편집자: 크리스토프 에버트
벡터 컨설팅 서비스 christof.ebert@vector.com

소프트웨어 시스템 테스트

크리스토프 에버트, 디비스 바지즈, 마이클 웨이리히

편집기에서

테스트는 종종 지루한 작업으로 여겨집니다. 틀렸습니다! 테스트는 여러 가지 이유로 중요합니다. 제품 수명 주기 전반에 걸쳐 대부분의 비용이 투입되는 영역일 뿐만 아니라 적응형 소프트웨어, 애자일 배포, 인공 지능(AI) 기반 시스템과 같은 새로운 과제에는 새로운 테스트 기술이 필요합니다. 마이클 웨이리히, 디비스 바자즈, 그리고 저는 소프트웨어 테스트에 대한 현재 기술을 평가하고 각 테스트 활동에 대한 체계적인 위험 평가를 포함하여 방법과 도구에 대한 전반적인 관점을 제공합니다. AI 시스템 테스트

재 버전 날짜: 2022년 6월 20일

소프트웨어가 다음을 소비하고 있습니다.

결함으로 인해 소프트웨어가 소모되고 있습니다. 우리는 사회 전체를 소프트웨어에 맡기면서도 근본적인 품질에 대해서는 거의 생각하지 않습니다. 하지만 품질, 특히 책임은 중요합니다. 불충분한 소프트웨어 품질로 인한 제품 리콜 건수와 법적 조치 금액이 빠르게 증가하고 있습니다. 물론 소프트웨어 엔지니어로서 우리는 품질은 테스트가 아니라 제품에 설계되어야 한다는 것을 알고 있습니다. 하지만 인간으로서 우리는 결함이 발생한다는 사실도 알고 있습니다. 체계적인 검증 및 유효성 검사(V&V)를 피할 수 있는 방법은 없습니다.

디지털 객체 식별자 10.1109/MS.2022.3166755 현

이버 보안 공격,
이 사용성이 빠르

고 있으며, 수명 주기 예산의 절반 이상을 프로세스 및 제품의 안전에 중요한 시스템[1-4]에 투자하고 있습니다. 민첩성이 향상되고 지속적인 업데이트가 이루어지면서 테스트 활동의 수가 증가하고 있습니다. 예를 들어, 이는 일반적인 연기 테스트를 넘어서는 지속적인 검증과 지속적인 유효성 검사를 의미합니다. 연속-x 파이프라인을 통한 자동화가 필요하며, 자율 시스템의 인지 테스트(CT)와 같이 완전히 새로운 테스트 체계의 경우 더욱 그렇습니다.

테스트가 왜 그렇게 어려운가요? 다음과 같이 소프트웨어로 인한 위험과 문제가 꾸준히 증가하고 있습니다.

게 증가하고 있습니다. 따라서 테스트는 더욱 빠르게 진화해야 합니다. 동시에 품질 향상을 위한 준비도 병행해야 합니다. 테스트 전략은 결함을 찾아내는 것뿐만 아니라 시스템을 강화하여 견고성을 확보하는 것을 의미합니다. 수정 및 변경 사항은 무선으로 안정적으로 유동적인 방식으로 배포되어야 합니다. 점진적인 성능 저하 및 운영 실패 시나리오와 같은 복원력 전략을 설계하고 배포해야 합니다.

지속적 통합(CI)/지속적 배포(CD) 프로세스의 결과로 체계적인 지속적 V&V가 요구되는 시대입니다. 하지만 다음과 같은 이유로 인해 실제로는 거의 달성되지 않습니다.[1-5]

0740-7459/22©2022IEEE

- 지속적인 업데이트와 유연한 데브옵스 배포로 인한 상태 폭발로 인해 기능 테스트 및 코드 커버리지와 같은 전통적인 V&V 방법이 불가능해졌습니다.
- 인터페이스 및 통합 실패의 증가에서 볼 수 있듯이 정의된 소프트웨어 프로세스의 강점은 민첩한 "프로세스 없음" 태도 하에서 약화되었습니다.
- 연속 X는 어제 특정 품질 수준으로 배포된 소프트웨어가 오늘 훨씬 낮은 품질로 업데이트된다는 의미입니다.
- 시스템 복잡성이 점점 더 높아지고 시간 압박이 증가함에 따라 결함 및 사이버 보안 취약성이 증가하고 있습니다.
- 사이버 범죄와 공격은 전 세계 국가와 영리 조직에 의해 이루어지고 있으며, 그 규모는 전 세계 마약 밀매보다 더 큽니다.
- 머신 러닝(ML), 인 공 지능(AI), 적응형 플랫폼은 기반 소프트웨어를 지속적으로 변화시키므로 기존의 커버리지 체계와 기능을 금지합니다.
  테스트.
- 복잡성과 사용 가능한 컴퍼턴스 및 용량 간의 격차가 커지면서 숙련되지 않은 사람들이 개발하는 소프트웨어의 양이 점점 더 많아지고 있습니다.

