소프트웨어의 오류 원인 분석

최규식
건양대학교
{che}@konyang.ac.kr

Test Resources Allocation for SRGM

Che Gyu Shik
Konyang University

요 약

최근 운영시스템, 제어프로그램, 적용프로그램과 같은 여러 가지 소프트웨어 시스템이 더욱 더 복잡화 및 대형화되고 있기 때문에 신뢰도가 높은 소프트웨어 시스템을 개발하는 일이 매우 중요하며, 따라서 소프트웨어 제품 개발에 있어서 소프트웨어의 신뢰도가 핵심사항이라고 할 수 있다. 소프트웨어가 주어진 시간동안 고장이 발생하지 않을 확률 즉, 신뢰도는 소프트웨어의 테스트 과정을 계속하면서 반복해서 결함을 발견 및 수정하면 더욱 더 향상될 것이다. 그러한 검출현상을 설명해주는 소프트웨어 신뢰도 모델을 소프트웨어 신뢰도 성장모델(SRGM)이라 한다.

1. 서론

1972년 Jelinski와 Moranda의 "Software Reliability Research"에 의해서 제기된 소프트웨어의 신뢰도 이론 후 소프트웨어의 신뢰도에 대한 수많은 논문이 발표되었다. 그 이후 소프트웨어의 전 주기에 대한 비용개념이 도입되면서 연구는 급진전되는 단계로 접어들었다. 예로 Okumoto와 Goel은 전체평균 소프트웨어 비용을 최소화시키는 비용-최적 SRP(software reliability process)를 발표하였다. Yamada와 Osaki는 전체 평균 비용을 최소화시키고 소프트웨어 신뢰도를 만족시키는 전체평균비용-신뢰도-최적 SRP를 도입하였다. 이러한 연구 결과를 참조하여 Hou, Kuo, Chang은 지수 곡선과 로지스틱 곡선에 적용하는 연구를 수행하였다.

본 논문에서는 소프트웨어의 신뢰도가 도입된 배경을 우선 조사하고 그 신뢰도 개념에 대해서 연구한다.

2. S/W의 신뢰도

H/W 고장은 마모 현상이나 우연성에 의해서 유발되지만 S/W 고장은 S/W가 개발될 당시 오류로서 내재하고 있다가 어떤 특수 입력 자료에 의해 발견될 때 비로소 나타나게 된다. 오류가 있다고 해서 전부 고장이 발생되는 것은 아니며, 특정한 자료가 입력되지 않아서 영원히 나타나지 않을 수도 있다. 여기서, S/W의 신뢰도란 이러한 오류가 발생되지 않고, 즉 고장이 나지 않고 계속 운전될 수 있느냐 하는 것이다. 이것은 곧 S/W의 내부에 얼마나 많은 오류가 잠재해 있느냐 하는 것에 비례할 것으로 생각된다. 이러한 의미에서 S/W의 신뢰도 관리는 어떻게 하면 이러한 오류를 S/W의 개발 단계에서 발생하지 않도록 예방하느냐 하는 것과 이렇게 개발된 S/W 중의 오류를 어떠한 테스트 방법을 통해서 검출할 것이냐, 그리고, 어느 정도 검출이 되었을 때 오류를 예측하고 그 예측에 근거하여 S/W를 발행하는 것이냐 하는 문제가 된다. Shooman은 S/W의 신뢰도를 "예정 기간동안 S/W가 제시된 규격에 따라 성공적으로 그 기능을 다 할 확률"로 정의하였다. 반대로 합리적으로 그 기능을 다하지 못할 때 고장이라고 하고 원인을 S/W의 오류라고 정의하였다. 또한, 오류가 컴퓨터에 미치는 영향에 따라 major error와 minor error로 분류하였다. Schneidwind는 S/W의 오류를 설계상의 오류, 프로그램 작성상의 오류, 문서 취급상의 오류, 검정과정에서 발생하는 오류 등으로 오류의 발생시기를 기준으로 분류하였다.

