게임 소프트웨어의 확인 및 검증에 대한 신뢰도 영향 분석※

손한성, 노창현
중부대학교 게임학과
{hsson, chroh}@joongbu.ac.kr

Reliability Effect Analysis for Game Software Verification and Validation

Han-Seong Son, Chang-Hyun Roh
Dept. of Game Engineering, Joongbu University

요 약

게임 서비스를 위한 소프트웨어의 경우 그 신뢰도에 대한 중요성은 지속적으로 증가하고 있다. 따라서 소프트웨어 신뢰도에 대한 평가 또한 매우 중요하다. 본 연구에서는 개발 공정에 대한 대표 활동인 확인 및 검증이 소프트웨어 신뢰도에 어떠한 영향을 미치는가를 정량적으로 분석하기 위하여 실험을 수행하였다. 이러한 실험 결과는, BBN (Bayesian Belief Network) 기반 신뢰도 평가와 같이, 개발 공정에 근거하여 신뢰도를 평가할 때 매우 유용한 근거 자료로 활용될 것이다.

ABSTRACT

Since the importance of software reliability for game service increases continuously, the reliability evaluation becomes very important. This research performed an experiment which was intended to analyze the effect of software verification and validation, a representative activity of the software development process, on the software reliability. The results from the experiments provided the reliability evaluation based on the development process (e.g., Bayesian Belief Network based reliability estimation) with very useful bases.

Keywords : Software(소프트웨어), Reliability(신뢰도), Verification and Validation (확인 및 검증), Game Service (게임 서비스)

접수일자 : 2011년 11월 16일 일차수정 : 2011년 12월 01일 심사완료 : 2011년 12월 08일
교신저자(Corresponding Author) : 노창현
※ 본 연구는 한국연구재단의 지원을 받아 수행된 연구 결과입니다.

1. 서 론

게임 서비스를 위한 소프트웨어의 경우 그 신뢰도는 매우 중요하며 소프트웨어 신뢰도의 저하로 인한 손실 또한 지속적으로 증가하고 있는 추세이다. 따라서 소프트웨어 신뢰도를 철저하게 관리하고 평가하는 것은 게임 서비스 전체 품질의 향상을 위해서 매우 중요하다고 할 수 있다[1,2].

소프트웨어 신뢰도 평가는 크게 개발 공정에 대한 평가와 개발 제품에 대한 평가로 이루어질 수 있다[3]. 개발 공정에 대한 평가는 소프트웨어 개발자들의 품질과 공정상의 다양한 활동의 품질 등이 최종 소프트웨어 제품의 품질에 영향을 미친다는 것을 전제로 한다. 한편, 개발 제품에 대한 평가는 시험 및 유사 제품 사용 이력 등을 통해 수집된 오류 발생 수 등과 같은 각종 신뢰도 자료를 분석하여 신뢰도를 평가하는 것이다.

본 연구에서는 소프트웨어 개발 공정의 대표 공정 활동인 확인 및 검증이 소프트웨어 신뢰도에 어떠한 방식으로 영향을 미치는가를 정량적으로 분석하기 위하여 실험을 수행하였다. 확인 및 검증이 소프트웨어 신뢰도에 긍정적으로 영향을 미친다는 것은 주지의 사실이다. 그러나 확인 및 검증을 적용한다는 사실 자체가 소프트웨어 신뢰도를 향상시키는가, 아니면 확인 및 검증 공정이 잘 조직되고 잘 적용될 경우에 한해 소프트웨어 신뢰도를 향상시키는가에 대해서는 구체적인 증거가 필요한 상황이다. 또한 소프트웨어 개발 공정에서 확인 및 검증 공정 활동에 대한 자원의 배분을 어떻게 하는 것이 신뢰도 관점에서 바람직한 것인가에 대한 분석이 필요하다. 본 연구에서는 이러한 점들을 분석하기 위한 실험을 수행하였으며, 이 실험 결과는, Bayesian Belief Network(BBN) 기반 신뢰도 평가와 같이 개발 공정에 근거하여 신뢰도를 평가하고자 할 때[4,5], 매우 유용한 근거 자료로 활용될 수 있을 것이다. BBN 기반 신뢰도 평가는 확률분포를 가지는 변수들 사이에 정성적인 인과관계 존재함을 모델링하여 시스템의 신뢰도를 평가하는 기법으로서, 이 기법을 소프트웨어 신뢰도 평가에 적용하고자 하는 시도들이 최근에 많이 있기 때문에 본 실험이 중요하다고 하겠다.