> 연속-X 파이프라인을 통한 자동화가 필요하며, 자율 시스템의 인지 테스트와 같이 완전히 새로운 테스트 체계의 경우 더욱 그렇습니다.

트이 있다고 믿고 자란 새
로운 세대의 개발자들이 능
장한면서 기존의 품질에 대
한 태도는 점점 좋어들고
있습니다.

고도로 분산된 서비스, 자율 시스템의 자동 기능, 점점 더 복잡해지는 현실 세계와의 상호작용을 고려할 때 완화는 결코 사소한 일이 아닙니다. 이로 인해 많은

자율 주행 차량 시스템의 검증에 대한 질문입니다. AI와 머신러닝은 알고리즘적 투명성을 충족해야 합니다. 예를 들어, 더 이상 알고리즘으로 설명할 수 없는 신경망의 규칙은 무엇이며, 자율 주행 차량이 동시에 여러 위험에 직면했을 때 누가 공로를 인정받을지 또는 어떻게 반응할지 결정할 수 있을까요? 기존의 추적성 및 회귀 테스트는 확실히 작동하지 않을 것입니다. 오히려 미래의 V&V 도구에는 빅데이터 익스플로잇, 비즈니스 인텔리전스, 시스템 자체 학습을 기반으로 하는 더 많은 인텔리전스가 포함되어 동적인 방식으로 소프트웨어 품질을 개선할 것입니다.[5]

체계적인 테스트는 다음과 같은 다양한 관점을 고려해야 합니다[2,3,5]:

- 지정된 기능은 무엇인가요?
- 시스템이 사양을 준수하나요?
- 모든 관련성 있고 중요한 상황과 그 상관관계가 적절하게 명시되어 있나요?
- 의도된 기능성은 무엇이며 안전한가요?
- 의사 결정을 추적하고 이에 대한 판단을 내리는 방법은 무엇인가요? 이를 어떻게 감독할 수 있을까요?
- 장애 발생 시 신뢰성을 어떻게 정의할 수 있을까요?

- 의도한 기능이 실수, 부적절한 설계 또는 공격으로 인해 실패하면 어떻게 될까요?
- 어떤 경우에도 금지되어야 하는 부정적인 결과는 무엇인가요?
- 어떤 적응형 또는 학습 하위 니까?
- 소프트 웨어 업데이트 및 ML에 대한 어떤 정보를 사용자에게 제공해야 하나요?
- 지속적인 개발 주기에서 테스트 종료 기준을 어떻게 식별할 수 있나요?
- AI 기반 시스템에서 변경, 업데이트, 학습 시 실행 가능한 최소한의 테스트 전략은 무엇인가요?

합니다. 법적인 이유로 주문자 상표 부착 생산업체는 제품이 충분한 테스트를 거쳤음을 입증해야 합니다.

테스트 기술

테스트는 지난 수십 년 동안의 무차별적이고 임시방편적인 접근 방식에서 벗어나고 있습니다. 테스트 중심 요구사항 엔지니어링 및 테스트 중심 개발(TDD)과 같은 방법을 사용하여 설계 전에 테스트를 시작합니다.[4] 소프트웨어 품질을 달성하기 위한 좋은 기반은 ISO/국제전기기술위원회(IEC) 12207 표준 및 해당 소프트웨어입니다.

> 소프트웨어로 인한 위험과 문제는 다음과 같이 꾸준히 증가하고 있습니다. 결함, 사이버 보안 공격, 불충분한 사용성이 빠르게 증가하는 것을 볼 수 있습니다.

V&V 프로세스.[6] 왼쪽(개발)의 각 프로세스에 대해 오른쪽(테스트)에 해당하는 V&V 프로세스가 있는 V자형 추상화를 사용합니다. 이를 기반으로 추적성으로 연결된 세 가지 영역, 즉 설계 및 검증 요구 사항을 구분할 수 있습니다. ISO/IEC/ IEEE 29119는 다양한 소프트웨어 테스트 활동을 보다 정확하게 명시하고 있습니다.[7] 그림 1은 지난 세기의 TV 시리즈 이름에서 따온 '*트리플 피크*'라고 불리는 이 세 가지 영역을 보여줍니다. 이 추상화에서 단위 테스트는 소스 코드가 로우레벨 설계를 준수하는지 여부를 확인합니다. 통합 테스트는 다음을 확인합니다.

이전에 테스트한 구성 요소의 적합성, 즉 통합된 방식으로 작동하는지 여부를 확인합니다. 시스템 테스트는 완전히 통합된 제품이 사양을 충족하는지 여부를 확인합니다. 마지막으로 수락 테스트는 제품이 사용자 또는 고객의 기대치를 충족하는지 여부를 확인합니다.

