연구제안서

(간략본)

소프트웨어 무결점 연구센터
http://rosaec.snu.ac.kr/

08/2008

제대로 작동할 지를 미리 검증할 수 없는 기계설계는 없다. 제대로 서 있을지를 미리 검증할 수 없는 건축설계는 없다. 인공물이 자연세계에서 문제없이 작동할지를 미리 엄밀하게 분석하고 검증하는 기술은 잘 발달해 왔다.[1]

이러한 분석 검증 기술은 다른 엔지니어링 분야에서는 당연히 확립된 기술이다. 왜냐하면 모든 엔지니어링의 제일 근본적인 질문이 바로, 우리가 만든 것이 우리가 의도한대로 움직인다는 것을 어떻게 확인할 수 있는가? 이기 때문이다.

컴퓨터 소프트웨어에 대해서는 어떤가? 작성한 소프트웨어가 제대로 실행될지를 미리 엄밀하게 확인해 주는 기술은 있는가? 그래서, 작성한 소프트웨어가 가질 수 있는 오류를 자동으로 미리 모두 찾아주거나, 없으면 없다고 확인해 주는 기술은 있는가? 그래서, 소프트웨어의 오류 때문에 발생하는 개인/기업/국가/사회적 비용을 절감시켜주는 기술은 있는가?

오류 검출 및 검증 기술의 첨단에서 실용가능한 기술을 규명해내고 남들보다 먼저 산업화해내는 그룹은 확산 직전에 있는 거대한 무결점 소프트웨어 개발 도구 시장의 주도권을 잡게 될 것이다.

1 배경 및 제안

- 컴퓨터 소프트웨어의 오류 문제**:** 컴퓨터 소프트웨어가 개인과 사회의 인프라가 되면서, 소프트웨어 오류의 문제를 지금대로 방치할 수는 없다는 압력은 나날이 커지고 있다. 소프트웨어의 오류로 생기는 사회경제적 비용이 급격히 증가하고 있기 때문이다. 미국 표준연구소(National Institute of Standards and Technology)의 2002년 리포트[2]에 따르면, 소프

[1] 뉴튼역학, 미적분 방정식, 유체역학, 통계역학 등이 그러한 기술들일 것이다.

[2] "The economic impacts of inadequate infrastructure for software testing." Program Office Strategic Planning and Economic Analysis Group, National Institute of Standards and

트웨어의 오류로 미국경제가 지불한 비용은 595억 달러로 추정되고 있다. 우리의 경제규모(미국의 1/20)로 추정해보면 년 3조원 정도가 된다.[3]

- 무결점 소프트웨어 시장**:** 또한 무결점 소프트웨어가 시장에서 절실히 필요해지는 이유를 따로 부연할 필요는 없을 것이다. 소프트웨어 자체가 상품인 경우는 말할 것도 없고, 거의 모든 제품(기계, 교통, 통신, 국방, 에너지, 의료, 가전)에 내장된 소프트웨어의 신뢰도가 제품의 경쟁력을 결정짓고 있다.[4]

- 소프트웨어 녹색기술**:** 더군다나 요즘 "녹색 기술(green technology)"이라는 전 지구적인 화두에 컴퓨터 소프트웨어를 대입해 보면, 그 핵심 요소기술로는 오류 없는 소프트웨어를 제작하는 기술이 있게 된다. 낭비없이 안전하게 지속하는 세상은 그 세상을 지탱시키는 소프트웨어들의 무결점과 직결되기 때문이다. 오류 없는 소프트웨어는 오랫동안 재사용 될 것이고 헛되이 멈추거나 잘못 진행되지 않으므로 에너지 낭비가 없다.[5]

- 제안**:** 문제는, 어떻게 하면 그러한 무결점 소프트웨어를 값싸게 만들어 낼 수 있는가인데, 본 센터가 이에 대한 실용적인 답안을 연구개발하고자 한다.

2 연구 목표

- 수요 격발**:** 본 센터는 위의 목표를 달성하는 과정에서 제대로 개발된 솔루션들을 내놓게 될 것이고, 이 솔루션들은 폭발 직전에 있는 무결점 소프트웨어 개발 도구 시장[6]의 수요를 격발할 것으로 보인다.

