한국시뮬레이션학회 논문지
Vol. 20, No. 3, pp. 19-28 (2011. 9)

통합 시스템을 위한 출력 분포 기반 적응적 랜덤 테스팅

신승훈[1] · 박승규[2†] · 최경희[2] · 정기현[3]

Adaptive Random Testing for Integrated System based on Output Distribution Estimation

Seung-Hun Shin · Seung-Kyu Park · Kyung-Hee Choi · Ki-Hyun Jung

ABSTRACT

Adaptive Random Testing (ART) aims to enhance the performance of pure random testing by detecting failure region in a software. The ART algorithm generates effective test cases which requires less number of test cases than that of pure random testing. However, all ART algorithms currently proposed are designed for the tests of monolithic system or unit level. In case of integrated system tests, ART approaches do not achieve same performances as those of ARTs applied to the unit or monolithic system. In this paper, we propose an extended ART algorithm which can be applied to the integrated system testing environment without degradation of performance. The proposed approach investigates an input distribution of the unit under a test with limited number of seed input data and generates information to be used to resizing input domain partitions. The simulation results show that our approach in an integration environment could achieve similar level of performance as an ART is applied to a unit testing. Results also show resilient effectiveness for various failure rates.

Key words : Random Testing, Adaptive Random Testing, Integration Test, Test Case Generation, Simulation

요 약

적응적 랜덤 테스팅(ART)은 순수 랜덤 테스팅의 효율성을 개선하기 위해 제안된 방법으로 효과적인 테스트 케이스의 선택을 통해 보다 적은 수의 테스트 케이스로 소프트웨어 내에 존재하는 오류 영역을 찾는 것을 목적으로 한다. 기존의 ART는 하나의 시스템 혹은 유닛에 대한 테스트를 적용 대상으로 하고 있으며, 다양한 접근 방법을 이용해 순수 랜덤 테스팅보다 우수한 성능을 보여 왔다. 하지만 통합 시스템을 구성하는 특정 유닛에 대해 ART를 적용하고자 하는 경우에는 시스템을 구성하는 타 유닛들의 영향으로 인해 기대 이하의 효율성을 보이게 된다. 따라서 본 논문에서는 이와 같은 테스트 환경 대한 ART 적용 방법의 하나로, 테스트 대상 유닛에 부여되는 입력 데이터의 분포를 통합 시스템에 대한 제한된 수의 입력을 사용하여 예측하고, 이를 바탕으로 시스템의 입력 도메인 분할 크기를 조절하는 방법을 제안하고 시뮬레이션을 통해 성능을 평가한다. 시뮬레이션 결과, 제안된 방법은 유닛 테스팅에 ART를 적용했을 때와 유사한 수준의 성능으로 통합 시스템 내의 특정 유닛을 테스트 가능하도록 하며, 오류 비율의 변화가 ART의 성능에 미치는 영향 또한 1% 이내 수준으로 안정임을 확인하였다.

주요어 : 랜덤 테스팅, 적응적 랜덤 테스팅, 통합 테스트, 테스트 케이스 선택, 시뮬레이션

1. 서 론

랜덤 테스팅은 임의로 생성한 입력 데이터를 이용하여 소프트웨어를 테스트 하는 가장 기초적이고 자동화가 용이한 테스팅 방법이다. 하지만 테스트 데이터의 선택 과정에서 테스트되는 시스템을 고려하는 수준이 낮기 때문에 체계적(systematic) 테스트에 대한 상대적인 개념으로

접수일(2011년 3월 10일), 심사일(1차 : 2011년 7월 27일), 게재 확정일(2011년 7월 27일)
[1] 아주대학교 정보컴퓨터공학부 강의교수
[2] 아주대학교 정보컴퓨터공학부 교수
[3] 아주대학교 전자공학부 교수
주 저 자 : 신승훈
교신저자 : 박승규
E-mail; sparky@ajou.ac.kr

이해되고 있으며, 이로 인해 랜덤 테스팅의 효율성에 대한 회의적 시각 또한 존재한다(Hamlet, 1994). 그럼에도 불구하고 랜덤 테스팅은 테스팅 방식이 가지는 고유한 특성인 임의성을 이용하여 시스템의 신뢰성 평가에 유용하게 사용될 수 있으며(Chen 등, 2006), 취약성 검사를 목적으로 하는 퍼즈(fuzz) 테스팅에도 효과적으로 적용되어 왔다(Ganesh 등, 2009; Faruk, 2008; Sutton 등, 2007).

적응적 랜덤 테스팅(Adaptive Random Testing, ART)은 랜덤 테스팅의 단점인 낮은 효율성을 보완하기 위해 Chen 등이 제안한 방식(2004)으로, 소프트웨어의 오류(failure)를 유발하는 입력 데이터의 분포 형태(Finelli, 1991)와 오류 유발 입력 데이터의 분포 형태에 따른 테스팅 정책의 효율성 변화(Chan 등, 2002)를 고려하여 테스트 케이스를 선택하도록 하는 정책이다. ART는 오류를 유발하는 입력이 입력 도메인 내에 몇 가지 유형을 가지고 인접해 존재하는 경향이 있다(Finelli, 1991)는 사실을 바탕으로 테스트 케이스를 선택한다. 좀 더 명확히 기술하자면, ART는 새로운 테스트 케이스 선택 시 기존 테스트 케이스들과 충분한 거리를 갖도록 하되, 가능한 입력 도메인 내에 고르게 분포되도록 유도하는 방법을 사용한다. 이러한 목적 달성을 위한 다양한 접근 방법이 제안되었는데, 이들은 크게 테스트 케이스 사이의 유클리드 거리(Euclidean distance)를 이용하는 거리 기반 방법(Chan 등, 2002; Chen 등, 2004; Mayer 등, 2006)과 입력 도메인을 일정한 방식으로 분할하고, 분할된 파티션에서 임의의 데이터를 선택하는 방법(Chen 등 2006; Mayer, 2006; 신승훈 등, 2008, 2009)으로 나뉜다. 각 접근 방법에는 이전의 ART가 보였던 단점을 보완하기 위한 다양한 변형이 존재하며, 이들 모두는 테스팅 속도와 오류 검출의 효율성 모두를 확보하기 위한 방향으로 발전되어왔다.