시스템 및 소프트웨어 엔지니어링에 관한 ISO 25000 시리즈는 다음과 같이 주요 제품 품질 특성 몇 가지를 정의하여 품질 보증 및 테스트 요구 사항[5]을 지원합니다:

- *기능적 적합성*: 제품의 기능적 적합성 정도 또는 시스템은 지정된 조건에서 사용할 때 명시적 및 묵시적 요구 사항을 충족하는 기능을 제공합니다.
- *보안*: 제품 또는 시스템이 다음을 보호하는 정 도 입 니 다 . 사람과 다른 제품 및 시스템이 적절한 데이터에 액세스할 수 있도록 정보를 제공합니다.
- *유지보수성*: 제품 또는 시스템을 개선, 수정 및 적응하기 위해 수정할 수 있는 효과성과 효율성입니다. 환경 및 요구 사항의 변화
- *사용성*: 특정 사용자가 제품 또는 시스템을 사용하여 규정된 목표를 효과, 효율성 및 만족도로 달성할 수 있는 정도입니다.
- *성능 효율성*: 명시된 조건에서 사용된 리소스 수 대비 성능입니다.

표 1은 방법론에 매핑된 테스트 활동에 대한 개요를 제공합니다,

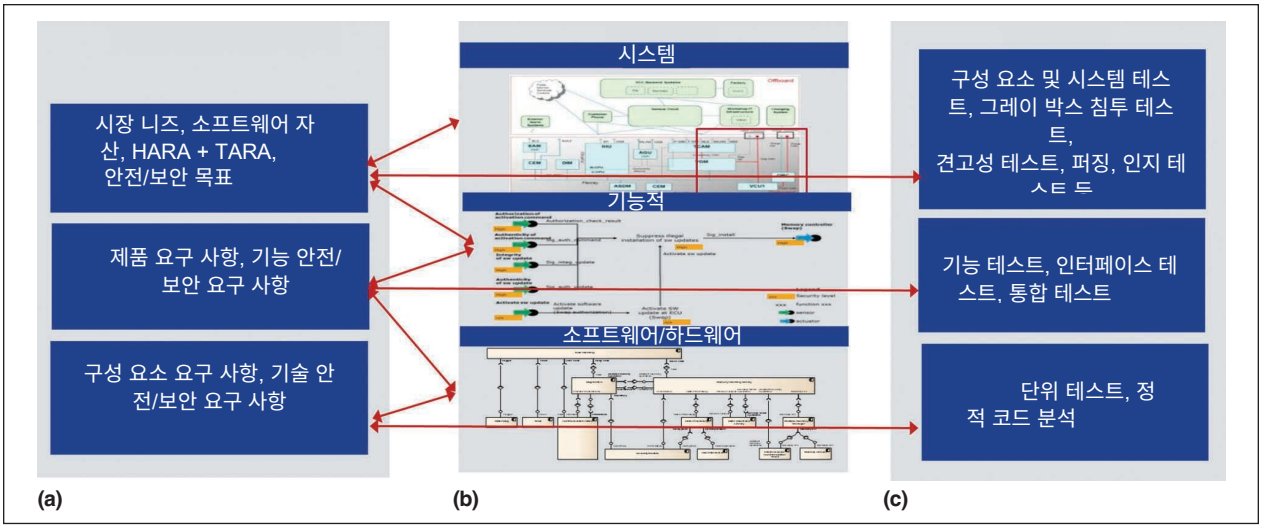

그림 1. 트리플 피크 모델(수직 계층)과 다양한 추상화(수평 계층). (a) 요구 사항, (b) 설계, 그리고 (c) 테스트. HARA: 위험 평가 및 위험 분석, TARA: 위협 평가 및 개선 분석.

표 1. 테스트 활동과 방법, 도구 및 사용 체계.

