Journal of the Korea Academia-Industrial cooperation Society
Vol. 15, No. 7 pp. 4522-4527, 2014

http://dx.doi.org/10.5762/KAIS.2014.15.7.4522
ISSN 1975-4701 / eISSN 2288-4688

정보통신 소프트웨어 테스트 설계 효율성에 대한 연구

장진욱[1*]
[1]건국대학교 인터넷미디어공학부

A study of The effects on applying test design to Info-communication Software

Jin-WooK Jang[1*]
[1]**Division of Internet & Multimedia Engineering, Konkuk University**

요 약 정보통신 소프트웨어는 다양한 기능과 서비스를 포함하고 있다. 소프트웨어 테스트는 인수단계에서 구현을 확인하는 정도로 인식되었으나 최근에는 개발단계별 테스트에 대한 관심이 높아지고 있다. 특히 소프트웨어가 사용자에게 인도되는 인수단계에서의 특히 테스트의 중요성이 높다. 따라서 인수테스트 단계의 제한된 시간과 리소스를 효과적으로 사용하여 테스트하기 위한 기법과 방법에 대한 연구가 많이 이루어지고 있다. 본 논문에서는 테스트 설계기법을 이용하여 전자통신 소프트웨어에 적용하고 테스트 케이스 실행 개수와 시간을 측정하여 테스트 설계의 효율성을 측정하였다. 본 연구는 테스트 설계 및 실행과정에서 최소한의 리소스를 이용한 효과적인 테스트 실행에 기여할 것으로 본다.

Abstract Info-communication software contains various functions and services. A software test is the only acceptance test step. Recently, however, all development steps have become important. In particular, the software acceptance step is very high, and a study of the effective use of limited time and resources on acceptance step is needed. This study examined the efficiency of the Test Design Technique for Software Test.
The proposed Test Design process was applied to the domain systems of info-communication and the degree of improvement was measured. This paper establishes Software Test design process Infrastructure based on the developed software.

Key Words : Software Test, Software Quality, Test Design, Test Design Process, Test case

1. 서론

정보통신 소프트웨어의 규모와 복잡도가 증가하면서 소프트웨어 테스트에 대한 관심이 높아지고 있다. 과거에는 소프트웨어 테스트는 개발과출시 마지막 단계에서 구현을 확인하는 정도로 인식되었으나 최근에는 개발단계별 테스트에 대한 관심이 높아지고 있다. 특히 소프트웨어가 사용자에게 인도되는 인수단계에서의 테스트를 그 중요성이 매우 높다. 따라서 인수테스트 단계의 제한된 시간과 리소스를 효과적으로 사용하여 테스트하기 위한 기법과 방법에 대한 연구가 많이 이루어지고 있다[1].

본 논문에서는 테스트 설계기법을 이용하여 전자통신 소프트웨어에 적용하고 테스트 케이스 실행 개수와 시간을 측정하여 테스트 설계의 효율성을 측정하였다.

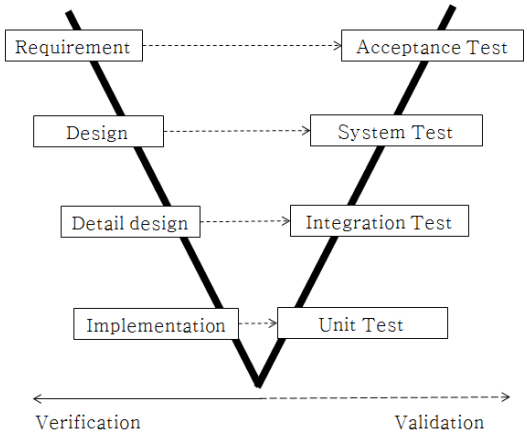
그리고 테스트 설계 프로세스를 제안하여 테스트 설계과정을 객관화함으로서 테스트 케이스의 활용도를 높이기 위하여 노력하였다. 본 연구는 테스트 설계 및 실행과정에서 최소한의 리소스를 이용한 효과적인 테스트 설계 및 실행에 기여할 것으로 본다.