한편, 소프트웨어의 결함을 찾아내기 위한 방법들의 효과를 비교하기 위한 실험은 기존에도 많이 수행된 바 있다[6,7]. 그러나 이러한 실험들은 서로 다른 프로그램을 사용하여 수행되었기 때문에 각 방법의 효율성을 객관적으로 비교할 수는 없는 상황이다. 본 연구에서 수행한 실험은 동일한 두 프로그램을 대상으로 하여 유사한 실력을 가진 실험 참가자들로 동일한 확인 및 검증 기법을 적용하면서 오류 발생 가능성을 비교 분석하기 위한 실험이다. 본 실험을 통해서는 오류 발생 가능성을 감소시키는 정도를 파악할 수 있으며 어떻게 확인 및 검증 활동을 조직하는 것이 소프트웨어의 결함을 발견하는데 더 효과적인가를 알 수 있다.

2. 본 론

2.1 팀 구성 및 소프트웨어 개발

본 실험에서는 두 개의 소프트웨어 개발 팀을 구성하여 두 종류의 소프트웨어를 개발하도록 한 후, 확인 및 검증 교육 전후의 오류 발생 수의 차이, 오류 종류 별 발생 수의 차이 등을 분석하였다. 실험에 참가하는 소프트웨어 개발 팀원들은 게임 및 소프트웨어 분야를 전공하는 18명의 대학교 3학년 학생들로서 프로그래밍 경험이나 실력이 비슷한 수준이었다. 실험에서 개발한 소프트웨어는 공정제어 계통을 모사하는 소프트웨어와 오델로 게임 소프트웨어이며, 두 소프트웨어의 복잡도 및 특성은 유사하였다. 실험 결과, 확인 및 검증 교육 전에는 두 개발 집단에서 개발한 소프트웨어의 오류 발생 수가 통계적으로 유사한 것으로 평가되어, 두 소프트웨어가 개발 난이도 측면에서 유사하다는 것을 뒷받침하였다.

실험 결과의 객관성 확보를 위해 표본의 크기를 30명 이상으로 하여 실험을 하는 것이 일반적이다

[8]. 그러나 본 연구에서 수행한 실험의 경우에는 표본의 크기를 18명으로 하였다. 이는 본 실험에서 특히 실험연구의 내적타당도(Internal Validity)를 중요하게 고려하였기 때문이다[8]. 내적타당도는 실험처치의 결과가 정말로 실험처치의 효과에 기인한 것으로 볼 수 있는가의 문제이다. 피실험자가 많은 경우 가외변인에 대한 통제가 어려워 실험처치 효과가 왜곡될 가능성이 있다[8]. 본 실험에서는 확인 및 검증 교육과 확인 및 검증 활동의 적용 수준의 차별화가 실험처치에 해당하며, 이러한 실험처치 전후에 대하여 오류 발생 가능성의 차이를 분석하는 것이 본 실험의 핵심 내용이다. 따라서 탈락에 의한 참여자 유실 가능성을 최소화하고 프로그래밍 실력의 차이, 집중력의 차이 등과 같은 가외변인이 실험처치 효과를 왜곡할 수 있는 가능성을 최소화하기 위하여 표본의 크기를 18명으로 하였다. 실험연구의 내적타당도를 위하여 표본 크기를 본 연구와 유사하게 한 사례는 사회과학 및 체육과학 분야에서 많이 발견할 수 있다[9,10,11].

두 개발 팀은 각각 서로 다른 소프트웨어를 개발하였다. 먼저, 확인 및 검증에 대한 교육 전에는 요구사항명세서만을 토대로 소프트웨어를 구현하고, 확인 및 검증 교육 후에는 소프트웨어 종류를 서로 바꾸어 요구사항 분석, 설계, 구현의 각 개발 공정에 대하여 확인 및 검증 기법을 적용하며 개발하는 방식으로 실험을 수행하였다. 팀원들 간에 의사소통은 철저히 제어되었다. 적용된 확인 및 검증 기법은 인스펙션[6], 블랙박스 시험 (Black-box Testing)에 해당하는 기능 시험, 화이트박스 시험 (White-box Testing)에 해당하는 Decision Coverage Testing 이었다[12].