활동	테스트 방법론	결함 유형	사각지대	도구	노력
정적 분석	• 소프트웨어 유닛의 개발 초기 단계에서 결함을 감지하는 방법 • 소프트웨어 유닛에 대한 설계 검사 • 최종 사용자 기능 검토(예: 사용 사례) • 기능 요구 사항 검증 • 데이터베이스, 인터페이스, 입력/출력, 하드웨어, 소프트웨어 및 네트워크에 대한 설명 검토 • 기술: 코드 리뷰, 워크스루, 검사 및 흐름 분석	• 설계 결함 • 불충분한 기능 요구 사항 • 일관성 없는 설계 및 아키텍처 사양 • 메모리 및 버퍼 오버플로 • 코드 복잡성 • 코드, 구문 및 표준화된 라벨링 오류	• 코드 실행의 동적 동작 • 프로세서 사용률 • 보고된 결함의 수로 인해 대부분의 엔지니어는 경고를 끄게 되고, 이로 인해 대규모 사각지대가 발생합니다.	Lint, Soot, CheckStyle, SourceMeter, KlocWork, Parasoft C/C++(CERT C 규칙 지원 포함), CodeSonar C/C++, Imagix 4D, Testwell CMT++ CTC++, VectorCast	중간, 약간 자동화됨
단위 테스트	• 소프트웨어 유닛을 테스트하는 방법 • 화이트박스 테스트 • TDD 관찰 • 기능(소프트웨어) 요구 사항 "입력 X가 Y를 생성한다"를 검증하기 위한 테스트 • 기법: 고분산 클래스에 대한 동등성 클래스 분석, 경계값 분석, 코드 커버리지 테스트(문, 분기 및 수정된 조건/결정 커버리지)	• 코드 베이스의 잠재적 버그 • 요구 사항에 비해 누락된 기능 • 버퍼 오버플로 • 예외 제어 및 프로세스 활용	• 통합 오류 • 시스템 오류 • 여러 소프트웨어 장치에 걸친 기능 • 소프트웨어 단위 간의 내재적 종속성 • 품질 요구 사항 • 불명확한 보험 적용 범위	Junit, PHPUnit, NUnit, UnitTest ++, pytest, , 테스트웰 CMT++ 및 CTC++, 칸타타, ADATest 95, Mockito, TestNG, VectorCast, CANoe4SW	낮은 비용, 고도의 자동화
기능 테스트(소프트웨어)	• 시스템 요구 사항을 기반으로 SW 기능을 테스트하는 방법 • SW 애플리케이션의 블랙박스 테스트 • 정의, 예상, 기록된 입력 및 출력 값 분석 • 기능 및 비즈니스 요구 사항에 기반한 테스트 케이스 정의 • 기법: 연기 테스트, 정신 테스트, 인터페이스 테스트, 승인 테스트	• SW 애플리케이션의 주요 기능 • 사용자 인터페이스, API, I/O 및 SW 데이터베이스의 기능 • 독점 애플리케이션별 SW 유닛의 작동 제한 사항 • 여러 SW 장치에 걸쳐 있는 기능	• 품질 요구 사항 • 하드웨어 결함을 캡처할 수 없습니다. • 코드 기반 취약점을 식별할 수 없습니다. • AI 및 적응형 시스템은 적용되지 않습니다.	TestComplete, UFT, Parasoft Suite, Selenium, QTP, SoapUI, Watir, Wireshark, Tcpdump, LoadRunner, JMeter, VectorCast, VT System	중간, 고도로 자동화됨

(계속)

표 1. 테스트 활동과 방법, 도구 및 사용 체계. (계속)

활동	테스트 방법론	결함 유형	사각지대	도구	노력
기능 테스트(구성 요소)	• 시스템 요구 사항에 따라 소프트웨어 구성 요소를 테스트하는 방법 • 소프트웨어 애플리케이션의 블랙박스 테스트 • 전체 시스템의 일부 기능을 테스트합니다. • 테스트 요구 사항 및 사용 사례에 대한 유효성 검사 • 정적 분석 및 단위 테스트와 유사	• 함수(스텁) 및 호출 함수(드라이버)의 오류 • 설계 결함 • 소프트웨어 구성 요소 작업의 오류 • 테스트 계획 및 사양의 버그 및 부정확성	• 품질 요구 사항 • 설계 조건 • 디자인 기본 설정 • 테스트 대상 구성 요소의 구조적 한계	CANalyzer, CANoe, vTESTstudio, VectorCAST, JUnit, PHPUnit, NUnit, UnitTest ++, Pytest Selenium, QTP, SoapUI, DSO, 소프트웨어 정의 무선 통신, Locust, NeoLoad	중간, 고도로 자동화된
통합 테스트	• 소프트웨어 애플리케이션을 테스트하는 방법(예: 여러 소프트웨어 유닛의 조합) • 접근합니다: 화이트 박스, 블랙 박스, 그레이 박스 • 기법: 빅뱅, 증분, 하향식 및 상향식 • 중요 모듈의 우선 테스트(하향식) • 통합 테스트는 요구 사항 변경에 대한 소프트웨어 애플리케이션의 평가를 지원합니다.	• 서로 다른 소프트웨어 유닛의 통합 오류 • 여러 소프트웨어 장치에 걸친 기능 • 소프트웨어 단위 간의 내재적 종속성 • 외부 인터페이스 및 입력/출력 포트의 오류 • 설계 결함 식별 가능 • 품질 요구 사항	• 특정 결함의 원인 • 코드 기반 취약점 • 독점 애플리케이션별 소프트웨어 유닛의 작동 제한 사항 • 소프트웨어 유닛 수에 따라 인터페이스 수가 증가함에 따라 인터페이스 링크가 누락될 수 있습니다.	칸타타, LDRA 티브룬, LDRA유닛, 시트러스, 각도기, 쇼단, 핏네세, 햄크레스트	고도의 자동화
소프트웨어를 사용한 임베디드 테스트 루프 내 모델, 루프 내 하드웨어	• 시스템 및 하드웨어 시뮬레이션을 통해 소프트웨어 기능에 대한 루프 내 소프트웨어 • 모델 기반 환경에서 소프트웨어 테스트를 위한 모델 인 더 루프 • 실제(임베디드) 구성 요소를 포함하여 루프에 하드웨어 포함 • 입력 및 출력의 예상 및 기록된 동작을 검사하여 다양한 테스트 시나리오를 검증합니다. • 소프트웨어 요구 사항 검증 • 하드웨어를 루프에 포함하면 실시간 시뮬레이션이 가능합니다.	• 소프트웨어 애플리케이션의 기능 오류 • 암시적 코드 커버리지 테스트의 코딩 오류(소프트웨어 인 더 루프 테스트의 종료 기준) • 테스트 중인 하드웨어/장치의 기능 오류는 "인더루프" 시뮬레이션으로 시뮬레이션됩니다. • 루프 내 모델: 모델링된 하드웨어 및 소프트웨어의 버그 및 부정확성 • 루프 내 하드웨어: 인터페이스, 교환 메시지 및 네트워크의 결함 • 소프트웨어와 하드웨어 간의 종속성	• 특정 결함의 원인 • 코드 기반 취약점 • 독점 애플리케이션별 소프트웨어/하드웨어 유닛의 작동 제한 사항	루프 내 소프트웨어: CANoe, MATLAB, SimuLink, VSim, RT-LAB, vVIRTUALtarget, IoTIFY, IBM 블루믹스(현 IBM 클라우드), 테스트뷰, 스코프뷰. 루프 내 모델: PREEvision, MATLAB, SimuLink, vVIRTUALtarget, VISUALCONNX, MICROGen, LOOPY, PLECS, Dymola, ASCET, Scilab 하드웨어 루프: VT System, PixHawk, vTESTstudio, DSpace, CANoe, NI Lab View, VeriStand, PXI, CompactRIO, LabVIEW, TestStand	고도의 자동화