하지만 ART는 테스트 대상이 되는 소프트웨어를 하나의 독립된 시스템 혹은 유닛으로 가정하고 있기 때문에 제한적인 테스팅 적용 범위를 가진다. 일례로 통합 테스팅에 ART를 적용하는 경우, 시스템을 구성하는 유닛들 사이에 존재하는 의존 관계로 인해 전체 유닛에 고른 분포를 가지는 테스트 케이스를 부여하는 것은 거의 불가능에 가깝다. 따라서 일부 테스트 대상 유닛에서는 ART에 의해 부여된 테스트 케이스가 전혀 영향을 미치지 못하는 블라인드 영역이 생길 수 있고, 이 블라인드 영역 내에 오류가 위치하는 경우에는 ART로 추적이 불가능해진다. Shin 등(2010)은 이와 같은 현상이 발생하는 경우를 보이고, 전체 입력 통계를 바탕으로 문제의 해결을 시도하였다. 하지만 테스트 이전 단계에서 정확한 입력 통계를 구하는 작업은 높은 작업량 증가를 불러오게 되므로 실제 테스트에 적용하기에는 무리가 있다. 따라서 본 논문에서는 ART를 통합 혹은 회귀 테스트에 적용 가능하도록 하는 접근 방법 획득을 위한 초기 연구의 일환으로 우선 이미 배포된 하나의 기존 소프트웨어와 새로 개발된 하나의 소프트웨어의 통합이 이루어지는 환경에서의 문제 해결을 시도한다. 즉, 2종의 소프트웨어가 통합된 환경에서 입력을 수용하는 유닛에 의한 테스트 케이스 분포 왜곡이 일어나는 경우에 대한 ART 개선 방안을 제시한다. 제시된 방안은 제한된 수의 샘플 입력 데이터를 사용하여 UUT (Unit under Test)에 부여되는 입력 데이터의 분포를 예측하는 방식을 사용한다. 한편 테스트 케이스 분포 왜곡을 발생시키는 기존 소프트웨어로 Chen 등(2004)이 오류 탐지에 사용되었던 소프트웨어 가운데 본 논문에서 가정하는 환경에 적합한 것을 선택하며, 이 때 오류가 없는 소프트웨어의 원형을 이용한다. 그리고 개발된 소프트웨어는 기존 ART 관련 연구(Mayer, 2006, 신승훈 등, 2008, 2009)에서와 동일하게 입력 도메인 내의 임의의 위치에 오류가 존재하는 가상의 환경을 작성하여 실험을 수행한다.

2. 관련 연구 및 정의

2.1 관련연구

퍼즈 테스팅은 기본적으로 테스트 되는 시스템을 블랙박스로 보고 해당 시스템에 임의의 데이터를 삽입한 후, 그 결과로 보이는 시스템의 예상치 못한 행동 양식의 확인을 통해 취약성의 존재 여부 및 소프트웨어의 오류를 확인하는 보안 테스팅 방법이다(Miller 등, 1990). 이와 같은 퍼즈 테스팅은 초기의 블랙박스 랜덤 퍼징으로부터 시작하여, 랜덤 방식의 효율성 개선을 위한 다수의 지능적인 기법을 사용하는 지능적 퍼징(smart fuzzing) 및 소프트웨어 코드를 직접 퍼즈 대상으로 하는 화이트박스 퍼즈 등으로 발전해 다양한 영역에 적용되었는데, 이들 가운데 MIT의 가네쉬(Ganesh) 등이 제안한 BuzzFuzz(Ganesh 등, 2009)는 화이트박스 퍼징 도구의 하나로 테스트되는 프로그램의 핵심 컴포넌트 내부의 코드 테스트를 위한 체계적인 테스트 케이스 생성을 목적으로 작성되었다. BuzzFuzz는 우선 테스트 되는 프로그램에 데이터 변형 추적을 위한 코드를 삽입한 후, 프로그램에 샘플 입력을 부여하여 해당 입력이 프로그램의 특정 위치에 어떤 식으로 영향을 미치는지를 확인하는 동적 감염(taint) 추적을 수행하고 이를 분석한다. 이러한 과정을 통해 BuzzFuzz는 공격 대상이 될 수 있는 프로그램 내의 특정 위치에 영향을 줄 수

있는 입력 데이터를 자동 생성하고, 이를 이용하여 프로그램을 테스트하는 방식을 사용한다. 화이트 박스 기반 테스트에서는 테스터가 소프트웨어의 소스 코드를 직접 조작하여, BuzzFuzz에서와 같이 소프트웨어의 내부에 임의의 추적 도구를 삽입하는 등의 작업을 수행할 수 있으나, 블랙박스 기반 테스트에서는 이와 같은 작업이 불가능하다. 따라서 블랙박스 테스팅의 하나인 ART를 이용하여 프로그램 내의 특정 위치 혹은 유닛에 대한 테스팅을 수행하고자 하는 경우, SUT(System under Test)에 주어지는 입출력 데이터의 정보만을 확인할 수 있으므로 이 데이터의 제어를 통한 접근 방법이 요구된다.

Shin 등(2010)은 ART를 통합 테스트에 사용하는 경우, SUT에 주어진 균일한 분포의 테스트 케이스가 시스템의 행동 양식에 의해 변화되어 UUT에는 왜곡된 분포가 주어질 수 있고, 이에 따라 ART 효율성에 변화가 발생함을 지적하였다. 또한 이러한 현상의 보완을 위해 UUT에 대한 입력 통계를 구해 이를 기반으로 입력 도메인을 분할하는 방법을 제안하였다. 그러나 UUT에 부여되는 정확한 입력 통계를 구하는 작업은 테스트에 수반되는 작업량을 다량 증가시키게 되므로 실제 테스트에 적용하는 데에는 무리가 있다.

2.2 정의

유한 크기를 가지는 하나의 소프트웨어에 대한 입력 데이터 집합을 입력 도메인(D)이라 하고, D를 구성하는 전체 입력 데이터 가운데 소프트웨어의 오류를 유발하는 입력 데이터의 비율을 오류 비율(failure rate)이라 정의한다. 그리고 소프트웨어의 오류를 유발하는 입력 데이터들은 입력 도메인 내에서 군집을 이루어 오류 영역(failure region)을 형성하며, 이 오류 영역은 몇 가지 유형을 가지는데 이를 오류 패턴(failure pattern)이라 한다. 오류 패턴은 크게 블록(Block), 스트립(Strip) 및 포인트(Point) 패턴으로 분류되고, 이들 가운데 블록 패턴과 스트립 패턴이 상대적으로 빈번하게 발생하는 오류 패턴이다(Chan 등, 1996).

F-measure는 소프트웨어 테스트 수행 과정에서 첫 번째 오류를 발견할 때까지 소요된 테스트 케이스의 수를 의미한다(Chen 등, 2004). 이 F-measure는 타 척도에 비해 직관적인 효율성 평가 지수로 인식이 되고 있으며, 이에 근거는 다음과 같이 정리될 수 있다(Finelli, 1991; Chen 등, 2004).