처음 소속되었던 팀과는 관계없이, 모든 개발자들에 대하여 확인 및 검증 교육 결과에 대한 평가를 거쳐 우수한 그룹(A 등급)과 그렇지 않은 그룹(B 등급)으로 분류하였다. 이는 충실한 확인 및 검증 적용과 충실하지 않은 확인 및 검증 적용 사이의 차이를 분석하기 위한 조치였다.

앞서 실험에 참가한 학생들의 실력이 비슷한 수준이었다고 언급하였다. 그러나 이에 대한 검증이 필요하므로 본 실험에서는 개발 팀 간의 실력 차가 있었는가를 검증하기 위하여 확인 및 검증 성적에 대한 빈도차 검증을 카이-스퀘어 (X2) 분석을 통하여 수행하였다. 분석 결과 두 팀 모두에서 산출된 확인 및 검증 성적에는 차이가 없는 것으로 나타났다. 즉, 두 집단 모두에서 A등급 5명 (55.6%), B 등급 4명 (44.4%)으로 균등하게 나타났다 ($X2=.000$, $p<0.05$). 따라서 개발 팀 간의 소프트웨어 개발 실력에는 차이가 없는 것으로 판단할 수 있었다.

2.2 오류 분석을 위한 소프트웨어 시험

오류를 검출하기 위하여 일정한 수의 시험 시나리오가 미리 개발되었다. 이 시나리오들은 개발자들에게 알려지지 않도록 관리되었으며, 소프트웨어 시험 전문가에 의해 생성되었다. 오류 검출을 위한 시험 또한 동일한 전문가에 의해 수행되었다. 분석 대상이 되는 오류의 종류는 구현 오류와 구현 전 오류로 한정하였다. 구현 오류란 요구사항을 정확하고 완전하게 이해하였으나 알고리즘 구현이나 프로그래밍 상에서 발생한 오류를 의미하며, 구현 전 오류란 요구사항을 정확하고 완전하게 이해하지 못한데서 기인하는 오류를 의미한다. 구현 오류의 검출을 위한 시험 시나리오와 구현 전 오류 검출을 위한 시험 시나리오는 동일한 수로 개발되었다.

2.2.1 공정제어 모사 소프트웨어용 시험 시나리오

공정제어 모사 소프트웨어는 0에서 100까지의 정수 값을 갖는 A, B, C 세 개의 공정변수를 파일로부터 입력 받아 설정치 범위를 벗어나면 'TRIP'을 그렇지 않으면 'NORMAL'을 출력하는 소프트웨어이다. 입력 주기를 정확하게 지키는 것과 각 공정변수의 설정치 범위 이탈 여부에 대한 판단이 본 소프트웨어의 중요한 특성이다. 시험 시나리오는 다음과 같다.

1) 프로그램 시작 후 최초 10주기 동안 입력과 관계없이 'NORMAL' 출력 (시험입력: 프로그램 시작, 예상출력: NORMAL)
2) 모든 공정변수의 상태가 NORMAL에서 움직이다가 공정변수 C만 설정치를 감소할 때 설정치 밖에서 TRIP 되는지의 여부 (시험입력: C 설정치 45 C 변수 50, 예상출력: TRIP)
3) 모든 공정변수의 상태가 NORMAL에서 움직이다가 공정변수 B만 설정치를 벗어날 때 TRIP 되는지의 여부 (시험입력: 공정변수 B 20에서 40으로 증가, 예상출력: TRIP)
4) 출력을 요구사항이 원하는 형태대로 하는지의 여부 (시험입력: 공정변수가 임의로 변하는 파일, 예상출력: NORMAL과 TRIP)
5) 공정변수 유효범위 초과 시 TRIP 발생 여부 (시험입력: 공정변수 A 110, 예상출력: TRIP)
6) 공정변수 B에 대하여 설정치가 10/주기이나 -10/주기에 대해서도 TRIP으로 처리하지 않는가의 여부 (시험입력: 공정변수 B 80에서 50으로 감소, 예상출력: NORMAL)
7) 공정변수 C의 설정치 조정이 40 미만으로는 동작하지 않는지의 여부 (시험입력: 설정치 조정 버튼 9회 이상 조작 및 공정변수 C 40 이하 유지, 예상출력: NORMAL)
8) 공정변수 A와 C가 20미만에서 TRIP을 발생하는지의 여부 (시험입력: 공정변수 A와 C를 20 미만으로 한 파일, 예상출력: TRIP)
9) 입력을 요구사항이 원하는 형태대로 하는가의 여부 (시험입력: 요구사항에 맞지 않는 형태의 입력파일, 예상출력: TRIP)
10) 공정변수 데이터가 부족할 때(데이터가 하나 또는 두 개일 때)의 TRIP 발생 여부 (시험입력: 데이터가 하나인 입력파일, 예상출력: TRIP)