(계속)

표 1. 테스트 활동과 방법, 도구 및 사용 체계. (계속)

활동	테스트 방법론	결함 유형	사각지대	도구	노력
시스템 테스트	• 소프트웨어 제품을 검증하는 방법(예: 단일 또는 여러 애플리케이션의 통합) • 시스템 요구 사항에 대한 검증 • 블랙박스 테스트 • 소프트웨어 애플리케이션의 구성 요소에 대한 엔드투엔드 테스트 시나리오 도출 • 전용 볼륨 테스트(예: 스트레스, 성능, 부하) • 가용성, 접근성, 사용성 등과 같은 품질 요구 사항을 테스트합니다.	• 다양한 소프트웨어 애플리케이션에서 발생하는 시스템 오류 • 소프트웨어 제품의 기능적 결함 • 통신 인터페이스, 교환 메시지 및 네트워크 기능의 결함 • 입력/출력 인터페이스의 결함 • 여러 소프트웨어 애플리케이션에 걸친 기능 • 소프트웨어 애플리케이션 간의 내재적 종속성 • 설계 결함 식별 가능	• 특정 결함의 원인 • 코드 기반 취약점 • 독점 애플리케이션별 소프트웨어 애플리케이션의 작동 제한 사항 • AI 및 적응형 시스템은 적용되지 않습니다.	품질 센터/ALM, SpiraTest, TestMonitor, 가지, vFlash, vMDM, vConnect, 우편배달부, 셀레늄, 와이어샤크, 소프유아이, 테스팅위즈, 테스트완료, 라노렉스	매우 높음, 중간 정도 자동화
보안 테스트	• 시스템 및 보안 목표를 검증하는 기술 • 신호 및 프레임 퍼징을 통한 전자 제어 장치 버스 퍼징 • 보안 기술 검증 및 견고성 테스트 • 화이트 박스, 블랙 박스, 그레이 박스(인터페이스 검색, 네트워크 검색, 네트워크 침투 테스트, 소프트웨어 침투 테스트) • 유형: 신호 및 프레임 퍼징 • 조합법: 순차, 쌍, 조합, 무한 조합	• SW 및 시스템 설계의 취약성(예: 무결성, 기밀성) • 통신 기반 취약점 • 인터페이스 기반 취약점 디버그 • 의도하지 않은 행동(신원 스푸핑, 데이터 변조, 거부 위협, 정보 공개, 서비스 거부, 권한 상승) • 취약점 익스플로잇으로 이어질 수 있는 숨겨진 경로 조사	• 요구 사항 결핍을 캡처할 수 없습니다. • 하드웨어 결함을 캡처할 수 없습니다. • 코드 기반 취약점은 식별할 수 없습니다. • 무차별 퍼징 및 성능 제한으로 인해 결함 및 취약점을 놓치게 됩니다.	VectorCAST, CANoe, vTESTStudio, 칼리 리눅스, 하드웨어/소프트웨어 디버거, 컨트롤러 영역 네트워크/로컬 인터커넥트 네트워크/이더넷/플렉스레이 인터페이스	매우 높음, 퍼징을 위해 부분적으로만 자동화됨
서비스 테스트	• 단위 테스트와 유사하지만 추상화 수준이 다른 웹 서비스 및 서비스 지향 아키텍처를 검증합니다. • 웹서비스 구현 접근 방식: 단순 객체 액세스 프로토콜, 표현 상태 전송 및 웹 서비스 설명 언어 • 데이터 통신을 위해 웹 서비스를 사용하는 소프트웨어 애플리케이션	• 웹서비스 생성에 사용된 프로토콜에 결함이 있습니다. • 누락 및 잘못된 매개변수 • 규정 주소 불일치	• 웹서비스 가용성 오류 • 인터페이스 오류 • 클라이언트 애플리케이션에 대한 기존 및 신규 웹서비스 인터페이스의 불변성	SoapUI, Axis2 API, SOAPSonar, SOAtest, TestMaker, Postman, Httpmaster, vREST	높고, 잘 자동화 될 수 있습니다.