(1) 소프트웨어 내에 존재하는 오류의 비율은 발견된 오류의 수에 대해 선형 로그(log-linear) 형태를 따른다.
(2) 하나의 오류는 다른 오류의 존재를 노출시키거나 감추는 경향을 가진다.
(3) 소프트웨어의 테스트 과정에서 오류 발견 시, 이에 대한 원인 파악 및 조치가 우선 수행된다.

F-measure는 입력 도메인 내의 오류 비율에 영향을 받는 지수이기 때문에 이를 직접 사용하지 않고, 순수 랜덤 테스트의 효율성을 기준으로 하는 상대 F-measure를 이용한다. 하나의 소프트웨어의 입력 도메인 내에 존재하는 오류 비율이 θ일 때, 이 소프트웨어에 대한 순수 랜덤 테스팅의 이론적 F-measure는 $F_{RT} = 1/\theta$이 된다. 그리고 제안된 테스트 케이스 생성 방법의 효율성($F_{X\text{-}ART}$)은 F_{RT}에 대한 상대 수치인 평균 상대 F-measure로 평가한다. 즉, 평균 상대 F-measure는 하나의 ART 정책을 사용했을 때 얻어지는 F-measure 평균값과 순수 랜덤 테스트를 수행했을 때의 F-measure 비율이며, 단일 시스템 혹은 유닛에 대해 ART를 적용한 경우의 평균 상대 F-measure는 다음과 같은 방법으로 얻어진다.

$$\text{평균 상대 F-measure} = (\text{평균 } F_{X\text{-}ART}) \;/\; F_{RT}$$

즉, 평균 상대 F-measure가 1인 경우, 사용된 방법이 순수 랜덤 테스트와 동일한 효율성을 가짐을 의미하고, 평균 상대 F-measure가 1보다 클수록 낮은 효율성을, 0에 가까울수록 높은 효율성을 가지고 있음을 의미한다.

3. 분포 예측을 통한 ART 기반 통합 테스트

3.1 시스템 구성

본 논문에서 제안하는 테스트 케이스 생성 방법은 그림 1과 같은 형태로 통합되는 시스템의 테스트를 적용 대상으로 하며, 테스트의 목적은 시스템에 통합되는 유닛 내에 존재하는 오류 영역의 검출이다. 통합은 이미 배포되어 사용되고 있는 Unit A와 개발 과정에 있는 Unit B를 대상으로 이루어지며, 시스템에 대한 입력은 Unit A에 수용되며, Unit A의 출력이 Unit B의 입력으로 주어진다. 이와 같은 형태의 통합은 소프트웨어 개발 과정에서 빈번하게 찾아볼 수 있는 형태로, 3자 컴포넌트를 사용한 개발 혹은 기존 웹 서비스 기반 소프트웨어 개발 과정 등에서 쉽게 발견할 수 있다.

이와 같은 시스템을 대상으로 기존의 ART 방법을 사용하여 테스트 케이스를 생성해 사용하는 경우, Unit A에는 ART에 의해 생성된 균일한 분포를 가지는 테스트 케

이스가 주어진다. 하지만 Unit A의 행동 양식에 따라 Unit A로부터 출력되는 데이터는 균일하지 못한 분포를 이루게 될 확률이 높으므로 Unit B에 균일한 입력을 부여하기 어렵다. 더욱이 분포의 왜곡이 심한 경우에는 Unit B에 대한 입력 도메인의 일부 구간으로 전혀 입력이 발생되지 못 할 가능성도 존재한다. 이는 ART에 의한 Unit B 내 오류 영역 검출 효율성이 Unit A의 행동 양식에 의존적이 된다는 사실을 의미하기 때문에, 테스트 케이스 생성 방법이 일정 수준의 효율성을 보장하도록 하기 위해서는 이와 같은 의존성을 배재할 수 있는 방법이 필요하다.

하지만 ART는 블랙박스 테스트를 가정하고 있어 Unit A 내부에 도구를 추가하는 등의 작업 수행이 불가능하기 때문에 Unit A 행동 양식을 추정할 정보가 입출력 데이터

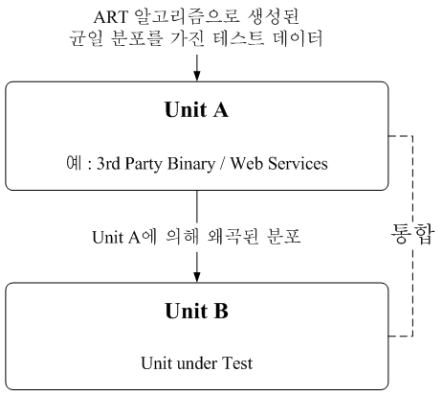

그림 1. 테스트 대상 시스템(Structure of System to Test)

정보 등으로 제한될 수밖에 없다. 따라서 본 논문에서는 Unit A의 출력 분포를 Unit B에 존재하는 오류 추적에 사용할 수 있는 가장 직관적이고, 접근하기 용이한 자료로 간주하고, 이를 이용해 ART의 입력 도메인 분할 크기를 조절하는 출력 분포 예측 기반 ART(DES-ART, Distribution EStimation-based ART)를 제안한다. 한편 DES-ART의 적용 대상은 통합 테스트로 기존 ART의 대상과 다르기 때문에 기존의 F-measure를 DES-ART의 효율성 평가에 그대로 적용하기 어려우므로 다음과 같이 수정 정의하여 사용한다.

$$\text{평균 상대 } F\text{-measure} = (\text{평균 } F_{X\text{-}ART\_INT}) \,/\, F_{RT\_UnitB}$$

즉, 효율성의 평가 지수는 Unit B 내에 존재하는 오류 영역을 찾기 위해 Unit B에 순수 랜덤 테스트를 사용했을 때 필요한 이론적 테스트 케이스 수(F_{RT_UnitB})와 통합 시스템에 ART를 사용한 경우에 필요한 테스트 케이스 수($F_{X\text{-}ART\_INT}$) 평균의 비율을 의미한다. 이와 같은 F-measure 정의의 수정은 시스템 내의 오류 영역이 Unit B에 한정되어 있고, 또한 테스트 되는 시스템이 통합된 시스템으로 테스트 케이스가 개별 유닛에 부여되는 것이 아닌 시스템에 부여되기 때문에 요구되는 사항이다.