위와 같은 시험 시나리오는 공정제어 모사 소프트웨어에 대한 요구사항명세서를 기반으로 개발되었으며, 경계치 시험, 기능 시험, Decision Coverage 시험 등의 기법을 적용하여 개발하였다.

2.2.2 오델로 게임 소프트웨어용 시험 시나리오

오델로 게임 소프트웨어는 가로 8개, 세로 8개의 격자를 갖는 판위에 돌을 올려놓으면서 자신의 순서가 되면 상대방 돌에 인접한 곳에 자신의 돌을 놓아 자신의 돌로 상대방 돌을 둘러싸는 게임 소프트웨어이다. 상대방 돌을 둘러싸게 되면 자신의 돌 사이에 있는 상대방 볼이 자신의 돌 색깔로 바뀌게 하되 연쇄 반응은 없어야 하며, 돌을 놓을 수 있는 곳과 없는 곳을 판단하는 것이 본 소프트웨어의 중요한 특성이다. 시험 시나리오는 다음과 같다.

1) '98'을 입력할 때 (Boundary Conditions) 메시지 출력 여부 (시험입력: 98, 예상출력: 잘못 놓으셨습니다)
2) '0'을 입력하여 종료할 경우 점수를 표시하고 종료되는지의 여부 (시험입력: 0, 예상출력: 점수 표시 후 게임종료)
3) 숫자 0이 아니라 문자 'O'를 입력해도 종료되는지 여부 (시험입력: 'O', 예상출력: 잘 못 입력하셨습니다)
4) 입력되는 숫자가 두 개가 아닌 경우 메시지 출력 여부 (시험입력: 456, 예상출력: 잘 못 입력하셨습니다)
5) 게임이 어느 정도 진행되는 중에 놓일 수 없는 곳 점검하는 기능 (시험입력: 33에 돌을 놓고 33을 입력, 예상출력: 돌을 놓을 수 없습니다)
6) 놓을 곳에 놓으면 말이 바뀌게 하는 기능 (시험입력: 흰돌이 33, 34 위치에 있고 검은돌이 32 위치에 있는 경우 검은돌 순서에서 35를 입력, 예상출력: 흰돌이 검은돌로 바뀜)
7) 꽉 차지 않았는데 더 이상 놓을 곳이 없을 경우에 턴을 넘겨야만 하는데, 놓도록 하는

경우는 없는가의 여부 (시험입력: 검은돌이 더 이상 놓을 곳이 없는 경우, 예상출력: 흰돌 순서로 턴을 넘김)

8) 말의 가로 세로 좌표를 요구사항대로 정확히 놓을 수 있게 하는 기능 (시험입력: 34, 예상출력: 가로 3 세로 4 위치에 돌 출력)
9) 연쇄반응이 일어나지 않는가의 여부 (시험입력: 돌의 색깔을 바꾸도록 하는 입력, 예상출력: 바뀐 돌에 의한 연쇄반응이 일어나지 않음)
10) 꽉 채우면서 끝낼 때 마지막 말을 표시하는지의 여부 (시험입력: 판을 꽉 채우면서 끝내는 마지막 돌 입력, 예상출력: 마지막 돌 표시)

위와 같은 시험 시나리오는 오델로 소프트웨어에 대한 요구사항명세서를 기반으로 개발되었으며, 경계치 시험, 기능 시험, Decision Coverage 시험 등의 기법을 적용하여 개발하였다.

3. 실험 결과

본 실험의 주요 결과는 다음과 같다.