Technology, 2002. `www.nist.gov/director/prog-ofc/report02-3.pdf`

[3] 예를 들어, 국내 S정보기기(주)는 휴대정보기기의 소프트웨어 오류로 800억원 상당을 반품처리하였고, 국내 모 휴대폰은 "버그폰", "뽑기폰"이라는 별명을 얻어 해당 회사의 브랜드 가치를 현저하게 떨어뜨리고 있다. 또한 자동차 급발진 사고들이나 T-money의 초창기 문제들도 소프트웨어 오류 때문인 것으로 추정된다.

[4] 미 의회 일반회계 사무소(US Congress General Accounting Office)의 리포트에 따르면 최신예 전투기 F/A-22 Raptor 개발 비용의 80%가 무결점 소프트웨어 개발 비용이다. 국내에서 개발한 초음속 전투 훈련기 T-50의 경우도, 1대 출고 때 마다 소프트웨어 비용으로 McDonald Douglas사에 약 30억원을 지불한다.

[5] 물론 소프트웨어의 실행 비용(전기와 메모리 소모량 등)을 최소로 하는 기술도 녹색기술의 한 축이겠지만, 이와 관련해서는 원천기술 선점을 통한 산업화 기회는 상대적으로 풍부하지 못하다. 오래전부터 이미 소프트웨어의 실행 속도와 자원소모량 최적화라는 문제를 통해서 많은 성과가 나와 있기 때문이다.

[6] 이 시장의 크기를 유추할 수 있는 사실들: (1) 소프트웨어의 오류로 미국경제가 2002년 지불한 비용 중에서, 유효한 SW 개발 도구가 있었다면 줄일 수 있었을 비용을 106억 달러(10조6000억원)로 추정하고 있다. (2) 2010년도의 전세계 소프트웨어 시장의 규모가 3100억 달러(310조)로 예측되고 있으므로 소프트웨어 개발에 투자되는 비용 중 이러한 도구에 투자되는 비용으로도 시장 크기가 유추될 수 있다. 소프트웨어 개발비 중에서 오류 수정에 약 70%가 소모된다고 알려져 있다.

원천기술 선도	오류 없는 소프트웨어를 저렴하게 생산 가능하게 하는 원천기술의 한 축을 세계적으로 선도하고
실용적 도구 개발	그에 기반한 실용적인, 소프트웨어 소스 오류 자동 검출 및 검증 도구들을 개발하며
특화된 산업화	이 도구들을 국내 산업과 밀접히 연계되도록, 순수 소프트웨어 개발뿐 아니라 지능형 로봇/금융공학/무인 비행체 소프트웨어에 특화시켜 산업화한다.

- 주도권 선점**:** 물론 본 센터의 성과로 소프트웨어 오류 자동 검증의 꿈이 완전히 실현될 수는 없다. 사실 소프트웨어의 무결점을 완전히 보장해 주는 완성된 자동 기술은 달성되지 않을지도 모른다. 하지만 조금 더 간편하고 적은 비용으로 좀 더 많은 결점들을 자동으로 검증해 주는 기술은 계속해서 만들어질 것이고, 그러한 기술을 달성하고 현장에 발 빠르게 적용하는 그룹이 소프트웨어 기술 시장의 주도권을 잡을 것으로 보인다.

연구 초점	본 센터는 매우 넓은 오류 자동 검증기술의 스펙트럼[7]중에서 기술 난이도와 장벽이 높지만 실용화 가능성이 우수하고 큰 리턴을 가져오는 부분에 집중한다.
연구 목표의 세부 사항	• 원천기술: 의미기반 정적분석 (semantic-based static analysis) • 방향: 정통적인 원천기술 심화 + 혁신적인 신 기술 연구 • 목표 소프트웨어 오류: 안전성 오류 + 보안 오류 + 기능성 오류 • 목표 도구: 오류 검출기 + 무결점 검증기 + 특질 추출기 + 혁신적인 도구 • 실용성: 전자동 도구에 집중 + 기술의 제한점 대응 완성

정통적인 (orthodox) 원천기술의 심화

이 목표를 위해 집중하는 원천기술은 의미기반 정적분석 **(semantic-based static analysis)** 기술이다.