3.2 가정 출력 모델

본 논문에서 대상으로 하는 시스템의 Unit A는 배포되어 사용 중인 소프트웨어를 가정하고 있다. 따라서 UnitA는 이미 테스트가 완료되어 내부에 오류가 존재하지 않고,

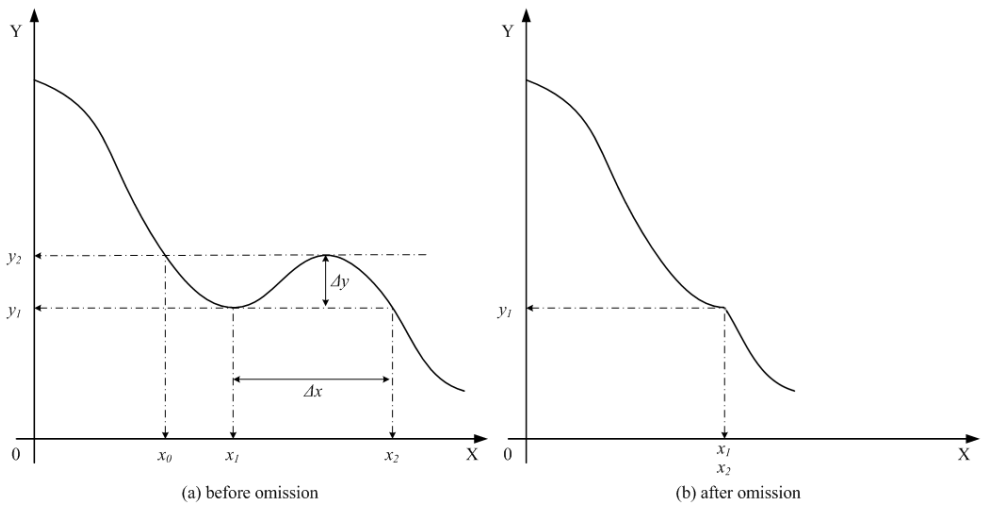

그림 2. 중복 입력 도메인 생략(Omission of Overlapped Part from Input Domain)

시스템 내의 오류 영역은 Unit B에만 존재하는 것으로 가정한다. 그리고 Unit A는 1차원 입력을 수용하여 1차원 출력을 생성하며, 입력 데이터 집합에서 출력 데이터 집합 사이에는 전사(surjection) 사상이 성립하는 것으로 가정한다.

또한 DES-ART에서는 Unit A의 출력이 단조 증가 혹은 단조 감소 함수 형태를 가지는 것으로 본다. 이에 대한 근거는 그림2의 예제를 통해 기술한다. 그림 2의 X축은 시스템의 입력 도메인을, Y축은 Unit A의 출력을 나타내며, 그림 2(a)는 입력 데이터가 주어졌을 때의 Unit A의 실제 출력 값 자취를 보이고 있다. 본 논문에서 대상으로 하는 시스템에서 Unit A의 출력은 Unit B의 입력 데이터로 주어지기 때문에, 그림 2(a)와 같은 형태의 출력에서는 $[x_0, x_2]$ 구간의 입력이 시스템에 주어지는 경우 $[y_1, y_2]$ 사이의 값은 반복되어 나타난다. 즉, 별도의 고려가 없는 경우, Unit B에 주어지는 테스트 케이스가 일부 Δy 구간에서 불필요하게 중복되는 현상이 발생된다. 이처럼 중복 생성된 테스트 케이스는 테스트에 불필요한 작업량만을 가중시키므로, 중복 출력을 유발하는 X축 상의 영역 $[x_1, x_2)$에서는 테스트 케이스 생성을 생략할 필요가 있다. 이와 같이 해당 구간을 생략하는 경우 Unit A의 입력에서 출력은 전단사(bijective) 사상이 가능하게 되어, 출력은 그림 2(b)와 같은 단조 감소 혹은 증가 그래프 형태로 표현된다. 이러한 입력 구간 생략은 입력 데이터 집합에서 출력 데이터 집합이 전사 사상되는 모든 경우에 적용 가능하며, 이 때 중복 유발 구간을 생략하게 되면 두 집합은 전단사 사상이 가능한 단조 증가 혹은 단조 감소 형태를 띠게 된다. 따라서 본 논문에서는 Unit A의 출력이 구간 생략이 완료되어 단조 증가 혹은 감소하는 형태를 취한다고 가정한다.

3.3 출력 분포 예측

그림 3은 DES-ART를 사용하는 경우의 테스트 흐름을 보이고 있다. 테스트는 크게 두 단계로 구성되는데, 첫 번째 단계에서는 Unit A의 출력 분포 예측을 수행하며, 두 번째 단계에서는 예측된 정보를 바탕으로 실제 테스트를 수행한다. 첫 번째 단계에서 이루어지는 출력 분포 예측은 시스템의 입력 도메인으로부터 동일한 간격을 갖도록 입력 데이터를 추출하여 샘플 입력 데이터를 구성하고, 이를 Unit A에 부여하여 얻어지는 출력 데이터를 기초로 '분포 예측 도구'가 수행하게 된다. 이를 통해 얻어지는 출력 분포 보고서(ODR)는 이후 단계에서 ART가 입력

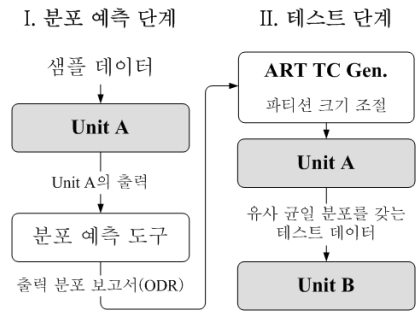

그림 3. 출력 통계 예측 방법(Process of Output Distribution Estimation)

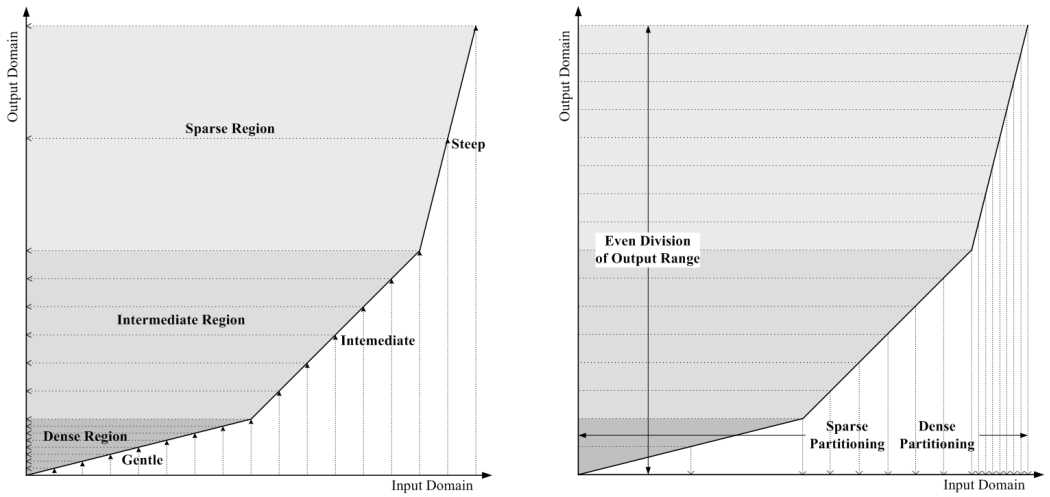

그림 4. 파티션 크기 조절 방법의 기본 개념(Fundamental Idea for Partition Size Adjustment)