◉ 충실한 확인 및 검증의 적용은 오류 발생 수를 통계적으로 유의하게 감소시킴
◉ 구현 전 오류 수 감소와 구현 오류 수 감소는 통계적으로 차이가 없음

첫째로, 충실한 확인 및 검증의 적용이 오류 발생 수를 통계적으로 유의하게 감소시킨다는 것은 소프트웨어 개발 과정이 최종 결과물의 신뢰도 품질에 영향을 미친다는 것을 의미한다. 이는 또한 BBN 기반 방법 등을 활용하여 소프트웨어 개발 과정을 기반으로 소프트웨어 신뢰도를 평가하는 것이 매우 의미 있는 활동임을 보여준다. [표 1]에서 알 수 있는 바와 같이, 확인 및 검증 교육 평가에서 좋은 성적으로 A 등급을 받은 집단 (N=10)과 좋지 않은 성적으로 B 등급을 받은 집단 (N=8) 사이에서, 교육 전에는 오류 총 평균수, 구현 오류 평균수, 구현 전 오류 평균수에서는 통계적으로 차이가 없었으나(모두 유의확률 $p>0.05$), 교육 후에는 오류 총 평균수, 구현 오류 평균수, 구현 전 오류 평균수에서는 두 집단 간에 통계적으로 유의미한 차이가 있는 것으로 나타났다(모두 $p<0.05$). 이는 확인 및 검증 활동이 오류(구현 오류, 구현 전 오류)를 감소시키는데 효과적임을 보여주는 것이다. 참고로 [표 1]에서 M은 평균, SD는 표준편차, t는 표준점수, df는 자유도(Degree of Freedom)를 의미한다. 표준점수는 어떤 점수와 평균 간의 차이인 편차를 표준편차로 나누어서 변환시킨 점수를 의미한다.

실험 결과에 의하면, A 등급을 받은 집단은 교육 전에 비하여 교육 후 전체 오류에서 -3.40, 구현 오류 -1.80, 구현 전 오류 -1.50 로 변화하였으며, B 등급을 받은 집단은 전체 오류 +0.125, 구현 오류 +0.250, 구현 전 오류 -0.125로 변화하였다. 즉, 상대적으로 충실하게 확인 및 검증을 적용한 A 등급 집단은 구현 오류와 구현 전 오류 모두에서 오류수가 통계적으로 유의미하게 낮아졌으며, 그렇지 못한 B 등급 집단에서는 오류수가 낮아지지 않았다. 이는 충실한 확인 및 검증 활동이 오류(구현 오류, 구현 전 오류)를 감소시키는데 효과적이지만, 그렇지 못한 확인 및 검증 노력은 오류 감소에 효과가 없음을 보여주는 것이다. 따라서 확인 및 검증 팀을 잘 조직하고, 확인 및 검증 공정 활동을 잘 적용하였는가를 평가하는 것이 소프트웨어 신뢰도 평가에 필수적으로 포함되어야 한다는 것을 알 수 있다.

[표 1] 확인 및 검증 성적 별 교육 전후의 오류 수 비교

영역 1)	집단	N	M	SD	t	df	유의확률
오류 1	A	10	5.00	1.491	0.239	12.763	0.815
	B	8	4.88	0.641			
구현 1	A	10	2.20	1.687	0.115	16	0.910
	B	8	2.13	0.835			
구현전 1	A	10	2.80	1.398	0.088	16	0.931
	B	8	2.75	0.886			
오류 2	A	10	1.60	0.843	6.684	16	0.000
	B	8	5.00	1.309			
구현 2	A	10	0.40	0.516	3.197	8.056	0.013
	B	8	2.38	1.685			
구현전 2	A	10	1.30	0.675	2.446	9.559	0.035
	B	8	2.63	1.408			

주 1) 오류 1 등의 1은 확인 및 검증 교육 전, 2는 교육 후를 의미

둘째로, 구현 전 오류 수 감소와 구현 오류 수 감소는 통계적으로 차이가 없다는 실험 결과는 소프트웨어 개발 과정의 모든 단계에서 발생할 수 있는 오류에 대하여 확인 및 검증의 효과가 단계별로 동일하다는 것을 의미한다. 이는 또한 소프트웨어 개발 전 과정에 대하여 단계별로 차등화하지 않고 확인 및 검증을 철저하게 수행할 때 소프트웨어 신뢰도 품질의 향상을 극대화 할 수 있다는 것을 의미한다. 이 실험 결과를 통해서, BBN 기반 방법 등을 활용하여 소프트웨어 개발 과정을 기반으로 소프트웨어 신뢰도를 평가할 때, 각 단계별 확인 및 검증 노력이 소프트웨어 오류 발생 가능성에 미치는 영향이 동일한 비중으로 평가되어야 함을 알 수 있다.