(계속)

표 1. 테스트 활동과 방법, 도구 및 사용 체계. (계속)

활동	테스트 방법론	결함 유형	사각지대	도구	노력
클라우드 테스트	• 클라우드 서비스를 사용하여 소프트웨어 애플리케이션, 소프트웨어 장치 및 시스템을 테스트합니다. • 서비스형 소프트웨어 테스트에는 서비스형 테스트가 수반됩니다. • 클라우드 도구를 통한 품질 보증 테스트 • 클라우드 도구, 아키텍처, 인프라, 플랫폼 등의 테스트 • 클라우드 서비스 및 연결성 테스트 • 성능 • 사이버 보안 • 견고성	• 다양한 소프트웨어 애플리케이션에서 발생하는 시스템 오류 • 소프트웨어 제품의 기능적 결함 • 통신 인터페이스, 교환 메시지 및 네트워크 기능의 결함 • 입력/출력 인터페이스의 결함 • 구성 관리를 통해 데이터 손실 및 데이터 복구 방지	• 클라우드 모델 간 동기화 오류 • 보안 및 개인 정보 보호 문제 무선 및 클라우드 구현 • 테스트에 영향을 미치는 서버, 스토리지 및 네트워크 장애 • 연결성 및 가용성 문제	SOASTA CloudTest, LoadStorm, BlazeMeter, Amazon Web Services, 젠킨스, 자마린 테스트 클라우드, 네서스, 개팅, 아파치 JMeter, 셀레늄	서비스 액세스 및 성능 테스트를 위한 높은 자동화 수준
CT	• 인지 테스트 방법을 통한 시스템 검증 • 블랙박스 테스트 • 방법: AI/ML 알고리즘, 수학적 모델 • 학습을 사용하여 테스트 사례 생성에서 중복 제거(무차별 대입) • 발견되지 않은 시나리오를 설명하는 소프트웨어 애플리케이션 • 코너, 오용 및 남용 사례에 대한 회계 소프트웨어 애플리케이션 • 시나리오 기반 테스트 • X-in-the-loop 테스트 스위트와 함께 사용 가능	• 다양한 장면, 상황 및 시나리오에서 소프트웨어 애플리케이션의 기능적 결함 • 기능, 통합 및 시스템 테스트에서 발견된 결함을 보완합니다. • X-인더루프 테스트 스위트와 함께 접근 방식을 사용하는 경우 실시간 결함 감지	• 품질 요구 사항 • 소프트웨어 결함 위치 파악 어려움 • 하드웨어 결함 위치 파악 어려움	요구사항 엔지니어링 데이터베이스(예: DOORS, TensorFlow, Apollo Baidu, Autoware, Open pilot, MATLAB, LGSVL, Vires, dSPACE ASM) 툴체인, 칼라, 카심, 에어심	고도의 자동화
자격 테스트	• 관련 이해관계자, 기능 및 비즈니스 요구사항에 따라 제품을 테스트합니다. • 요구 사항에 대한 소프트웨어의 충실도를 보장하는 결과 중심 방식 • 자격 평가의 합격/불합격 기준 • 품질 요구 사항도 포함됩니다. • 자격 검증을 위한 실패 모델 및 효과 분석 • 유사성, 비교, 목표 인증, 수명 주기 예측 등 계약상 자격 수준과 비교하여 소프트웨어를 테스트합니다.	• 요구 사항 체크리스트 대비 소프트웨어의 성능 기능 • 적격 소프트웨어/하드웨어에 장애를 일으킬 수 있는 잠재적 결함 • 스트레스 테스트에 대한 소프트웨어의 취약성	• 설계 결함 • 설계 조건 • 디자인 기본 설정 • 구조적 한계 • 특정 결함의 원인 • 코드 기반 취약점 • 독점 애플리케이션별 소프트웨어 애플리케이션의 작동 제한 사항	요구사항 엔지니어링 데이터베이스(예: DOORS, PREEvision, PCAN View, PCAN 탐색기, CS+, S32 Studio 및 LT Spice)	높음, 자동화되지 않음

일반적인 결함 유형, 위험, 도구 및 상대적 노력. 이 구조는 현재 표준화 및 업계 관행을 기반으로 합니다.[2,5,8] 30년 동안 테스트 및 컨설팅 분야에서 일한 경험을 바탕으로 항목을 보강했습니다.