도메인을 분할하는 근거 정보로 사용된다. ART는 첫 번째 단계에서 획득된 출력 예측 보고서를 기반으로 입력 도메인을 일정한 크기가 아닌 Unit A의 출력을 가능한 균일하게 이끌어 낼 수 있는 크기로 분할한다. 그리고 이를 Unit A에 부여하여 Unit A로 하여금 균일 분포에 가까운 출력을 발생시키도록 유도함으로써 Unit B에 균일한 테스트 케이스가 부여될 수 있도록 한다.

그림 4는 본 논문에서 제안하는 출력 값 기반 파티션 크기 조절 방법의 기본적인 개념을 묘사하고 있다. 구간에 따라 기울기(⊿output data / ⊿input data)가 변화하는 형태의 입출력 관계를 보이는 함수에서는 균일 간격으로 입력 데이터를 부여하고 각 입력에 대한 출력을 확인했을 때, 상대적으로 낮은 기울기를 갖는 구간에 부여된 입력 데이터일수록 높은 밀도를 가진 출력 구간을 유도한다는 사실에 바탕을 두고 있다. 즉, 이와 같은 환경을 가진 테스트에서 Unit A의 출력이 고른 분포를 갖도록 유도하기 위해서는 테스트 케이스의 밀도 조절이 고려되어야 함을 의미한다. 하지만 기존의 ART에서 사용하는 입력 도메인 분할 방법은 전체 입력 도메인에서 테스트 케이스가 균일한 분포로 선택되도록 하므로, 이에 대한 수정이 필요하다. 따라서 본 논문에서는 Unit A의 출력들이 가능한 서로 균일한 간격을 갖도록 하기 위해 입력 도메인에서 추출한 샘플 데이터를 이용하여 Unit A의 출력 밀도를 예측하고, 예측된 출력 밀도를 바탕으로 테스트 케이스 선택이 이루어지도록 파티션의 크기를 조절하는 방법을 제안한다.

3.4 출력 값 기반 파티션 크기 조절 방법

테스트 케이스 선택을 위한 첫 번째 단계에서는 입력 도메인(***D***)로부터 균일한 간격을 갖도록 추출된 샘플 데이터(***S***)를 이용하여 Unit A의 출력 분포 예측 및 이를 바탕으로 하는 입력 도메인 분할을 그림 5와 같은 형태로 수행한다. 이와 같은 작업을 위해 우선 추출된 ***S***를 Unit A에 입력으로 부여하고, 입력 ***S***에 대한 출력 ***O***를 수집한다. 수집된 ***O***를 기반으로 Unit A의 출력 범위(***OUA***)를 결정하는데, 본 논문에서는 Unit A의 출력 부분 가운데 중첩된 영역은 이미 소거된 상태를 가정하므로 ***O***를 구성하는 출력 데이터는 단조 증가 혹은 단조 감소 형태를 가진다. 일단 ***OUA***가 결정되면, 이를 사용된 입력 데이터 수와 동일한 수의 경계 값을 갖도록 하기 위해 |***S***|-1 분할하고, 각 분할 경계 값을 연산한다. 경계 값은 ***O***를 바탕으로 작성된 가상 그래프를 이용해 예측하는 방법을 사용하며, 이

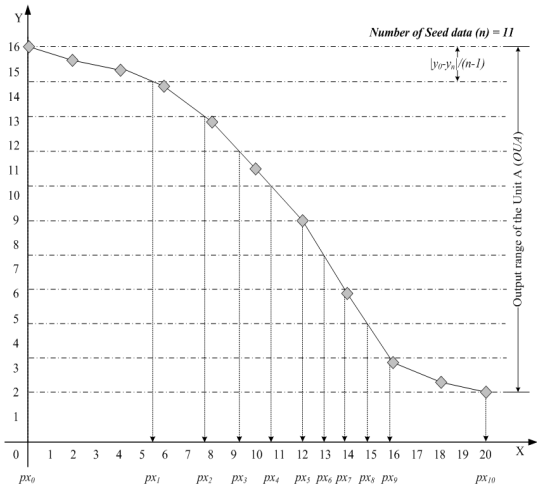

그림 5. ODR 생성 방법(A Strategy of ODR Generation)

때 이용되는 가상 그래프는 ***O***를 구성하는 각 출력 값을 그림 5에 나타난 것과 같이 직선으로 연결하여 작성한다. 정밀한 가상 그래프의 작성을 위해서는 LSM(Linear Square Method) 등 여러 가지 방법이 고려될 수 있으나, 입력 도메인을 분할하는 작업이 높은 정밀도를 요구하는 작업이 아니기 때문에, 이를 통해 얻어지는 이득이 요구되는 작업량에 비해 크지 않다고 판단되므로 단순히 직선을 연결한 형태의 가상 그래프를 생성하여 이용한다.

가상 그래프가 생성되면 ***OUA***를 균일 간격으로 |***S***|-1분할했을 때의 각 *y*값을 출력으로 가지는 입력 값들의 연산을 수행하고, 이 입력 값을 초기 입력 도메인 분할 경계(***ODR***)로 구성한다. 초기에 작성된 ***ODR*** 내의 데이터는 테스트 중에 변화되지 않고 그 값을 유지하게 되며, 입력 도메인의 요구 분할 수준이 ***ODR***을 구성하는 값의 수를 상회하는 경우에는 이를 동일 크기를 갖는 파티션으로 이분할 한다. 따라서 ***ODR***을 구성하는 값들의 간격은 예측된 가상 그래프의 형태 즉, 해당 구간에서의 기울기에 따라 서로 다른 간격을 가지지만, 이 값들은 균일한 간격을 가진 예상 출력 값을 바탕으로 생성된 것이기 때문에, 이를 파티션 분할의 기준점으로 사용하면 보다 균일한 출력 값을 유도할 수 있게 된다. ***ODR*** 생성 및 이를 ART에 적용하는 세부 절차는 표 1에 제시된 알고리즘과 같다. 기존 방법(Shin 등, 2010)의 경우 전체 입력 데이터의 분포 통계를 확인하는 작업을 수행하는데 반해, 본 논문에서 제안된 알고리즘에서는 샘플 데이터의 출력을 이용해 출력점간 직선 그래프를 작성하고, 이를 바탕으로 ***ODR***을 생성한다. 이를 통해 Unit A의 출력 분포를 구하는데 전

