테스트 및 운영중 소프트웨어의 오류 원인 분석

Software Cause Analysis during Testing and Operation Stages

최규식
건양대학교 의공학과

Che Gyu-Shik
Konyang Univ.

요약

H/W와 달리 S/W는 개발당시부터 그 속에 오류로 존재하고 있다가 어떤 특수한 기능을 필요로 할 때 드디어 검출되게 된다. 이로 인하여 소프트웨어의 고장이 발생할 수 있다.
본 논문에서는 소프트웨어의 테스트 단계 및 운영단계에서 검출되는 오류 및 이로 인한 고장 원인을 분석한다. 테스트 단계와 운영단계중 어느 쪽이 어떤 오류가 더 많이 발생하는가를 현장 경험을 통한 오류데이타 수집에 의하여 분석한다.

Abstract

The S/W error is in the S/W from the development stage other than H/W, and it may come out when the specific function is necessary. The S/W failure may happen due to these errors.
I analyze causes of the fault detected during testing stage and operational stage, and fgailure caused by it. And then, I analyze which case of testing stage and operational stage is much severe than the other case by real failure data acquisition.

I. 서 론

H/W 고장은 마모현상이나 우연성에 의해서 유발되지만 S/W 고장은 S/W가 개발 당시부터 오류로서 현존하고 있다가 어떤 특수입력자료에 의해 기능을 필요로 할 때 비로소 나타나게 된다. 오류가 있다고 해서 전부 고장이 발생되는 것은 아니며, 특정한 자료가 입력되지 않아서 영원히 발견되지 않을 수도 있다. Shooman은 S/W의 신뢰성을 "예정 기간 동안 S/W가 제시된 규격에 따라 성공적으로 그 기능을 다할 확률"로 정의하였다. 반대로 합리적으로 그 기능을 다하지 못할 때, 이를 고장이라 하고 그 원인을 S/W의 오류라고 정의하였다. 또한, 오류가 컴퓨터에 미치는 영향에 따라 major error와 minor error로 분류하였다. Schneidwind는 S/W의 오류를 설계상의 오류, 프로그램 작성시의 오류, 문서 취급상의 오류, 검정과정에서 발생하는 오류 등으로 오류의 발생시기를 기준으로 분류하였다.

II. S/W의 신뢰성 관리 역사

S/W의 신뢰성과 품질이 문제가 된 것은 IBM360/OS, Multics, TSS/360이 개발된 1960년대이다. 당시 4,000 모듈, 10^6 명령어 규모로 연인원 5,000명과 막대한 비용을 투입한 OSI360, FORTRAN-IV 컴파일러에서 약 2,000개의 오류가 발견되면서 S/W의 신뢰성에 대한 문제가 심각하게 대두되었다. 여기서 컴퓨터 S/W의 테스트 방법에 대한 연구와 현황이 연구되었는데, 그 결과 표 2-1에서 보는 바와 같이 테스트단계에서 발견된 코딩 오류보다는 설계단계에서 더 많은 오류가 첨가된다는 것이었다.

[표 2-1] 발견되는 S/W의 원인 분류

S/W 프로젝트	설계오류(%)	코딩오류(%)
A	73.6	26.4
B	73.7	26.3
C	35.6	64.4
D	51.6	48.4
E	58.8	41.2
F	61.9	38.1
G	65.8	34.2

이로부터 오류방지를 위한 설계-프로그래밍, 다큐멘테이션의 새로운 기법과 방법론이 제시되었다. 1973년에 이르러 IEEE에 의해서 컴퓨터 S/W 신뢰도에 대한 학술토론을 시발로 1975년 S/W 공학이라는 새로운 분야가 개척되면서 S/W의 신뢰성문제가 정식으로 제기되었다. 1978년 Computer S/W and Applications Conference에서 S/W 프로젝트에 관한 회의가 있었다. 여기서, S/W의 품질 및 생산성에 관한 20개의 중요한 항목에 대해 문제점이 제기되었다.

III. 고품질 S/W

종래의 S/W에 관한 것은 주로 신뢰성과 효율성에 관한 것이었으나, 최근에는 다방면에 걸친 요구가 제기되고 있다.

1. 사용성

S/W가 그 상태에서 어느 정도 사용될 수 있느냐의 정도이다. 이것에는 신뢰성이 높고 효율이 좋으며, 조작성이 좋아야 한다는 성질이 만족되어야 함.