2. 관련연구

2.1 테스트 활동

*Corresponding Author : Jin-Wook Jang(Konkuk Univ.)
Tel: +82-2-450-0547 email: jwjang@konkuk.ac.kr
Received March 19, 2014 Revised (1st April 10, 2014, 2nd April 24, 2014) Accepted July 10, 2014

테스트는 테스트 계획 및 통제 활동, 테스트 조건의 선택, 테스트 케이스 설계, 테스트 수행 및 점검, 테스트 종료 조건 정의, 테스트 마감 등의 일련의 활동을 모두 포함한다. 이러한 테스트 활동 중 시간과 노력이 많이 소요되는 활동이 테스트 설계 활동이다[2]. 테스트 설계 활동은 일반적이고 추상적인 테스트의 목적을 달성하기 위하여 구체적인 테스트 조건을 정의하고 테스트 케이스로 변환하는 활동이다. 즉, 테스트에 필요한 소프트웨어 요구사항 및 설계 문서에 대한 리뷰를 통하여 테스트 조건을 식별하고 우선순위를 선정한 후 테스트 설계 기법을 활용하여 테스트 케이스를 도출하는 활동인 것이다. 전자통신 소프트웨어는 소프트웨어 특성상 대부분 폭포수형 기반의 수명주기를 갖는다[3,4]. 따라서 폭포수형 개발 수명주기와 통합된 테스트 수명주기 모델이 필요하다[5,6]. 왜냐하면 테스트 설계 활동은 Fig. 1과 같이 통합된 테스트 수명주기 모델에 의하여 공식적으로 이루어지지 않으면 생략되거나 인수단계에 이르러서야 수행되는 등 현실적으로 어려움이 많기 때문이다.



[Fig. 1] V-model

2.2 테스트 설계

2.2.1 문장 테스트

문장 테스트는 프로그램 내의 모든 문장들을 빠짐없이 실행시켜 보고 제대로 작동하는가를 확인하는 방법으로 정의하고 있다. 문장 테스트에서는 프로그램 내의 모든 문장을 어느 한 테스트 케이스에서 최소한 한 번은 테스트하도록 한다. 제어흐름 그래프에서는 문장이 노드로 표현되기 때문에 문장 테스트하도록 한다.

제어흐름 그래프에서는 문장이 노드로 표현되기 때문에 문장 테스트는 모든 노드를 커버하는 기준에 해당한다. 문장 테스트에서는 프로그램의 모든 부분을 테스트를 통하여 빠짐없이 살펴본다는 점에서 우수한 테스트 기법이다. 이런 이유로 IEEE 단위 테스트 표준에서는 이것을 테스트의 최초기준으로 정의하고 있다.

그러나 문장 테스트는 제어흐름 테스트 전략 중에서 가장 약한 기준이다. 테스트가 문장 테스트에도 미치지 못한다면 이것은 프로그램 내에 조사해 보지 않은 부분이 존재함을 의미하며 검증하지 않은 부분에 결함이 없음을 단정할 수 없다.

2.2.2 분기 테스트

프로그램 내의 모든 문장을 실행을 통하여 최소한 한 번은 테스트 한다는 점에서 분기 테스트는 우수한 테스트 전략이다. 그러나 "IF(C) S1; S2;" 같은 경우를 고려하여야 한다.

만일 테스트 데이터가 조건 C를 참이 되도록 선정되면 이 테스트로써 프로그램 내의 모든 문장 S1, S2가 모두 실행된다. 그러나 테스트 데이터가 조건 C를 거짓이 되도록 선정되면 문장 S2만이 실행된다. 이 논의를 통해서 우리는 테스트 데이터를 어떻게 선정 하는가에 따라서 모든 문장이 커버되기도 하고 일부의 문장만이 커버됨을 알 수 있다[7,8].

테스트 데이터의 선정결과에 따라서 위의 프로그램 문장과 이 프로그램 문장 "IF(C) {S1; S2;}"는 테스트에서 동일한 효과를 가질 수 있다.

그러므로 테스트 데이터 선정결과에 따라서 위에 있는 두 프로그램 문장사이의 차이를 드러낼 수 없는 경우가 발생한다. 즉, 동일한 일련의 프로그램 문장에 대하여 여러 개의 다른 실행방법이 가능하기 때문에 이들도 각각 테스트해 보아야 한다. 가능한 모듈 실행 가능성의 조합은 대개의 경우 너무 많기 때문에 시간적인 제약을 고려하여 우리는 프로그램의 모든 분기가 최소한 한번은 실행되는 것을 테스트 전략으로 활용할 수 있다. 이러한 테스트 전략을 분기 테스트라고 한다.