- 이 기술은 엄밀히 예측한다**:** 이 기술은 소프트웨어를 실행시키지 않고 소프트웨어의 소스만을 분석해서 실행 중에 일어날 일을 엄밀히 예측하는 기술이다. 이 기술은 다양한 이름의 많은 기술들을 모두 포괄한다. “static analysis”, “abstract interpretation”, “type system”, “software

[7]오류 자동 검증 기술의 스펙트럼은 두 개의 축을 따라 펼쳐진다. 소프트웨어 개발 과정의 축에 따라 필요한 오류 자동 검증 기술들과, 소프트웨어 핵심 로직의 복잡성 축에 따라 필요한 오류 자동검증 기술들이 있다. 이 많은 기술들은 다양한 난이도와 기술수준을 가진다.

model checking", "program logics", "proof system" 등이 포함된다. 모두 대상 소프트웨어 소스 언어의 의미구조(semantics)에 대한 엄밀하고 정밀한 정의를 기반으로 소프트웨어가 실행중에 가질 수 있는 모든 가능한 상황을 소스 분석만을 통해서 유한한 시간내에 계산해내는 기술이다.

- 이 기술은 테스팅의 단점을 보완한다**:** 테스팅은 프로그램을 실행시켜야 하고, 유한개의 입력에 대해서만 제대로 작동된다는 것을 확인할 수 있을 뿐이다. 테스팅에서 제외된 경우에 오류가 숨어있는 경우는 항상 존재한다.

  정적분석은 프로그램을 실행시키지 않고도 프로그램에 오류가 없는지를 확인시켜 줄 수 있다. 특히, 그 확인 과정이 또 다른 소프트웨어(분석기)를 통해서 자동으로 이루어진다.

  이 기술은 소프트웨어 개발의 초기 단계(디자인이나 구현 단계)에서 오류를 찾아주기 때문에 오류 수정 비용을 현저하게 줄여준다. 모든 실행환경이 구비되거나 소비자에게 전달된 후에 오류가 발견되어 야기되는 막대한 수정 비용을 없애준다.

- 이 기술의 실용성은 확인되었다**:** 본 연구진은 10년 이상의 집중된 연구를 통해서 이 기술이 소프트웨어 오류 자동 검증기술로 응용될 수 있다는 것을 확인하였다. 이 확인은 단순한 실험실 시제품 개발이 아니라, 실제 국내외 시장을 개척할 제품으로서 본 연구진이 최근 산업화를 완료한 Sparrow라는 시스템[8]을 통해서 이루어졌다.

혁신적인 (unorthodox) 분석기술 연구

의미기반 분석의 한계를 극복해 줄 가능성이 있는 혁신적인 방법에 대해서도 연구할 것이다.

- 분석**(analysis)**보다는 종합**(synthesis)**하는 방식**:** 많은 양의 소프트웨어 소스와 거기서 발견된 오류들에 대한 초대형 히스토리 데이터베이스(corpus)를 기반으로, 소프트웨어의 오류를 유추해 낼 수 있는 방법을 연구할 것이다.

[8] Sparrow(`www.spa-arrow.com`)는 C 소스에 있는 메모리 접근 오류(buffer overrun)와 메모리 누수 오류(memory leak) 등 10가지의 오류를 자동으로 찾아주는 분석기이다. SPARROW의 현재 성능은 Pentium 4, 3.2GHz, 4Gbyte 기계로 초당 100 줄의 분석 속도이고, 평균 1000줄 당 6개의 오류를 예측해 준다. 수백만 줄 단위의 C 소스를 전자동으로 분석한다. 현재 국내 및 해외 시장에서 선진국의 경쟁제품보다 우수하거나 차별성이 있다는 평가를 받고 있다. SPARROW의 성능에 대한 자세한 내용은 홈페이지와 아래 논문 및 발표자료(MIT, CMU, MIT Lincoln Lab.에서 발표) 참고: `ropas.snu.ac.kr/~kwang/paper/08-ismm-juyi.pdf`와 `ropas.snu.ac.kr/~kwang/paper/YiMITCMULL08.pdf`.