2. 신뢰성

S/W가 정해진 상태에서 정해진 기간 내에 정해진 기능을 발휘할 수 있는가 하는 정도로서 자기 포괄성, 정확성, 안전성, 견고성, 통합성, 무모순성 등이 요구됨

3. 효율

S/W가 자원을 낭비하지 않는 범위 내에서 목적을 달성할 수 있는 정도. 계측성, 장치효율, 접근가능성 등이 포함됨.

4. 조작성

사용자의 시간이나 정력을 낭비하지 않기 위해 얼마나 조작하기 쉬운가 하는 특성. 견고성, 통합성, 접근 가능성, 전달성 등에 의해 좌우됨.

5. 보수성

S/W가 사용환경 등에 의해서 생기는 변경요구를 얼마나 용이하게 받아들일 수 있는가? 또는 운전중에 발견된 S/W의 결함을 수정하기 용이한가 하는 성질. 여기에는 테스트 용이성, 이해성, 갱신성 등이 포함됨.

6. 테스트 용이성

S/W의 테스트나 성능의 평가가 용이한 정도를 말함. 이 특성은 계측성, 접근 가능성, 전달성, 자기 기술성, 구조성 등의 요소에 의해 영향을 받음.

7. 이해성

S/W가 어느 정도 이해하기 쉽도록 되어있는가 하는 정도를 말함. 무모순성, 자기기술성, 구조성, 간결성, 명확성 등이 요구됨.

8. 갱신성

S/W의 변경이 용이한 정도로서 이것은 구조성과 확장성의 특성에 관련됨.

IV. S/W 오류의 원인분석

Thayer의 연구에서는 표 4-1과 같은 4가지 프로젝트에서 오류를 수집하고 오류분석 자료를 표 4-2와 같이 보고하고 있다. 표 4-3은 프로젝트별 오류 발생 빈도 순위를 보여준다. 이 표에서 보면 프로젝트 A, B, C, D 공히 논리 오류가 가장 큰 부분을 차지하고 있음을 알 수 있다. 여기서, 논리오류는 사양서에 기술된 기능에 대응하는 코드부분이 존재하지 않거나 처리과정과 정보판정조건이 틀리는 경우이고, 연산오류는 처리과정 내의 연산과정이 틀린 경우이다. 논리오류의 경우를 보면 전체 오류정보의 26%를 차지하고 있는 바, 그 88%의 오류가 설계시 원인이 있고 12%는 코딩시 원인이 있다.

[표 4-1] 프로젝트의 특성

특징	프로젝트 A	프로젝트 B	프로젝트 C	프로젝트 D
언어	JOVIAL J4	JOVIAL J4	PWS 마크로	FORTRAN ASSEMBLY
루틴수	173	249	190	531
Source수	96,931	115,346	(주1)	11,105(F) 17,459(A)
formal한 요구정의	기능레벨	기능레벨	S/W시스템 레벨	루틴레벨
문서화표준	SSD Exhibit 51-47B	SSD Exhibit 61-47B	TRW사 표준	MIL표준 490
공정개발자	없음	없음	없음	없음
시스템작동모드	배치	배치	온라인/배치	리얼타임/배치
formal 테스트	개발테스트 종합시스템테스트	개발테스트 종합시스템테스트 데몬스트레이션	-	루틴개발테스트 루틴종합테스트 프로세스테스트 성능평가, 데몬스트레이션
데이타수집	시스템 확장시 확장 IA-확장(주2) 응용S/W부의 오류	테스트기간 운용시	운용시	개발시(Top down 및 구조화프로그래밍)
S/W의 종류	지령제어시스템	지령제어시스템	데이타관리시스템	응용 S/W simulator operating system

[표 4-2] 중요오류발생상황(프로젝트 A,B,C)