2.2.3 조건 테스트

분기 테스트에서는 결정의 결과만을 고려한다. 그러나 많은 결정이 단순한 조건보다는 여러 개의 조건에 기반을 두고 이루어진다. 예를 들면 조건문장 "IF A<10 AND B>250 THEN"에서 참인결과는 AND로 결합된 각 조건

이 모두 참일 때 얻어지며 조건 중의 어느 하나라도 거짓이면 조건 전체가 거짓이 된다. 또한 2개의 조건이 OR로 결정된 조건문 “IF A<10 OR B>250 THEN”에서는 어느 하나의 조건이라도 참이면 조건 전체가 참이 되기 때문에 처음 조건이 참이면 다음 조건의 참, 거짓과 무관하게 조건 전체가 참이 된다.

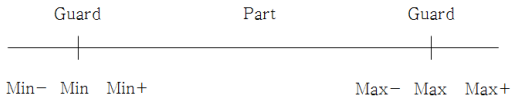
따라서 보다 철저하게 모든 경우를 고려하려면 복합 조건 내의 모든 단순한 조건의 참, 거짓의 경우를 최소한 한 번씩은 테스트할 필요가 있다. 이와 같은 기법을 조건 테스트라고 부른다. 대체로 조건 테스트는 분기 테스트보다 더 철저한 테스트가 된다고 할 수 있지만 예외가 존재한다. 이러한 결점을 보완하기 위하여 조건들의 참, 거짓인 경우를 적어도 한 번씩 테스트하는 조건에 추가하여 결정의 결과 또는 참, 거짓인 경우를 적어도 한 번씩 테스트하는 조건 결정 테스트를 정의하고 있다.

2.2.4 경계 테스트

경계 테스트(boundary value test)는 입력 영역의 경계상의 값 또는 경계에 가까운 값을 테스트 데이터로 사용하면 개발자의 실수를 검출할 확률이 높다는 경험적 지식에 기반을 둔 테스트 케이스 설계방법이다.

따라서 분할 테스트에서 테스트 데이터로 주로 사용하는 입력영역을 대표하는 값 이외에 경계선에 있는 값이나 경계의 바로 안팎에 있는 값들이 테스트 데이터로 포함되면 에러 검출에 도움이 될 수 있다.

경계분석에서는 분류의 정확성에만 관심을 두기 때문에 처리과정의 정확성은 다른 방법으로 확인해야 한다. 명세서에 정의된 영역의 집합이 모든 입력 영역을 커버하지 않거나 두 영역사이에 겹치는 부분이 있을 수 있기 때문에 영역분석은 명세서의 모호한 점이나 상충되는 내용을 점검 한다는 점에서 효과적이다. 영역분석은 분할 테스트의 한 형태로 볼 수 있다. 즉, 하나의 입력변수를 갖는 프로그램에 대해 경계값 테스트를 하는 경우에 프로그램의 입력영역이 몇 개의 부분영역으로 나누어진다고 가정하고 하나의 부분영역을 그려보면 Fig. 2와 같이 경계치 테스트에서는 부분영역을 대표 하나하나의 값을 포함하여 경계선에 있는 부분영역의 값 Min과 Max, 그리고 부분 영역 내에서 경계의 바로 안에 있는 두 값 Min+와 Max-를 테스트 데이터로 선정하는 것이 일반적이다[11].



[Fig. 2] boundary value test

2.3 테스트 개발 프로세스

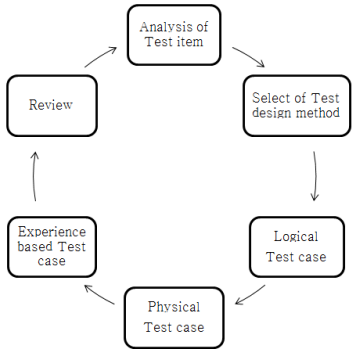
테스트 설계 과정에서 테스트 설계 기법을 이용하여 테스트 케이스와 테스트 데이터를 설계하고 명세화 한다.