> 사이버 범죄와 공격은 전 세계 국가와 영리 조직에 의해 이루어지고 있으며, 그 규모는 전 세계 마약 밀매보다 더 큽니다.

새로운 테스트 기술

테스트는 연구와 실무 모두에서 변화의 시점에 있습니다. 상태 폭발, 신뢰성 및 안전, IT 보안과 관련된 문제가 있는 자율 시스템에는 기존의 방법으로는 충분하지 않습니다. 의도된 기능의 안전성(또는 SOTIF라고도 함)은 안전이 중요한 자율 시스템의 승인을 검토합니다. 회귀 검증은 제어 알고리즘을 변경한 후 새로운 검사를 수행하여 소프트웨어를 자주 업데이트하는 참조된 CI/CD 파이프라인에서 기능이 더 중요해졌는지 확인하는 테스트입니다.

우리는 소프트웨어 정의 시스템으로 전환되는 미래 시나리오에 직면하게 될 것입니다. 최신 소프트웨어 업그레이드를 포함하지 않는다면 전체 인프라를 시작하지 말아야 할 수도 있으며, 그 자체도 철저한 테스트를 거쳐야 합니다. 자동차, 의료 기기, 항공 우주 차량, 제조 시설과 같이 안전에 중요한 시스템은 점점 더 소프트 웨어에 의해 정의되기 때문에 이 범주에 속합니다. 의료 기기, 로봇, 기타 자율 시스템과 같이 인간에게 직접적인 영향을 미치는 디바이스는 더욱 까다롭습니다. 계층적 소프트웨어 보증과 신뢰성을 입증할 수 있는 수단을 제공해야 합니다.

효율성과 효과성을 위해 AI 및 ML 기술을 통한 자동화를 V&V에 도입하려는 요구가 증가하고 있습니다. 여전히 관련성은 있지만, 전통적인 검증 방법으로는 점점 더 복잡해지는 AI 기반 시스템을 완벽하게 테스트하기에는 역부족입니다. 지능형 검증 기술은 전체 테스트 또는 특정 측면을 자동화합니다.[1,3,5] AI와 ML 알고리즘이 학습 제어 및 기술 시스템에 활용되면 소프트웨어 검증은 훨씬 더 복잡해집니다.

인지 테스트 - 방법

1) 학습
무한 상태 공간에서 규칙, 휴리스틱 등을 도출하고, 온톨로지를 구축 및 유지 관리하며, 시나리오에 클러스터링하고 테스트를 통해 학습하여 개선합니다.

2) 모델링
심각도 및 빈도 확률을 기반으로 선택한 각 시나리오에 대한 적절한 테스트 사례의 위험 기반 개발

3) 선택
테스트 전략에 따라 적절한 시나리오 및 테스트 사례 식별하기,
예: 회귀 범위 및 시스템과 구성 요소 비교

4) 테스트
실행 가능한 최소 시나리오 및 테스트 케이스 세트를 실행하여 시스템 동작을 확인하고, KPI로 결과를 평가하고, 릴리스를 결정합니다.

비유: 운전 시험관

1) 학습
관련 상황을 종합적으로 테스트하는 데 필요한 일련의 규칙 식별

2) 모델링
예상되는 인지 행동을 확인하기 위한 테스트 조합 개발하기

3) 선택
특정 시나리오 및 테스트 케이스 세트를 선택하여 테스트 실행하기

4) 테스트
운전자의 인지 행동을 평가하여 운전 면허 시험 실행

그림 2. 운전 면허 시험에 비유한 인지 테스트 방법. KPI: 핵심 성과 지표.

제거합니다. 휴리스틱과 자동 코너 케이스 식별을 사용하면 다음과 같은 작업을 수행하는 데 많은 시간이 필요하지 않습니다.

저자 소개

이 유효성 검사에는 다음과 같은 유형이 포함됩니다:

- 요구 사항 기반 테스트
- 온톨로지 기반 테스트
- CT.

새로운 AI 기반 테스트의 예로, 요구 사항에서 파생된 고전적인 기능 커버리지와 휴리스틱 및 학습을 결합한 CT를 간략히 살펴보겠습니다. 그림 2는 자율주행 차량의 CT에 대한 사례 연구입니다. AI 기반 방식은 사람이 특정 시나리오를 식별하고 생각하지 못할 수 있으므로 수동 도출과 관련된 잠재적 오류를

T

우리 모두에게 중요합니다. 이는 개발 후 별도의 활동이 아니라 요구 사항 설계부터 필수적인 부분입니다. 테

무차별 대입 검증을 위한 테스트 사례를 도출하는 데 투자해야 합니다.[5]

진정으로 투명한 검증 방법과 테스트 프로세스는 최대한의 관련성을 전제로 하며, 자율 행동을 향한 기술의 단계적 발전에 따라 더욱 중요해질 것입니다. 관련성은 있지만 기존의 검증 방법으로는 점점 더 복잡해지는 자율 주행 자동차를 완벽하게 테스트하기에는 부적절합니다. 상황 적응과 소프트웨어 업데이트 및 업그레이드를 지원하는 머신러닝에는 새로운 회귀 전략이 필요합니다.