오류범주	프로젝트 A									프로젝트 B		프로젝트 C	
	확장1A		확장1B		확장PR		확장2		합계				
	수	%	수	%	수	%	수	%	수	수	%	수	%
연산	19	7.2	26	7.2	21	13.7	96	13.3	162	353	8.0	7	1.3
논리	37	14.0	51	14.2	31	20.2	137	19.1	256	937	21.1	140	1.3
I/O	38	14.3	41	11.4	11	7.2	66	9.2	156	719	16.2	36	26.0
데이타취급	40	15.1	25	7.0	26	16.9	73	10.1	164	613	13.8	110	6.7
OS/지원S/W	0	0	0	0	1	0.6	0	0	1	4	0.1	0	0
구성	5	1.9	4	1.0	3	1.9	4	0.6	16	83	1.9	0	0
루틴간인터페이스	12	4.5	11	3.1	5	3.2	41	5.7	69	250	5.6	33	6.1
루틴/시스템간인터페이스	0		0	0	0	0	5	0.7	5	28	0.6	0	0
테이프처리인터페이스	0	0	0	0	0	0	1	0.1	1	9	0.2	9	1.7
사용자·인터페이스	21	0	44	12.3	15	9.7	34	4.7	114	334	7.5	34	6.3
데이타베이스·인터페이스	4	1.5	1	0.3	2	1.3	3	0.4	10	21	0.5	15	2.8
사용자변경요구	0	0	0	0	0	0	0	0	0	0	0	111	20.6
데이타베이스세트	15	5.3	25	7.0	12	7.8	60	8.3	111	370	8.3	9	1.7
공정변수정의	10	3.8	16	4.5	4	2.6	26	3.6	56	50	1.1	12	2.2
재발	7	2.6	7	1.9	2	1.3	8	1.1	24	80	1.8	0	0
문서화	30	11.3	68	18.9	11	7.2	63	8.8	172	196	4.4	23	4.2
요구불만족	2	0.8	2	0.6	0	0	6	0.8	10	47	1.1	0	0
식별불가	5	1.9	16	4.5	9	5.8	50	6.9	80	178	4.0	0	0
오퍼레이타	21	7.9	29	5.5	1	0.6	44	6.2	86	118	2.7	0	0
불명확	0	0	2	0.0	0	0	3	0.4	5	49	1.1	0	0
총오류	265		359		154		720		1498	4439	1.1	539	

논리오류 중 설계 표준과 대조하여 52.7%가 검출 가능하고 코딩 표준과 대조하여 31.5%, 설계 지침서에 의해서 86.5%, 코딩 지침서에 의해서 78.9%가 검출 가능하다. 기능과정 테스트에서는 79.6%, 특이치 테스트에서는 57.3%, 통합 테스트에서는 24.2%, 요구과정 테스트에서는 52.3%가 검출가능하다. 단, 알고리즘에서는 논리오류가 전혀 검토되지 않았음을 보여주고 있다.

[표 4-3] 오류발생빈도순위

순위	프로젝트 A	프로젝트 B	프로젝트 C	프로젝트 D			
				응용S/W	시뮬레이터	OS	PA방법
1	논리	논리	논리	DB	논리	논리	논리
2	인터페이스	데이타취급	데이타취급	논리	연산	데이타취급	데이타취급
3	데이타취급	인터페이스	인터페이스	연산	DB	I/O	I/O
4	연산	I/O	I/O	데이타취급	데이타정의	인터페이스	데이타정의
5	I/O	연산	데이타정의	인터페이스	I/O	데이타정의	DB
6	DB	DB	DB	I/O	데이타취급	DB	인터페이스
7	데이타정의	데이타정의	연산	데이타정의	인터페이스	연산	연산

V. S/W의 오류 특성

지금까지의 오류분석결과를 요약하면 오류의 특성을 다음과 같이 정리할 수 있다.

- S/W 오류형태에 대한 발생분포는 S/W의 종류에 따라 차이가 난다.
- S/W 오류형태에 대한 발생분포는 S/W의 규모에 따라 차이가 난다.
- S/W 개발 사이클과 발생된 오류 형태의 경향에 차이가 있다.
- 대규모 S/W 오류의 대다수가 S/W의 초기단계에서 발생되고 있으며, 요구오류, 설계 오류가 많다.
- S/W 오류는 라이프 사이클의 각 단계에서 계속 발생해서 검출되지 않은 상태로 다음 단계로 이월되는 경향이 크다. 특히 요구 오류가 남아 있을 경우에는 큰 영향을 미치게 된다.
- S/W 오류검출 수정비율은 오류가 삽입된 시점으로부터 지연되면 될수록 높아지게 된다.
- S/W 오류 수정시 새로운 오류가 발생하는 것도 큰 비율을 차지한다.
- S/W 오류 검출 분포에는 일정한 패턴이 없다.
- Top down 개발에서의 오류발생 현상은 top으로부터 bottom으로 레벨이 이전함에 따라서 오류 형태로 발생분포가 변화한다.
- 시스템 확장은 반복하는 프로젝트에서의 오류 발생 형태에 따라 변한다.
- 어떤 오류에 대해서 어떤 검출 수단이 유효한가 판명되었다.
- S/W 오류를 검출하는 수단이 이미 존재하고 있다.
- S/W 개발 전체에 대한 테스트 작업비용, 분포 경향이 드러나고 있다.
- S/W 개발의 제 1단계인 S/W 요구정의에 많은 문제와 원인이 집중되고 있다.