[표 2]를 통해 확인 및 검증 교육에 대한 평가에서 A 등급을 받은 집단의 경우, 확인 및 검증 적용 전과 비교할 때 총 오류, 구현 오류, 구현 전 오류 발생 수에서 모두 감소하였다는 것이 통계적으로 유의함을 확인할 수 있다. 하지만 구현 오류에 대한 변화와 구현 전 오류에 대한 변화의 차이가 통계적으로 유의하지 않다. 즉, 확인 및 검증 노력이 구현 오류의 감소에 미치는 영향과 구현 전 오류의 감소에 미치는 영향은 동일하다.

[표 2] 오류 종류 별 교육 전후의 오류 수 비교 (A 등급 집단)

집단	N	M	SD	t	dt	유의확률
오류 1	10	5.00	1.491	11.129	9	0.000
오류 2	10	1.60	0.843			
구현 1	10	2.20	1.687	3.515	9	0.007
구현 2	10	0.40	0.516			
구현전 1	10	2.80	1.398	3.737	9	0.005
구현전 2	10	1.30	0.675			
구현변화	10	1.80	1.619	0.350	9	0.734
구현전변화	10	1.50	1.269			

또한, [표 3]에서 나타나는 결과에 의해 확인 및 검증 교육에 대한 평가에서 B 등급을 받은 집단의 경우, 확인 및 검증 적용 전과 비교할 때 총 오류, 구현 오류, 구현 전 오류 발생 수에서 모두 감소하였다는 것이 통계적으로 유의하지 않으며, 구현 오류에 대한 변화와 구현 전 오류에 대한 변화의 차이에서도 통계적으로 유의하지 않았다.

[표 3] 오류 종류 별 교육 전후의 오류 수 비교 (B 등급 집단)

집단	N	M	SD	t	dt	유의확률
오류 1	8	4.88	0.641	0.284	7	0.785
오류 2	8	5.00	1.309			
구현 1	8	2.13	0.835	0.357	7	0.732
구현 2	8	2.38	1.685			
구현전 1	8	2.75	0.886	0.215	7	0.836
구현전 2	8	2.63	1.408			
구현변화	8	0.125	1.642	0.310	7	0.765
구현전변화	8	0.250	1.982			

4. 결 론

본 연구는 BBN 기반 방법과 같이 정성적 판단이 개입될 수밖에 없는 신뢰도 평가 방법에 대해 실험적이고 정량적인 근거 자료를 제공한다. 본 실험은 좋은 소프트웨어 개발 과정이 좋은 신뢰도의 소프트웨어를 생산한다는 것을 밝혔다. 이는 소프트웨어 신뢰도를 평가할 때 개발 공정이 얼마나 잘 조직되고 공정 활동이 얼마나 잘 적용되었는가를 평가하는 것이 필수적이라는 사실을 보여준다. 소프트웨어의 개발 과정이 최종 결과물의 신뢰도에 영향을 미칠 것이라는 가설은 전문가들 사이에서 이견이 없는 상식적 수준의 사실이지만, 소프트웨어 개발 과정의 품질이 소프트웨어의 신뢰도에 미치는 영향을 정량적으로 평가하고 이를 통하여 소프트웨어 신뢰도 평가에 대한 지침을 제공하였다는 점에서 본 실험은 큰 의미를 갖는다고 하겠다.

또한, 본 실험은 확인 및 검증 노력이 소프트웨어 개발 과정의 각 단계 별로 오류 발생 가능성에 어떤 방식으로 영향을 미치는가를 규명하였다는 점에서 큰 의미가 있다. 본 실험의 결과에 따르면, 확인 및 검증 노력이 구현 오류의 감소에 미치는 영향과 구현 전 오류의 감소에 미치는 영향은 동일하다. 이는 소프트웨어 개발 과정의 각 단계 별 확인 및 검증 노력이 해당 단계의 오류 발생 가능성에 동일한 비중으로 영향을 미친다는 것을 의미한다. 따라서 BBN 기반 방법 등을 활용하여 소프트웨어 개발 과정을 기반으로 소프트웨어 신뢰도를 평가할 때, 각 단계별 확인 및 검증 노력이 소프트웨어 오류 발생 가능성에 미치는 영향이 동일한 비중으로 평가되어야 할 필요가 있다.