표 1. 분포 예측 기반 ART(DES-ART) 알고리즘(Distribution Estimation-based ART Algorithm)

D : 통합 시스템의 입력 도메인
S : 분포 예측을 위한 샘플 입력 x_i의 집합,
$\boldsymbol{S} = \{ x_0, x_1, x_2, \ldots, x_n \}$
x_0 = ***D***의 최솟값, x_n = ***D***의 최댓값,
$|x_1 - x_0| = |x_2 - x_1| = \ldots = |x_n - x_{n-1}|$
O : ***S***에 대한 Unit A의 출력 y_j 집합,
$\boldsymbol{O} = \{ y_0, y_1, y_2, \ldots, y_n \mid \text{Unit A}(x_i) \rightarrow y_j, x_i \in \boldsymbol{S}, i = j \}$
OUA : Output range of the Unit A, ***OUA*** = $[y_0, y_n]$
py_k : ***OUA***의 n 분할 시 각 경계, k = 0, 1, 2, ... , n
ODR(output distribution report): 출력 분포 보고서,
px_n : ***ODR***에 기록될 파티션 기준 경계.
ODR = $\{ px_1, px_2, \ldots, px_i \}$, $0 \leq i \leq n$
pl : ***D***의 분할 수준, $1 \leq pl$
ox_p : ***D***의 분할 시 이전 분할 수준에서의 경계 값, 이 때 p = 1, 2, 3, ... , n
cx_p : ***D***의 분할 시 현재 분할 수준에서의 경계 값

First Phase
1. ***D***로부터 ***S*** 생성
2. ***S***를 unit A에 적용하여 ***O*** 생성
3. ***OUA*** 연산
4. if ($y_0 > y_n$), $py_k := y_n + (| y_n - y_0 | / n) * k$, else ($y_0 < y_n$), $py_k := y_0 + (| y_n - y_0 | / n) * k$
5. $px_0 := x_0$, $px_n := x_n$
6. for each integer k, ($1 \leq k \leq n$-1)
   {
   조건 $y_{j-1} \leq py_k < y_j$를 만족하는 영역 $[y_{j-1}, y_j)$ 검색
   두 점 (x_i, y_j), (x_{i-1}, y_{j-1})을 지나는 $y = f(x)$을 작성
   이 선상에서 $py_k = f(px_k)$를 만족하는 px_k 연산
   $px_k \rightarrow$ ***ODR***에 추가
   }

Second Phase
1. if 첫 번째 입력 도메인 분할
   $cx_p := px_p$, p = 0, 1, 2, ... , n // $pl := |\boldsymbol{S}|$
   else // ***ODR*** 유지한 채로 각 파티션을 균등 이분할
   {
   $pl := 2pl$, $cx_0 := ox_0$,
   $cx_{2p-1} := cx_{2p-2} + (ox_p - ox_{p-1}) / 2$, $cx_{2p} := ox_p$
   }
2. Input Domain Tiling ART(Shin 등, 2010) 적용

체 입력 데이터를 이용하지 않아도 되도록 하고 있으며, 이는 테스트에 필요한 사전 작업량을 감소시키는 효과를 갖도록 한다.

표 2. 실험 환경(Simulation Environment)

Unit A	*probks, bessj0*
입력 도메인	1 dimension
오류 비율(θ)	0.01, 0005, 0.002, 0.001, 0.0005
오류 패턴	Block
오류 비율 당 실험 수	10,000
샘플 데이터 수	Case I - 30, 60, 120, 960 Case II - 30, 50, 100

4. 성능 평가

4.1 실험 환경

제안된 알고리즘의 성능 평가를 위한 시뮬레이션은 표 2와 같은 환경에서 수행되었다. 이 실험은 이미 배포된 두 가지의 프로그램을 출력 분포를 생성하는 Unit A로 사용하여 수행되었으며, 이 때 사용된 두 프로그램 모두는 C++ 언어를 이용해 작성된 간단한 수치 연산용 프로그램이다(Press 등, 2002). 테스트 대상 유닛에 존재하는 것으로 가정된 오류 영역은 블록 패턴을 따르되 다양한 오류 비율을 갖도록 하였고, 그 위치는 Unit B의 입력 도메인 내 임의의 위치에 생성되도록 하였으며, 블록의 크기는 입력 도메인의 크기와 오류 비율에 의해 결정하되 블록의 형태는 넓이와 높이가 동일하도록 구성하였다.

실험은 샘플 데이터의 수를 변화시켜가며 진행하되, 샘플 데이터를 순수하게 출력 분포 예측 목적으로 이용하는 경우(Case I)와 샘플 데이터 또한 테스트 케이스 가운데 일부로 사용하는 경우(Case II)로 나누어 진행하였다. 이 때 Case I의 경우에는 *probks*, *bessj0* 두 개의 Unit A 프로그램을 적용하였다. 하지만 Case II는 샘플 데이터를 테스트 케이스로 사용하는 경우에 발생되는 영향 파악이 주목적이므로 하나의 프로그램(*probks*)만 대상으로 하였다. 제안 알고리즘의 성능 평가는 각 오류 비율 당 10,000회의 시뮬레이션을 수행하고, 이 과정에서 얻은 F-measure의 평균을 이용하여 비교 평가 하였다.

4.2 실험 결과

그림 6과 7은 *probks*와 *bessj0* 프로그램을 Unit A로 사용했을 때 제안 알고리즘이 보이는 평균 상대 F-measure를 나타낸다. 그림에 나타난 것과 같이 제안 알고리즘을 이용했을 때의 성능은 두 프로그램 모두에서 약 0.56으로 ART를 단일 유닛을 대상으로 적용한 경우와 유사하게 나타났다. 이와 같은 값은 대상이 되는 시스템에 대해 제

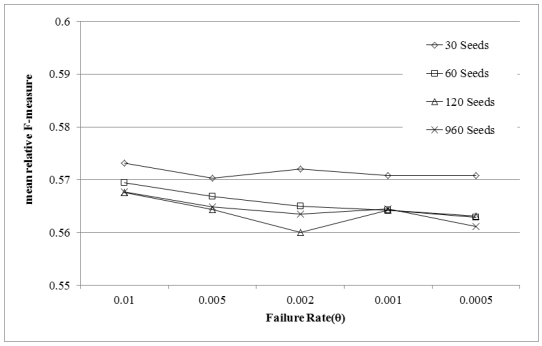

그림 6. 평균 상대 F-measure, *probks*(Mean Relative F-measure, *probks*)