2.1 S/W의 신뢰도 관리 역사

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

여기서, 신뢰성 있는 S/W의 개발은 코딩 단계보다는 설계 단계에서 고려할 중요한 요소임이 인식되기 시작했고 오류 방지를 위한 설계-프로그래밍, 다큐멘테이션의 새로운 기법과 방법론이 제시되었다. 1973년에 이르러 IEEE에 의한 컴퓨터 S/W 신뢰도에 대한 학술토론을 시발로 1975년 S/W 공학이라는 새로운 분야가 개척되면서 S/W의 신뢰도 문제가 정식으로 제기되었다. 1978년 COMPSAC(Computer S/W and Applications Conference)에서 S/W 프로젝트에 관한 회의가 있었다. 여기서, S/W의 품질 및 생산도에 관한 다음과 같은 20개의 중요한 항목에 대해 문제점이 제기되었다.

표 2-1 테스트중에 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

2.2 고품질 S/W

종래의 S/W에 관한 것은 주로 신뢰도와 효율성에 관한 것이었으나, 최근에는 다방면에 걸친 요구가 제기되고 있다. 예를 들면 Wulf는 S/W의 품질 특성으로서 보수성, 갱신성, 견고성, 이해성, 성능, 휴대성, 조작성 등을 연구하고 있다. 이 외에도 여러 가지가 있으며, Boehn등에 의해서 단계적인 품질 특성이 전개되었다.

2.3 S/W 오류의 원인분석

지금까지 언급한 바와 같이 S/W의 신뢰도는 그 S/W가 가지고 있는 오류에 의해서 연유되므로 이러한 오류의 원인을 분석하는 것은 매우 의의가 깊다. Thayer의 연구에서는 표 2-2와 같은 4가지 프로젝트에서 오류를 수집하고 오류분석 자료를 표 2-3과 같이 보고하고 있다.

표 2-2 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/W시스템레벨	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	지령제어 시스템	지령제어 시스템	데이터관리 시스템	응용 S/W simulator operating system

각 프로젝트의 오류 정보는 우선 프로젝트 A, B, C에 대해서 수집하고, 그 분류 결과에 의해서 프로젝트 D의 오류정보를 수집 및 해석하고 있다. 표 2-4에서는 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 정보 판정 조건이 틀리는 경우이고, 연산오류는 처리과정 내의 연산과정이 틀린 경우이다.

표 2-3 중요 오류 발생 상황(프로젝트 A, B, C)

오류범주	프로젝트 A									프로젝트 B		프로젝트C	
	확장1A		확장1B		확장IPR		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	

표 2-4 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

프로젝트 B를 대상으로 하여 이 것을 분석한 결과가 표 2-5와 같다. 논리 오류의 경우를 보면 전체 오류정보의 26%를 차지하고 있는 바, 그 88%의 오류가 설계시 원인이 있고 12%는 코딩시 원인이 있다.

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 논리 오류가 전혀 검토되지 않았음을 보여주고 있다.

2.4 S/W의 오류 특성

지금까지의 오류 분석 결과를 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 수명 주기의 각 단계에서 계속 발생해서 검출되지 않은 상태로 다음 단계로 이월되는 경향이 크다. 특히 요구 오류가 남아 있을 경우에는 큰 영향을 미치게 된다.

○ S/W 오류 검출 수정비율은 오류가 삽입된 시점으로부터 지연되면 될수록 높아지게 된다.