- 실행전 분석**(static analysis)**과 실행중 분석**(dynamic analysis)**의 조합**:** 소프트웨어를 실행해 보면서 분석하는 기술은 한계가 많지만 장점만을 취한다면 정적분석의 단점을 보완할 가능성이 있다. 이에 대한 연구를 진행할 것이다.

목표로 하는 소프트웨어 오류들

- 안전성 오류**(safety bug):** 소프트웨어의 실행을 파행적으로 몰고 가는 오류이다. 메모리 접근오류(buffer overrun), 메모리 누수 오류(memory leak), 수치값 상하한 오류(over/underflow), 나누기-0 오류(divide-by-0), 메모리 초기화 오류(uninitialized access), 유효하지 않은 메모리 주소 접근오류(null dereference), 다형성 구현오류(parametric/subtype-polymorphism error) 등이 포함된다.

- 보안 오류**(security bug):** 해커가 소프트웨어의 실행을 얻어 타고 대상 컴퓨터 시스템에 침입할 수 있는 여지를 주는, 소프트웨어 소스에 있는 오류들이다. 이러한 오류는 해커의 침입기술이 규명된 후에야 정의될 수 있다. 지금까지 알려진 해커의 침입기술[9]은 웹 소프트웨어의 경우, 입력 스트링 메모리 접근 오류, 크로스사이트스크립팅 오류(XSS), SQL 삽입 취약 오류(SQL injection vulnerability) 등이 있다.

목표로 하는 오류 검증 도구들

- 소스 오류 자동 검출기**(bug-finder):** 소프트웨어 소스의 실행의미를 분석해서, 오류의 지점을 자동으로 찾아준다.

- 소스 무결점 자동 검증기**(verifier):** 오류 검출기를 특정 소프트웨어에 특화시켜서, 허위 경보 없이 목표로 하는 오류를 모두 찾아준다.

- 소스 특질 자동 추출기**(property-finder):** 검출된 오류의 진위여부를 사람이 쉽게 파악할 수 있도록 소스의 여러 특질을 분석한다.

- 혁신적인 도구들**:** 정통기술에 기초하지 않지만 효과적일 수 있는 도구들도 개발한다. 예를 들어, 데이터 집약형 분석기(data-intensive/corpus-based analyzer)나 정적/동적 분석을 조합한 하이브리드 분석기(hybrid analyzer) 등이다.

[9] CERT(`www.cert.org`)나 MITRE(`cve.mitre.org`)가 중심이 되어 침입기술들과 문제점들이 공개되고 있다. C 소프트웨어의 경우, 스택 스매싱(stack smashing)이나 힙 스매싱(heap smashing) 등이 있다.

자동화 목표

위의 도구들을 모두 전자동 도구들(one-button solutions)로 개발해 낸다. 전자동이 필요한 이유는 두 가지다.

- 소프트웨어의 크기: 지난 30년간 소프트웨어의 크기는 기하급수적으로 커져왔다.[10] 사람이 손수 소프트웨어를 검증하는 것은 거의 불가능하다.
- 수요 폭발: 전자동 기술이 완성되어야 수요를 폭발시킨다. 마치 전자동 전화 스위칭 시스템이 전화 교환원을 대체하면서 전화의 수요를 폭발시켰던 것과 같다.

  본 센터는 실용가능성을 규명하는 데 멈추지 않는다. 그 가능성을 실용적인 전자동 도구의 개발로 드라이브해 갈 것이다. 기술들을 최대로 자동화하는 데에도 세계적인 선도에 설 것이다.

우리는 의식하지 않고 사용하게 되는 기술을 목표로 하는 셈이다. 최대한 성숙한 기술은 사용하는 사람이 사용하는지도 모르게 사용하는 기술일 것이다.