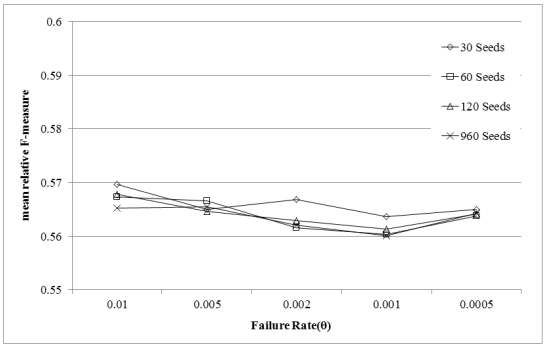

그림 7. 평균 상대 F-measure, *bessj0*(Mean Relative F-measure, *bessj0*)

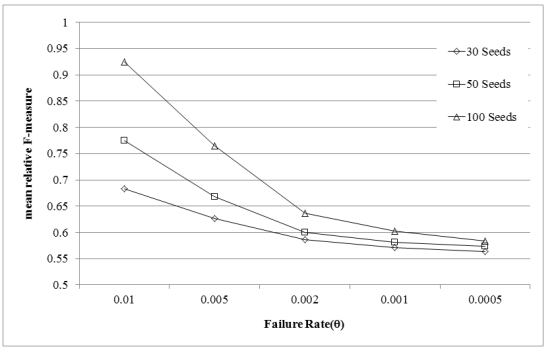

그림 8. 평균 상대 F-measure, 샘플 데이터 ∈ TC, *probks* (Mean Relative F-measure, Seed data ∈ TC, *probks*)

안 알고리즘을 적용하는 경우, 분포가 고려된 랜덤 테스팅을 적용했을 때에 비해 약 1/2 정도의 테스트 케이스를 이용하여 오류 영역을 검출할 수 있음을 의미한다. 또한 Unit A의 출력 분포가 고려되지 않은 경우 순수 랜덤 테스팅보다도 낮은 성능을 보이던 ART(Shin 등, 2010) 정책을 제안된 방법을 통해 안정적으로 적용할 수 있게 되었음을 보인다. 제안 알고리즘의 세부적인 성능 변화를 살펴보면, 우선 샘플 데이터의 수가 고정된 경우 오류 비율의 변화에 따라 1% 이내의 성능 변화를 보여 오류 비율에 의한 영향이 크지 않았다. 또한 많은 수의 샘플 데이터가 이용될수록 상대적으로 우수한 성능을 보이기는 하나, 성능의 최대 편차가 약 1% 수준으로, 사용된 샘플 데이터의 수에 의한 영향 또한 그리 크지 않는 것으로 확인되었다. 이는 통계학에서 이미 널리 알려진 바와 같이 30개 정도의 샘플로도 충분한 정확도를 가진 분포 예측이 가능하다는 사실이 이 실험에서도 확인되었으며, 제안된 알고리즘 또한 30개 정도의 적은 수의 샘플 데이터만으로도 충분한 효과를 얻어낼 수 있음을 의미한다.

그림 8은 샘플 데이터를 출력 분포를 파악하는 용도로만 사용하는데 그치지 않고 실제 테스트 케이스로도 이용한 경우의 제안 알고리즘의 성능을 *probks* 프로그램을 대상으로 평가한 결과를 제시하고 있다. 실험 결과에 따르면 서로 다른 수의 샘플 데이터를 사용한 경우와 서로 다른 크기의 오류 영역이 적용된 모든 경우에서 평균 상대 F-measure는 1이하로 랜덤을 적용하는 때보다 우수한 것으로 나타났다. 하지만 샘플 데이터를 테스트 케이스로 사용하지 않는 경우에 비해 성능은 나빠졌고, 더 많은 수의 샘플 데이터가 사용될수록 더 낮은 성능을 보이며, 오류 영역의 크기가 작을수록 성능이 개선되는 등의 차이점이 확인되었다. 이와 같은 현상은 샘플 데이터는 출력 분포에 대한 고려 없이 균일한 간격을 갖도록 선택되기 때문에 Unit B에 균일하지 못한 입력이 부여되는 상황을 유발하기 때문에 발생된다. 또한 이러한 현상을 유도하는 샘플 데이터의 수가 많아질수록 그 영향 또한 커지기 때문에, 많은 수의 샘플 데이터를 사용할수록 더 낮은 성능을 보이게 된다. 하지만 이러한 현상은 오류 영역의 크기가 작아 오류 영역을 검출하는데 소요되는 테스트 케이스의 수가 증가하여 전체 테스트 케이스 수에 비해 샘플 데이터가 차지하는 비율이 감소할수록 완화되는 현상을 보이게 된다.

5. 결 론

ART 정책은 효율성 개선 및 적용성 확장을 위한 다양한 형태의 시도를 통해 발전되어 왔다. 하지만 기존의 ART 정책은 하나의 시스템이나 유닛에 대한 테스트만을 고려하고 있기 때문에, 이를 다양한 테스팅 영역에 적용하기 어려운 제한점을 가지고 있다. 따라서 본 논문에서는 ART

정책의 테스팅 적용 범위 확장을 위한 시도의 하나로 통합 시스템의 테스팅에 적용 가능한 ART 기반 접근법을 제시하고 그 성능을 시뮬레이션을 통해 평가하였다.

통합 시스템은 다수 유닛의 결합체이기 때문에 이들 유닛 간의 상호 작용이 존재하게 된다. 따라서 시스템의 특정 유닛에 오류가 존재하는 것으로 추정되어 이를 ART를 이용해 테스트 하려고 하는 경우, 시스템에 주어지는 균일한 분포의 입력 데이터가 목적 유닛에 제대로 전달되지 못하여 테스트에 어려움을 겪게 된다. 따라서 본 논문에서는 목적 유닛에 부여되는 입력 즉, 타 유닛으로부터의 출력 분포를 샘플 데이터를 통해 획득하고, 이를 이용하여 목적 유닛에 균일한 입력 분포를 가진 입력이 부여되도록 함으로써 테스트의 효율성을 높이는 방법을 제안하고, 이를 실제 프로그램에 적용하여 평가하였다. 실험 결과에 따르면 제안된 방법은 오류 비율의 변화에 대해 안정적인 성능을 보였고, 그 성능 또한 ART를 유닛 테스트에 적용했을 때와 유사한 수준이었으며, 이러한 성능은 30개 정도의 적은 샘플 데이터로도 충분히 달성될 수 있음이 확인되었다.