테스트 케이스는 입력값의 묶음, 실행 사전조건, 기대결과(expected result)와 실행 사후조건으로 구성한다. 또한 특정 테스트 조건을 커버하기 위해 개발된다. 표준 문서인 소프트웨어 테스트 문서화 표준은 테스트 상황을 포함한 테스트 설계 명세와 테스트 케이스 명세 내용을 포함하고 있다[1].

3. 테스트 설계

3.1 테스트 설계 프로세스

본 논문에서 제시하는 테스트 설계 프로세스는 기존 엔지니어의 개인적인 능력과 경험을 기만으로 이루어진 활동들을 체계화하고 조직차원에서 관리할 수 있다[1].



[Fig. 3] Test Design Process

테스트 설계 프로세스는 Fig. 3과 같이 대상을 선정하고 테스트 설계기법을 선택하여 그 효율성을 분석하는 것으로 요약할 수 있다.

첫째, 테스트 설계 대상시스템의 범위 중 우선순위를 정의하고 대상범위의 개발 산출물, 설계서, 개발단계 테스트 결과서 등의 개발 산출물인 테스트 베이시스(test basis)를 분석한다[1]. 둘째, 대상시스템에 적용할 테스트

설계기법을 선정한다. 셋째, 논리적인 경우의 수를 고려하여 논리 테스트 케이스를 작성한다. 넷째, 테스트 케이스 자체로서 실행할 수 있는 식별자, 실행단계에 따른 기대결과를 기술한 물리적인 테스트 케이스를 작성한다. 다섯째, 고객센터의 불만사항을 분석한 케이스와 결함분석을 통하여 발견된 반복적인 경함유형의 경험적 테스트 케이스를 보완한다. 마지막으로 테스트 설계를 위해 투입된 노력대비 어떤 효율성이 있는지 분석한다[7].

3.2 테스트 설계 분석

테스트 설계 및 수행부분은 테스트 활동 중 개선필요성이 높은 부분으로 테스트 설계 및 수행활동이 요구사항, 테스트 레벨, 리스크에 따라 식별되고 관리되어야 한다. 특히, 테스트 설계의 공통적인 절차를 통하여 설계 및 실행되어야 한다[5]. 대부분의 테스트 케이스는 개발단계에서 요구사항 분석을 통하여 정리되어 구현된 단위기능들의 확인 체크리스트 형태로 테스트 실행조건이 부족하며 실행단계에 따른 기대결과가 불분명하여 결함발견 후에도 재현이 어렵거나 반복확인을 위한 시간적 리소스 낭비가 많다. 그리고 테스트를 실행하는 개인에 따라 테스트 케이스를 설계하고 확인하는 절차에 일관성이 없어 전체 테스트 범위 중 어느 정도의 테스트가 진행되었으며 요구사항이나 기능적 커버리지(coverage)에 대한 판단기준이 불분명 하다[8,11].

테스트 설계 프로세스를 기준으로 테스트 설계의 목적, 장점, 개선사항을 도출하였다. 테스트 수행은 그 절차와 단계에 있어 다양한 조직에서 수행되고 있다. 개발조직에서는 단위 테스트와 통합 테스트를 수행하고 품질관리 조직에서는 시스템 테스트와 인수테스를 준비하게 된다. 각 조직마다 테스트를 수행하는 범위에 있어 차이는 있으나 절차와 방법 및 관리가 필요하다. 따라서 조직차원의 테스트 설계 활동을 정의하고 있는 TMMi(test maturity model integration)과 같은 체계적인 접근이 필요하다[7,10]. 세부내용은 Table 1과 같다.

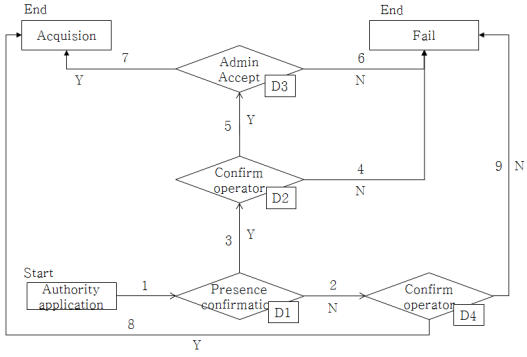
[Table 1] Test Design & Execution

Division	Test case design
Aim	Test Design is systematic, through formal technique, improvement effect of Test
Strength	Partly apply Test Design technique
Improvement	Review based requirement Risk based Test case design technique Test strategy Test design Template and guide

4. 적용 및 검증

4.1 적용

본 논문에서 테스트 케이스 설계는 국내 전자통신 기업의 지도검색 등록 관리권한 모듈을 대상으로 적용 및 검증하였다. [Fig. 4]와 같이 지도 검색 서비스 등록은 등록 사용자 수요가 많으며 권한에 따라 차등적인 서비스 이용권한을 부여하는 부분으로 운영자 확인과 현재 관리자 승인 단계를 거치게 되는 부분으로 총 9개 케이스가 존재한다.