정적분석의 제한점에 대한 대응

- 정적분석 기술의 두 가지 제한점: 소프트웨어의 텍스트를 가지고 실행 중 일어날 모든 상황을 자동으로 예측하는 데서 오는 근본적인 제약이다.
  - 허위 경보(false alarm):이론적으로 모든 전자동 정적분석기는 검증하고자 하는 오류 지점을 소스에서 모두 찾아 낼 수는 있지만, 오류가 아닌 지점들도 오류라고 지적해 주는 허위 경보가 항상 발생하게 된다. 허위 경보를 줄이려면 분석기의 정확도를 증가시켜야 하는데, 결국은 분석비용이 너무 커져서 자동화가 불가능해질 수 있다.
  - 마의 삼각형(elusive triangle): 실용적인 정적분석기가 동시에 갖출 수 없는 세 가지다. 분석실행의 저비용(scalability), 찾는 오류의 깊은 논리적 성질(deep property), 그리고 분석기의 자동화(automation)이다. 분석기는 항상 이 셋 중 하나를 양보해야 한다. 예를 들어 대형의 소프트웨어를 빨리 분석하면서(scalable) 자동화도 가능하게 하려면 분석해야 할 성질이 논리적으로 아주 복잡할 수는 없다.
- 센터의 대응 방법

[10] 30년전 소프트웨어의 크기가 서울에서 부산까지의 거리에 해당한다면 현재는 지구에서 화성까지의 거리만큼으로 증가했다. 2010년경 고급 승용차에 탑재되는 소프트웨어의 크기는 1억 줄까지 될 것으로 예상된다. 대형 의료장비에는 약 2천만 줄의 소프트웨어가 탑재되어 있다. "소프트웨어는 가스다"라는 말이 있다. 소프트웨어의 크기도 그렇지만 실행에 소모되는 자원은 주어진 컴퓨터의 모든 용량을 항상 꽉 채운다는 성질을 빗댄 것이다.

– 소스 오류 검출기(bug-finder): 허위 경보를 줄이는 대신 놓치는 오류를 허용한다(unsound). 대신에, 전자동이면서(automatic) 분석기 실행의 비용이 저렴하여(scalable) 초대형(천만 줄 단위의) 소스의 분석이 24시간 내에 가능하고, 대상 소스 언어로 작성된 임의의 소프트웨어를 효과적으로 분석할 수 있도록 한다(domain-independent).

– 소스 무결점 검증기(verifier): 대상 소프트웨어를 특정한 것으로 고정한다(domain-specific). 대신에 허위 경보가 없고(zero false alarm), 놓치는 오류가 없고(sound), 전자동이다(automatic). 이 검증기는 안전성이 특별히 중요한 핵심 소프트웨어의 소스에 특화되어 하나씩 개발된다.

– 기능성 오류의 종류: 범위가 좁혀진다. 기능성 오류는 안전성 오류나 보안 오류에 비해서 임의의 깊이를 가지는 논리적 복잡성을 가질 수 있다.[11] 기능성 오류를 검증해 주는 자동 도구들은 제한된 기능들의 검증으로 전문화될 수밖에 없다.

3 최종 및 단계별 목표

용어정의

$\text{허위경보율} = \frac{\text{허위 경보의 개수}}{\text{전체경보의개수}}$	분석기가 내놓는 전체 오류 경보들에서 허위 경보들의 비율. 낮을수록 좋다. 분석기는 소스에서 오류일 수 있는 지점을 자동으로 찾아 주는데, 그 중에는 오류가 아닌데도 오류라고 지적할 가능성이 항상 있다.
$\text{분석효율} = \frac{\text{검출 오류 갯수/허위경보율}}{\text{전체경보의개수}}$	분석기의 효율은 실제 오류들을 많이 찾을수록 좋고 허위경보율이 낮을수록 좋다. 따라서 분석효율은 소스 1000줄(KLoC)당 실제 오류들의 갯수를 허위경보율로 나눈다. 높을수록 좋다.

- 오류 검출기**(bug-finder):** 소프트웨어 소스에 있는 오류를 자동으로 찾아준다. 대상 오류를 모두 찾는 것을 보장하지 않지만(not sound), 허위 오류가 적고(low false alarm ratio), 대상 소스 언어로 짜여진 임의의 소프

[11] 기능성 오류에 대한 다양한 논리적 깊이의 예를 들면, 한 변수의 값이 항상 0과 10 사이를 유지해야 하는 데 그렇지 못한 경우에서 부터, 한 변수가 입력과 일정한 관계를 항상 유지해야 하는데 그 관계가 임의로 복잡해지는 경우 등이다.