대응 기업이 점점 더 많이 이해하고 있습니다. 즉, 애플리케이션 수명 주기 관리/제품 수명 주기 관리 파이프라인 내에서 테스트 도구를 오케스트레이션하고 시스템 엔지니어링, 설계 및 관련 활동에서 V&V 프로세스를 포지셔닝하는 등 V&V 접근 방식에 대해 지속적으로 학습해야 따라 달라집니다. 모 든 조직은 방법론과 개발 환경을 맞춤화해야 합니다. 기술은 종종 도구와 연관되지만, 소프트웨어 조직은 먼저 필수적인 V&V 역량을 구축해야 합니다. 복잡한 툴 체인은 있지만 실질적인 테스트 전략과 체계적인 프로세스가 없는 경우가 너무 많습니다. 다음 질문에 답해야 합니다 :

- 시간 외에 릴리스 기준 테스트 사례를 어떻게 찾아내나요?
- 중요한 코너와 기능의 상관관계를 체계적으로 테스트하는 방법은 무엇인가요?
- 소프트웨어의 품질을 어떻게 측정하고 개선하나요?

인적 위험을 줄이려면 소프트웨어 정의 시스템이 점점 더 많은

크리스토프 에버트는 벡터 컨설팅의 전무이사입니다. 서비스, 슈투트가르트, 70499, 독일. https://twitter 으로 문의하세요. com/christofebert 또는 christof.ebert@vector.com.

디비스 바자즈는 로보 테스트 및 벡터 컨설팅 서비스에서 근무하고 있습니다,

마이클 웨이리치(Michael Weyrich)는 산업연구소의 소장입니다. 자동화 및 소프트웨어 공학, 슈투트가르트 대학교, 슈투트가르트, 70565, 독일. michael.weyrich@ias로 문의하세요. uni-stuttgart.de.

자체 결함 및 장애 지점을 자동으로 감지할 수 있어야 합니다. 테스트에는 올바른 도구 체인을 갖춘 숙련된 실무자가 필요합니다. 사냥꾼에 대한 일화에서 알 수 있듯이 도구는 필요하지만 충분하지는 않습니다. 어느 날 한 사냥꾼이 활과 화살, 총과 같은 첨단 기술을 가지고 도착했을 때 새끼 호랑이는 매우 무서워했습니다. 하지만 호랑이는 새끼 호랑이에게 몰래 숨어 공격하는 방법을 가르쳐 주었습니다. 그 사냥꾼은 다시는 소식을 들을 수 없었는데, 이는 도구는 좋지만 올바른 과정을 결코 대체할 수 없다는 것을 증명합니다.

참조

1. "세계 품질 보고서(WQR) 2021-22", Capgemini, 프랑스 파리, 2021. [온라인]. 이용 가능: https://www.capgemini.com/re- 검색/세계-품질-보고서 -wqr-2021-22/
2. D. Graham *외.: 소프트웨어 테스트의 기초: ISTQB 인증*, 4권. 영국 앤도버: Cengage Learning, 2019.
3. S. Goericke, *소프트웨어 품질 보증의 미래*. 하이델베르크, 독일: SpringerOpen, 2020.
4. *소프트웨어 수명 주기 프로세스 표준*, ISO 12207-2017, 국가 간 표 준화 기구, 스위스 제네바, 2017. [온라인]. 사용 가능: https://www. iso.org
5. *시스템 및 소프트웨어 엔지니어, 시스템 및 소프트웨어 품 질 재요구 사항 및 평가 (SQuaRE), 품질 요구 사항 프레임워크*, ISO/IEC 25030-2019, 국제 표준화 기구, 제네바, 스위스, 2019. [온라인]. 사용 가능: https:// www.iso.org
6. *소프트웨어 테스트*, 국제 표준화 기구, 스위스 제네바, ISO/IEC/IEEE 29119, 2013.
7. C. Ebert 및 R. Ray, "테스트 중심 요구 사항 엔지니어링," *IEEE Softw.*, 38권 1호, 16-24쪽, 2021년 1월, 도이치: 10.1109/MS.2020.3029811.
8. C. Ebert와 M. Weyrich, "자율 시스템의 밸리데이션", *IEEE 소프트웨어*, 36권, no. 5, 15-23쪽, 2019년 9월, 도이: 10.1109/MS.2019.2921037.

디지털 객체 식별자 10.1109/MS.2022.3177874