표 2-5 프로젝트 B 오류데이타 VS 검출수단

오류범주	총오류내의 비율	설계표준	코딩표준	설계지침서	코딩지침서	기능공정테스트	복이치테스트	알고리즘테스트	통합테스트	요구공정테스트
연산	9.0	53.9	15.9	65.4	46.7	87.4	52.2	73.6	0	62.6
논리	26.0	52.7	31.5	86.5	72.9	79.6	57.3	0	24.2	52.3
I/O	16.4	21.7	23.5	21.7	78.6	98.5	35.2	0	50.0	57.8
데이타취급	18.2	2.7	51.0	24.4	73.9	84.2	76.9	0	63.0	58.8
OS/지원S/W	0.1	-	-	-	-	-	-	-	-	-
구성	3.1	11.1	0	0	0	0	0	0	0	0
루틴간 인터페이스	8.2	47.9	41.2	78.2	86.1	20.0	0	0	99.4	7.9
루틴/시스템간 인터페이스	1.1	27.3	0	80.0	0	100.0	0	0	100.0	0
데이프처리 인터페이스	0.3	0	0	0	46.7	87.4	52.2	73.6	0	62.6
사용자 인터페이스	6.6	12.7	0	24.6	0	89.6	87.3	0	100.0	12.7
DB세트	0.8	0	0	0	0	100.0	0	0	100.0	0
0사용자변경	0	-	-	-	-	-	-	-	-	-
DB세트	4.1	0	0	0	53.7	100.0	29.3	29.3	13.4	61.0
공통변위	0.8	60.0	12.5	60.8	81.3	87.5	81.3	62.5	100.0	0
재발	1.3	-	-	-	-	-	-	-	-	-
문서화	0.8	-	-	-	-	-	-	-	-	-
요구불만족	0.4	0	0	75.0	75.0	75.0	0	0	25.0	100.0
식별불가	1.0	-	-	-	-	-	-	-	-	-
오퍼레이타	0.7	-	-	-	-	-	-	-	-	-
불명확	1.1	-	0	-	-	-	-	-	-	-

○ 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.

○ 반복하는 프로젝트에서의 오류 발생 형태에 따라 시스템 확장이 달라진다.

○ 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.

3. S/W 신뢰도 모델의 특징

S/W는 프로그램 내의 오류에 의해 고장이 발생되고 경과되는 시간보다는 입력되는 데이타의 성질에 따라 고장을 일으킨다. 다시 말해 사용하지 않거나 같은 데이터를 계속 입력하는 경우에는 더 이상의 고장이 발생되지 않는다.

S/W에서 한 가지 재미 있는 가설은 프로그램 내에는 언제나 오류가 존재한다는 것이다. 이 가설을 역으로 하면 고장은 특정한 데이타가 특정한 오류를 포함한 경로를 지남으로써 발생하므로 프로그램 내에 오류가 있어도 입력 데이타가 오류를 건드리지 않으면 고장이 발생하지 않는다. 이 오류가 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 하에서 미리 정의된 사항 이외의 것에 의해 요구되는 기능과 다른 결합이 발생되지 않을 확률"을 말한다. 이것을 수학적으로 아래와 같이 표현한다.

4. 결론 및 미래의 연구 방향

현대적인 관리 방법 및 분석 기법은 누구나 다 인식하고 있으나, 실제 S/W 신뢰도 개발에 신뢰도 개념을 도입한 과학적인 기법을 도입하는 데에는 인색한 것이 현실이다. 신뢰도 측정은 먼 이웃나라에서 여유 있는 사람들이나 하는 것으로 간주되고 기본 요구사항만 만족시키는 방향으로 치닫는 것은 개발에 급급한 나머지 이용자 입장을 고려하지 못하는 처사라고 할 수 있다. 개발자의 보다 적극적인 신뢰도 개념도입과 아울러 S/W 신뢰도를 연구하는 사람도 이론에 치우친 모델 개발보다는 좀더 실용적인 직접개발에 적용할 수 있는 포괄적인 신뢰도 연구가 이루어져야 할 것이다.

참고문헌

[1] 강인선, 김진규, "신뢰성 공학", 한올출판사, 1997, pp183-204

[2] 이치우, 김선진, 이성우, 정상영, "신뢰성 공학", 원창출판사, 1993, pp21-124

[3] H. Ascher and H. Feingold, *Repairable Systems Reliability: Modeling, Inference, Misconceptions, and Their Causes*. New York: Marcel Dekker, 1984.

[4] W. D. Brooks and R. W. Motley, "Analysis of discrete software reliability models," Rome Air Development Center, New York, Tech. Rep. RADC-TR-80-84, 1980.