트웨어들에 유효하게 작동한다(domain-independent).
허위경보율 $\leq 20\%$. 분석효율 ≥ 20.

- 무결점 검증기**(verifier):** 소프트웨어 소스에 있는 오류를 자동을 찾아준다. 대상 오류를 모두 찾는 것이 보장되고(sound), 허위 오류가 없다(zero false alarm). 단점으로는 대상 소스 언어로 짜여진 소프트웨어 중에서 특정 부류의 소프트웨어에만 유효하게 작동한다(domain-specific). 오류 검출기보다 정확도나 분석의 깊이가 깊지만 위의 장점을 위해서 대상 소프트웨어에 특화될 필요가 있다.
허위경보율 $\leq 1\%$. 분석효율 ≥ 150.

최종 목표

목표	소프트웨어의 오류를 소스 레벨에서 검출/검증해주는 전자동 도구들을 연구개발하고 실용화를 통해서 국내 산업계의 국제 경쟁력에 이바지하고 원천기술을 확보한다.
세부목표	국내 산업과 밀접하게 결합되는 5개의 선도 프로젝트(flagship projects) • 시스템 소프트웨어의 소스 오류 자동 검증 도구의 개발 • 응용 소프트웨어의 소스 오류 자동 검증 도구의 개발 • 지능형 로봇 소프트웨어의 소스 오류 자동 검증 도구의 개발 • 금융공학 소프트웨어의 소스 오류 자동 검증 도구의 개발 • 무인 비행체 소프트웨어의 소스 오류 자동 검증 도구의 개발

- 목표 도구**:** 각 선도 프로젝트에서 개발하는 도구들은 오류 자동 검출기와, 각각의 핵심 소프트웨어에 특화된 무결점 자동 검증기들이다.

- 분석 대상**:** 각 선도 프로젝트의 대상 소프트웨어들은 각각의 분야에서 핵심적으로 사용되는 대형 소프트웨어들이다.

- 직접적인 결과물**:** 그러한 도구들과 더불어 오류 없음이 확인된 핵심 소프트웨어들, 그리고 그 오류 검증 과정을 엄밀하고 기계적으로 확인할 수 있도록 하는 정형적인 검증 기록들이다.

- 간접적인 결과물**:** 그러한 성과를 달성하기 위해서 새롭게 확장 고안된 의미기반 정적분석 기술과 그외의 다양하고 엄밀한 분석기술들이 될 것

이다. 이러한 기술들은 모두 국제적으로 선도적인 논문으로 발표되고 국내외 특허로 가치를 보전할 것이다.

단계별 목표

1단계 4년은 위의 세부목표 중에서 오류 검출기(bug-finder)의 개발을 완료하고 무결점 검증기로 확장하기 위한 발판을 마련한다. 2단계 3년은 무결점 검증기(verifier)들을 완성한다. 특질 추출기(property-finder)와 혁신적인 도구들은 오류 검출기와 무결점 검증기에 포함된다.

1단계(4년): 오류 검출기 (bug-finder) 단계	• 시스템 소프트웨어 안정성 및 보안 오류 검출기 완성 • 응용 소프트웨어 안정성 및 보안 오류 검출기 완성 • 지능형 로봇 소프트웨어 안전성 및 기능성 오류 검출기 완성 • 금융공학 소프트웨어 안전성 및 기능성 오류 검출기 완성 • 무인 비행체 소프트웨어 안전성 및 기능성 오류 검출기 완성
2단계(3년): 무결점 검증기 (verifier) 단계	• 시스템 소프트웨어 안정성 및 보안 무결점 검증기 완성 • 응용 소프트웨어 안정성 및 보안 무결점 검증기 완성 • 지능형 로봇 소프트웨어 안전성 및 기능성 무결점 검증기 완성 • 금융공학 소프트웨어 안전성 및 기능성 무결점 검증기 완성 • 무인 비행체 소프트웨어 안전성 및 기능성 무결점 검증기 완성

□