[Fig. 4] Authority Module

제시된 관리자 승인 플로우의 경우의 수는 Fig. 4과 같이 9개의 논리적 흐름을 가진다. 테스트 케이스는 9개의 경우를 최대한 포함하면서 효과적으로 설계하는데 목적이 있다. 각각의 9가지 경우의 수는 4개의 논리적인 선택인 D1~D4의 논리적 제어 흐름에 따라 각각의 케이스가 존재한다, 각각의 케이스의 제어 흐름을 기반으로 테스트 케이스를 설계하기 위하여 제어흐름 테스트 설계 기법을 활용하였다.

초기 테스트 케이스는 관리권한자 승인절차의 전체흐름을 포한한 테스트 케이스 Table 2와 같이 5개의 테스트 케이스가 존재한다.

[Table 2] Intial Test case

Division	Test case	note
Test case	TC 1 (1, 2, 8) TC 2 (1, 2, 9) TC 3 (1, 3, 4) TC 4 (1, 3, 5, 6) TC 5 (1, 3, 5, 7)	include control flow

그러나 각각의 초기 테스트 케이스는 조건 분기점인 D1에서 D4까지의 분기 조건을 고려하지 못하고 있으며 분기점에 대한 테스트가 이루어지지 않은 경우이다. 그러므로 다음 Table 3과 같이 분기점을 중심으로 input 되는 제어흐름과 출력되는 제어흐름을 교차하여 테스트 케이스를 설계할 수 있다. 이때 가능한 경우의 수를 최대한 고려하여야 하며 사용자의 경험적인 케이스가 있다면 더욱 좋다.

[Table 3] Test case foundation of condition

Division	input	output	decision
D1	1	2, 3	(1, 2) (1, 3)
D2	3	4, 5	(3, 4) (3, 5)
D3	5	6, 7	(5, 6) (5, 7)
D4	2	8, 9	(2, 8) (2, 9)

교차한 테스트 케이스 중 테스트를 위한 논리흐름을 정리한 초기 테스트 케이스에 기본 경로가 포함되어 있는지 확인한다. D1에서 D4에 해당하는 교차점을 포함하고 있는지 확인 후 테스트 케이스를 설계함으로 논리적인 분기점을 고려한 테스트 케이스 설계가 가능하다.

분기점을 중심으로 가능한 케이스를 고려해 보면 Table 4와 같은 테스트 케이스를 설계할 수 있다.

[Table 4] Logical Test case

Division	Test case	note
Test case	TC 1 (1, 2, 8) TC 2 (1, 2, 9) TC 3 (1, 3, 4) TC 4 (1, 3, 5, 6) TC 5 (1, 2, 5, 7)	consider Decision point

Table 4에서 논리적으로 설계된 테스트 케이스를 실행하기 위하여 품질관리자 및 개발자를 포함한 인력들이 이해할 수 있는 문장으로 변환이 필요하다. 논리적 테스트 케이스를 물리적 테스트 케이스로 표현하면 아래 Table 5와 같이 표현할 수 있다. 본 사례에서 논리적 흐름을 고려한 테스트 케이스 설계를 보여준다.

[Table 5] Physical Test case

No	Div. 1	Div. 2	Test Result	Expected Result
1	Management authority	Authority application	Not management authority, confirm operator	Acquisition
2	Management authority	Authority application	Exist management authority, confirm operator	Not Acquisition
...				
3	Management authority	Authority application	Exist management authority, operator refuse Transfer	Not Acquisition
4	Management authority	Authority application	Exist management authority, confirm operator but management authority refuse Transfer	Not Acquisition

4.2 효과분석

본 논문에서 제시하는 테스트 설계 프로세스의 절차와 기법을 정보통신기업의 소프트웨어의 권한인증 프로그램에 적용하였다.