VI. S/W 신뢰도모델

1. S/W 신뢰도모델의 특징

S/W의 경우는 프로그램 내에 언제나 버그가 존재한다는 것이다. 이 가설을 역으로 하면 고장은 특정한 데이타가 특정한 버그를 포함한 경로를 지남으로써 발생하므로 프로그램 내에 버그가 있어도 입력 데이타가 버그를 건드리지 않으면 고장이 발생하지 않는다. 이 버그가 있다는 가설에 의해 프로그램을 계속 수행하게 되면 언젠가는 고장을 유발시킬 수 있다는 것이 S/W 신뢰도의 근본원리이다. 이런 측면에서 S/W 신뢰도를 재정의하면 S/W 신뢰도란 “주어진 기간에 정의된 환경 하에

서 미리 정의된 사항 이외의 것에 의해 요구되는 기능과 다른 결합이 발생되지 않을 확률"을 말한다.

2. S/W 신뢰도 모델 분류

지난 35년간 S/W 신뢰도 모델, S/W 신뢰도 특성 및 신뢰도 모델의 검증에 관해서 약 300여 편에 달하는 연구논문이 발표되었다. S/W의 특수성에 의해 이들 모델도 각기 상이한 시각을 가지고 있으므로 모델들의 가정과 배경에 따라 두 가지 분류체계로 분류한다.

3. S/W 신뢰도 모델의 문제점과 미래의 연구방향

기존 신뢰도모델의 문제점과 앞으로 나아가야 할 방향을 정리해보면 아래와 같다.

첫째, 실제 시험과정을 고려한 입력영역 S/W 신뢰도모델이 더 많이 연구되어야 한다. 기존의 시간 영역모델은 실제 개발에 적용하기에는 현실성이 다소 결여되어 있다.

둘째, 대부분의 신뢰도 모델들은 일반 개발자들이 이해하기 힘든 단점이 있는데 이는 모델 개발시 너무 이론에 치중한 탓이다. 이로 인해 모델 선정시 세운 가정과 개발시 적용하는 시험 전략과는 커다란 차이가 생기게 된다. S/W 신뢰도 모델이 개발에 쉽게 적용되기 위해서는 S/W의 전 수명주기에 걸쳐 특히 시험환경에 입각한 이해하기 쉬운 신뢰도 모델 개발이 시급하다.

마지막으로, 신뢰도를 정량적으로 표현할 때 기존의 신뢰도 모델들은 버그 숫자 개념에 일관되어 왔다. 이것이 잘못되었다는 것이 아니라 이 개념만 가지고는 진정한 신뢰도를 나타낼 수 없는 경우가 있다. 예를 들면 논리 자체에 문제가 있어 개발된 S/W를 재설계해야 하는 경우 단순한 버그의 숫자는 별 의미가 없다.

VII. 결 론

S/W의 신뢰성과 품질이 문제가 된 것은 IBM360/OS, Multics, TSS/360이 개발된 1960년대이다. OSI360, FORTRAN-IV 컴파일러에서 약 2,000개의 오류가 발견되면서 S/W의 신뢰성에 대한 문제가 심각하게 대두되었다. 여기서 컴퓨터 S/W의 테스트 방법에 대한 연구와 현황이 연구되었는데, 그 결과 테스트단계에서 발견된 코딩 오류보다는 설계단계에서 더 많은 오류가 첨가된다는 것을 알게 되었다.

이로부터 오류방지를 위한 설계-프로그래밍, 다큐멘테이션의 새로운 기법과 방법론이 제시되었다. 오류중에는 논리 오류가 가장 큰 부분을 차지하고 있음을 알게 되었다.

▌참 고 문 헌▐

[1] K. Okumoto, A.L. Goel, "Optimum release time for software systems based on reliability and cost criteria", J. System Software, Vol.1, 1980, pp.315-318

[2] H.S. Koch, P. Kubat, "Optimal release time of computer software", IEEE Trans. Software Eng'g, Vol.SE-9, 1983 May, pp.323-327

[3] M. Xie, "On the determination of optimum software release time", Proc. 2nd Int' Symp. Software Reliability Eng'g, 1991, pp.218-224

[4] S. Yamada, S. Osaki, "Cost-reliability optimal release policies for software systems", IEEE Trans. Reliability, vol R-34, 1985 Dec., pp.422-424

[5] Rong-Huei Hou, Sy-Yen Kuo, Yi-Ping Chang, "Optimal Release Policy for Hyper-Geometric Distribution Software-Reliability Growth Model", IEEE Trans Reliability, Vol.45, 1996 Dec., pp.646-651