의 방법이 될 수 있을 것이다. 본 연구에서는 생명주기 단계 중 소프트웨어 구현 단계의 테스트를 위한 시험 측정 항목을 개발하였다.

2. 소프트웨어의 테스트

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



(그림 1) 소프트웨어 개발 공정의 순환

- 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.

- 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 한다. 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 공

정에서 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. 구현단계의 시험 측정 항목

본 연구에서는 구현단계에서 적용할 수 있는 시험을 수행하기 위한 시험 측정 항목을 개발하였다.

구현 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.

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

구현 단계의 테스팅을 위한 시험 측정 항목을 개발하기 위해 테스팅 요소를 분류할 수 있다. 구현 단계에 관해서는 데이터 통합 제어, 인증 규칙, 파일 응집성 제어 등 15가지 관점에서 시험 측정 항목을 개발하였다.

4.1 구현된 데이터 통합 제어

특정한 제어는 요구된 처리 정밀도를 달성하는 수단으로 구현될 필요가 있다. 적절하지 못하게 구현된 제어는 수립된 수준의 제어 허용 오차(tolerance)를 달성할 수 없을 것이다. 그리고 제어의 목표에 관한 만연된 오해 때문에, 극단적으로 단순화된 해결책이 구현될 수도 있으므로 복잡도 제어가 제어 목표를 달성하기 위해 필요하다. <표 1>에 데이터 통합 제어에 관한 시험 측정 항목을 나타내었다.

4.2 구현된 인증 규칙

인증 규칙을 우회하는 것을 어렵게 하는 수단으로 인증 규칙이 구현될 필요가 있다. 예를 들면, 인증 한계가 설정될 때, 사람들은 규정된 한계 하에서 다수의 항목을 입력함으로써 그 한계를 우회할 수 없어야 한다. 그러므로 인증 규칙은 규칙의 강제성을 고려해야 할뿐만 아니라, 이러한 규칙을 우회하는 좀 더 일반적인 방법들을 고려해야 한다.

<표 2>는 인증 규칙에 관한 시험을 위한 시험 측정 항목의 예를 나타내었다.

<표 1> 데이터 통합 제어에 관한 시험 측정 항목

#	테스트기준	평가				제안된 테스트
		매우 적합	적합	부 적합	N/A	
1	자동화된 시스템으로 가는 입구로서 트랜잭션을 어떻게 기록할 것인가를 나타내는 프로시저가 기술되었는가?					데이터 에러 메시지와 리스트 작성의 유용성을 시험하라.
2	입력이 시스템 명세에 따르는지를 보증하기 위해 데이터 타당성 체크가 구현되었는가?					데이터 타당성 체크의 완전성을 검토하라.
3	다음에 뒤따르는 행동이 수행될 수 있도록 하기 위해 에러들이 정확하게 확인되고 설명되었는가?					데이터 엔트리 프로시저들의 완전성을 검사한다.
4	프로시저들이 데이터 에러들을 바로잡는 행동을 취하도록 수립되었는가?					확인된 에러들을 바로잡는 행동을 취하기 위한 프로시저들의 타당성을 검사한다.
5	에러들이 적시에 교정되는 것을 보장하도록 프로시저들이 설립되었는가?					에러들이 적시에 교정되는 것을 보장할 프로시저들을 검증한다.

<표 2> 인증 규칙

#	테스트기준	평가				제안된 테스트
		매우 적합	적합	부 적합	N/A	
1	인증 방법이 수동과 자동으로 나누어져 있는가?					선택된 인증 방법의 타당성을 평가하라.
2	각 트랜잭션에 대해 수정 인증 절차를 명세하기 위한 프로시저가 준비되었는가?					수동 인증 절차의 적절함을 검토하라.
3	시스템의 자동화된 세그먼트에서 트랜잭션을 인증하기 위해 프로그램에 구현된 방법이 있는가?					인증 방법이 적절하게 구현되었는가를 결정하기 위해 프로그램 명세나 실제 프로그램을 시험하라.
4	수동 인증 절차의 위반을 나타내기 위해 수동 절차가 수립되었는가?					수동 인증에 대한 위반 절차의 타당성을 시험하라.
5	자동화된 인증 절차의 위반에 관해 식별하고 행동하기 위한 절차가 수립되었는가?					자동화된 인증 위반 절차의 적절함을 시험하라.

4.3 구현된 파일 응집성 제어

파일 응집성 제어는 파일 응집성 감소의 가능성을 최소화하기 위한 수단으로 구현되어야 한다. 그리고 파일 응집성 제어는 응집성의 감소를 막고 발견해야 한다.

<표 3>은 파일 응집성 제어에 관한 시험 측정 항목의 예를 나타내었다.

<표 3> 파일 응집성 제어에 관한 시험 측정 항목

#	테스트기준	평가				제안된 테스트
		매우 적합	적합	부적합	N/A	
1	각 파일의 응집성에 대한 책임자가 임명되어 있는가?					임명된 사람이 필요한 기술과 가용한 시간을 가지고 있는가를 확인하라.
2	파일 응집성 제어가 파일 응집성 요구에 따라 구현되었는가?					요구단계 동안 확립된 응집성 요구와 구현된 제어를 비교하라.
3	파일 응집성 문제의 적절한 사람을 통지하기 위한 절차가 수립되어 있는가?					파일 응집성 문제를 기록하기 위한 절차의 적절함을 시험하라.
4	정기적으로 파일의 응집성을 검토하기 위한 절차가 수립되었는가?					파일 응집성 검증 빈도의 합리성을 조사하라.
5	응집성 제어를 가져야 하는 파일의 서브셋이 있는가?					모든 파일 서브셋이 응집성 제어를 통해 적절히 보호된다는 것을 사용자와 함께 확인하라.
6	자동화된 파일 제어와 독립적인 유지보수 제어 합계 사이에 정기적인 조화를 위해 기록된 프로시저가 있는가?					자동화와 수동 유지보수 제어를 조화시키기 위한 절차의 합리성과 적시성을 검토하라.
7	특별한 인증 제어를 위한 *sensitive transaction* subject가 있는가?					sensitive transaction 인증 제어가 적절한가를 검토하라.

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 구현 단계 테스팅을 위한 시험 측정 항목을 활용하여 효과적인 구현 단계 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.

[9] 이하용, 양해술, "생명주기 단계별 테스트를 위한 체크리스트의 개발", 한국정보처리학회 학술발표논문집, 1999. 5.