테스트 설계 효과성은 테스트 대상을 최소한의 테스트 케이스 수와 시간 및 공수로 수행하는 것이다. 테스트 설계 대상으로는 권한인증 처리절차 프로그램 모듈을 대상으로 하였으며 테스트를 수행하는 인력은 권한인증 프로세스에 대한 교육을 통하여 충분히 이해할 수 있는 시간을 가졌으며 개발경험이 없는 인력으로 배정하였다.

적용결과 기존 테스트 수행 조직이 변경관리를 위하여 수행하는 테스트 케이스를 기준으로 동일한 권한인증 프로그램을 대상으로 하였다. 테스트 케이스는 35개 감소하였으며 따라서 실행시간도 70분 감소하였다. 투입공수 기준으로는 M/M(Man/Month)로 볼 때 0.4 M/M가 감소하였다. 이는 대형 프로젝트의 제품품질 투입 인력 예측의 기초자료로 활용할 수 있다. 세부사항은 Table 6과 같다.

[Table 6] Result of experiment

Division	Before	After	Result
Test case	122	87	▽ 35
Time(minute)	240	170	▽ 70
M/M	1	0.6	▽ 0.4

*Remarks: ▽ Down

5. 결론

지금까지 테스트 케이스 설계 프로세스를 수립하고 적용하는 연구를 하였다. 기존 테스트 설계의 문제점은 테스트 설계의 공통적인 기법이나 절차가 없이 수행하는 인력의 능력과 경험에 의존하고 있으며 테스트 케이스의 재사용이 어려워 노하우가 공유되기 어렵다는 것이다.

그리고 테스트를 수행한 후에도 어느 정도의 커버리지를 달성하였는지 판단하기 위한 기준이 불명확한 경우가 많았다. 테스트 설계에 공통적인 절차와 기법을 적용함으로서 객관적인 커버리지를 보장하고 테스터의 테스트 설계 능력을 향상시킬 수 있다.

조직이 소프트웨어 품질관리의 자산으로 테스트 케이스를 개선하고 설계하는 과정에서 인력의 능력향상을 가져올 수 있으며 품질기준의 계량화로 정량적 품질관리 기반을 마련하고 나아가 소프트웨어 프로세스 및 제품품질 향상에 기여할 수 있다.

본 논문은 테스트 케이스 설계 절차를 체계화 하여 품질관리 하고자 하는 기업 및 연구소 등의 조직에 적용 및 활용이 가능하며 적용 시 조직의 개발 프로세스를 충분히 고려할 때 효과를 극대화 시킬 수 있다.

References

[1] NIPA, STA Testing, “Software test practice guide", STA. 2012.

[2] James Bach, "James Bach on Risk-Based Testing", 1999.

[3] ISTQB, Certified Tester Advenced Level Syllabus, 2007.

[4] ISTQB, Certified Tester Foundation Level Syllabus, 2007.

[5] Erik Van Veenendaal, "Test Maturity Model Integration(TMMi)", Version 1.0 (February, 17 th 2008) Produced by The TMMi Foundation, 2008.

[6] Arnaud Dupuy, Alcatel, France, Nancy Leveson, MIT, USA," An Empirical Evaluation of MC/DC Coverage Criterion on the HETE-2 Satellite Software, 2001.

[7] Kwon, Yong Rae, "software testing", Saeng Neung Press, 2010.

[8] Kwon, Wonil , “Software testing practice”, 2007.

[9] Ron Patton, “Software Testing", Global press. 2003.

[10] ISO/IEC 29119 Test Process, 2008.

[11] Glenford J. Myers, Corey Sandler, “The Art of Software Testing”, JohnWiley&SonsInc 2004.

장 진 욱(Jin-Wook Jang) **[정회원]**

- 2013년 2월 : 건국대학교 신산업융합학과 (경영공학박사)
- 1999년 2월 ~ 2005년 2월 : 국방부 정보사령부 전산장교 (대위)
- 2011년 4월 ~ 2013년 2월 : SK communications Manager
- 2013년 2월 ~ 현재 : 건국대학교 정보통신대학 인터넷미디어공학부 산학교수

<관심분야>

Software Quality & Testing, Project Management