게임, IPTV, DMB 등의 멀티미디어 서비스를 시장에 출시할 때, 소프트웨어 신뢰도를 시험 결과 기반으로만 평가하기에는 한계가 있다. 게임과 같은 멀티미디어 서비스의 특성 상 막대한 시험 비용이 발생하며 시험에 필요한 시간이 상당히 소요되기 때문이다. 이를 보완하기 위하여 소프트웨어 개발 공정을 기반으로 소프트웨어의 신뢰도를 평가한다면 적절한 시점에 양질의 멀티미디어 서비스를 위한 소프트웨어를 시장에 출시할 수 있을 것이다. 특히, 본 연구의 결과를 반영하면서 소프트웨어 개발 공정 기반의 소프트웨어 신뢰도 평가를 수행하는 것이 평가의 유용성 및 확신도를 제고하는데 상당한 도움이 될 것으로 사료된다.

향후 연구로서, 본 연구에서 수행한 실험의 결과를 반영하여 소프트웨어 개발 공정을 기반으로 하는 소프트웨어 신뢰도 평가 기법을 개발하고자 한다. 이러한 연구는 기존의 소프트웨어 개발 공정 기반 소프트웨어 신뢰도 평가 방법의 성능을 개선하는데 기여하게 될 것이다.

참고문헌

[1] 정혜정, “게임 소프트웨어의 품질 평가 모델,” 인터넷정보학회논문지, 제8권, 제6호, pp. 115-125, 2007.

[2] 한준탁, 명원식, “교육용 온라인 게임 소프트웨어의 품질평가 개선,” 한국콘텐츠학회논문지, 제6권, 제5호, pp. 104-112, 2006.

[3] G. Dahll, “Combining disparate sources of information in the safety assessment of software-based systems”, Nuclear Engineering and Design, Vol. 195, No. 3, pp. 307-319, 2000.

[4] B. Cukic et al., “A bayesian approach to reliability prediction and assessment of component based systems,” Proc. of ISSRE, November 2001.

[5] H. S. Eom et al., “Reliability assessment of a safety-critical software by using generalized Bayesian nets,” Proc. of NPIC&HMIT, April 2009.

[6] Y. S. Lim et al., “The Use of Fagan Inspection in Software Fault Detection and Comparisons with Other Methods : An Experiment Study,” Journal of the Korea Information Science Society(C), Vol. 2, No. 1, pp. 1-11, 1996.

[7] S. S. So et al., “An empirical evaluation of six methods to detect faults in software,” Software Testing, Verification and Reliability,

Vol. 12, No. 3, pp. 155-171, 2002.
[8] 김석우, 최태진, 교육연구방법론, pp. 109-110, 322, 학지사, 2007.
[9] 이지영, 손정락, “역기능적 분노의 조절을 위한 Siegel의 대인신경생물학적 모형에 기반한 절충적 프로그램의 효과검증,” 한국심리학회지: 건강, 제16권, 제2호, pp. 243-261, 2011.
[10] 도기봉, 오주, 신정인, “학교폭력 피해 여고생의 자아존중감과 역량강화를 위한 임파워먼트 프로그램의 효과,” 청소년학연구, 제18권, 제1호, pp. 149-174, 2011.
[11] 김광준, 김효중, 정진욱, “12주간의 복합 파워트레이닝이 여자 프로골퍼의 근력, 파워 및 드라이버 수행력에 미치는 영향,” 체육과학연구, 제22권, 제1호, pp. 1635-1644, 2011.
[12] McLeod, R., “Software Testing”, JohnWiley & Sons Inc, 2007.

손 한 성 (Son, Han Seong)

1993 서울대 원자핵공학과 (공학사)
1995 KAIST 원자력공학과 (공학석사)
2000 KAIST 원자력공학과 (공학박사)
2002-2004 한국원자력연구원 선임연구원
2008-현재 중부대학교 게임학과 전임강사

관심분야 : 소프트웨어 신뢰도, 게임 소프트웨어공학, 소프트웨어 분석 및 설계

노 창 현 (Roh, Chang Hyun)

1993 KAIST 원자력공학과 (공학사)
1995 KAIST 원자력공학과 (공학석사)
2001 KAIST 원자력공학과 (공학박사)
2006-2007 엠게임 정보기획실장
2007 엔트리소프트 미국지사 자문위원
2002-현재 중부대학교 게임학과 조교수

관심분야 : 게임 기획, VR, Interactive Media