하지만 실험에 적용된 기존 소프트웨어의 수가 2종으로 제한되어 있어 이를 통해 제안 알고리즘의 성능이 충분히 검증되었다고 하기는 어렵다. 따라서 명확한 검증을 위해서는 다양한 영역에서 사용되는 더 많은 수의 소프트웨어를 이용한 실험이 수행될 필요가 있다.

샘플 데이터의 사용은 테스트 초기에 출력 분포 확인을 위한 실제 테스팅 외 추가 부하를 유발하게 되므로, 샘플 데이터의 선택 및 적용 절차 없이 테스팅 과정에서 실시간으로 출력 분포를 확인할 수 있는 방법의 개발이 필요하다. 또한 현재 본 논문에서는 기존 소프트웨어인 Unit A의 출력 분포가 출력 도메인 내에서 불연속성을 보이지 않는 것으로 가정하고 있다. 하지만 일부 수치 데이터를 다루는 소프트웨어를 제외하면 대개의 소프트웨어는 이산 출력 분포를 갖는 경우가 많다. 이는 실제 테스트에 ART를 적용하기에는 현실적으로 아직 무리가 있음을 의미한다. 따라서 제안 알고리즘의 실용화를 위해서는 다양한 종류의 입력 형태 및 출력 분포에 대한 추가 고려가 필요하다.

이에 따라 향후 연구로 샘플 데이터를 요구하지 않으면서, 다양한 출력 분포를 다룰 수 있는 테스트 케이스 생성 정책 개발에 대한 연구를 수행할 예정이다.

참 고 문 헌

1. 신승훈, 박승규, “입력 도메인 확장을 이용한 반복 분할 기반의 적응적 랜덤 테스팅 기법,” 한국정보처리학회논문지 D, 15-D(4), pp. 531-540, 2008년 8월.
2. 신승훈, 박승규, 최경희, “테스트 케이스 분포 조절을 통한 IP-ART 기법의 성능 향상 정책,” 정보과학회논문지 : 소프트웨어 및 응용, 36(6), pp. 451-461, 2009년 6월.
3. F.T. Chan, T.Y. Chen, I.K. Mak and Y.T. Yu, “Proportional sampling strategy: guidelines for software testing practitioners,” Information and Software Technology, vol. 38, issue 12, pp. 775-782, Dec. 1996.
4. K.P. Chan, T.Y. Chen and D. Towey, “Restricted random testing,” Proc. of the 7th International Conference on Software Quality, LNCS 2349, pp. 321-330, 2002.
5. T.Y. Chen, D.H. Huang and Z.Q. Zhou, “Adaptive random testing through iterative partitioning,” Proc. of the 11th International Conference on Reliable Software Technologies, LNCS 4006, pp. 155-166, 2006.
6. T.Y. Chen, H. Leung and I.K. Mak, “Adaptive random testing,” Proc. of the 9th Asian Computing Science Conference, LNCS 3321, pp. 320-329, 2004.
7. T.Y. Chen and R. Merkel, “Efficient and effective random testing using the voronoi diagram,” Proceedings of the 2006 Australian Software Engineering Conference, pp. 300-305, Apr. 2006.
8. G.B. Finelli, “NASA software failure characterization experiments,” Reliability Engineering and System Safety, vol. 32, issues 1-2, pp. 155-169, 1991.
9. V. Ganesh, T. Leek and M. Rinard (2009), “Taint-based directed whitebox fuzzing,” Proceedings of the 31st International Conference on Software Engineering, pp. 474-484, May. 2009.
10. R. Hamlet, “Random testing,” Encyclopedia of Software Engineering, Wiley, pp. 970-978, 1994.
11. J. Mayer, “Adaptive random testing by bisection and localization,” Proc. of the 5th International Workshop on Formal Approaches to Testing of Software, LNCS 3997, pp. 72-86, 2006.
12. J. Mayer and C. Schneckenburger, “Adaptive random testing with enlarged input domain,” Proceedings of the 6th International Conference on Quality Software, pp. 251-258, Oct. 2006.
13. B.P. Miller, L. Fredriksen and B. So, “An empirical study of the reliability of UNIX utilities,” Communications of the ACM, vol. 33, issue 12, pp. 32-44, Dec. 1990.
14. A.B.M. Omar Faruk, “Testing & exploring vulnerabilities of the applications implementing DNP3 protocol,” Master

Thesis, KTH Royal Institute of Technology, Aug. 2008.
15. W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery, Numerical recipes in C++, the art of scientific computing, 2nd Ed., Cambridge University Press, pp. 237, 631, 2002
16. S.H. Shin, S.K. Park, K.H. Choi and K.H. Jung, "Normalized adaptive random test for integration test," Proceedings of the 2nd IEEE International Workshop on Software Test Automation, pp. 335-340, Jul. 2010.
17. M. Sutton, A. Greene and P. Amini, Fuzzing, brute force vulnerability discovery, Addison-Wesley, pp. 387-416, 2007.

신 승 훈 (sihnsh@ajou.ac.kr)

2000 아주대학교 정보 및 컴퓨터공학부 학사
2002 아주대학교 정보통신공학과 석사
2011 아주대학교 정보통신공학과 박사
2011～현재 아주대학교 정보컴퓨터공학부 강의교수

관심분야 : 소프트웨어 테스트 자동화, 멀티미디어 서비스 정책 등

박 승 규 (sparky@ajou.ac.kr)

1974 서울대학교 응용수학과 학사
1976 한국과학원(KAIST) 전산학과 석사
1982 Institut National Polytechnique de Grenoble 전산학과 박사
1992～현재 아주대학교 정보컴퓨터공학부 교수

관심분야 : 임베디드 테스팅, 자가 컴퓨팅/치료 시스템, 차세대 컴퓨터 구조 등

최 경 희 (khchoi@ajou.ac.kr)

1976 서울대학교 수학교육과 학사
1979 프랑스 그랑데꼴 ESEEIHT 석사
1982 프랑스 Paul Sabatier 대학 정보공학과 박사
1982～현재 아주대학교 정보컴퓨터공학부 교수

관심분야 : 운영체제, 분산시스템, 실시간 및 임베디드 시스템 테스팅 등

정 기 현 (khchung@ajou.ac.kr)

1984 서강대학교 전자공학과 학사
1988 미국 Ilinois주립대 EECS 석사
1990 미국 Purdue 대학 전기전자공학부 박사
1991～1992 현대반도체 연구소.
1993～현재 아주대학교 전자공학부교수.

관심분야 : 컴퓨터구조, VLSI 설계, 멀티미디어 및 실시간 시스템 등