백서

임베디드 소프트웨어의 안전 및 보안을 향상하는 7가지 방법

정적 분석 및 정형 기법

컴퓨터공학 분야의 선구자 Edsger Dijkstra는 "프로그램 테스트는 버그의 존재를 보이는 데 사용될 수 있지만, 버그의 부재를 보이는 데는 사용될 수 없다!"고 한 적이 있습니다. 하지만 수많은 임베디드 프로젝트에서 테스트 실패가 없다고 해서 품질이 증명되었다고 잘못 해석하는 경우가 발생하곤 합니다. 본 백서에서는 이 문제를 해결할 고유한 방법을 제시합니다.

개발 팀들은 **정형 기법**을 적용하는 최신 정적 분석 툴을 사용하여 시스템을 통합하거나 하드웨어에서 테스트하거나 코드를 실행하지 않고도 ***버그의 부재를*** *증명할 수 있습니다.*

Polyspace® 정적 분석 소프트웨어는 다음과 같은 두 가지 제품에서 정형 기법 기술을 제공합니다.

- *Polyspace Bug Finder*™는 소스 코드에서 가능한 런타임 오류 및 기타 수백 개의 버그 클래스를 식별하고, 코딩 규칙 및 보안 표준의 준수 여부를 검사하고, 코드 메트릭 보고서를 생성합니다.
- *Polyspace Code Prover*™는 테스트 케이스나 코드 실행 없이도 코드 단위와 통합 코드에 심각한 런타임 오류가 없음을 증명합니다. Polyspace Code Prover는 정형 기법(추상적 해석)을 사용하여 거짓 음성 없이 모든 가능한 입력과 실행 경로, 변수 값을 고려합니다.

Polyspace는 ISO 26262, IEC 61508, IEC 62304와 같은 기능적인 안전 표준을 준수하는 데 필요한 개발 프로세스에서 사용하는 용도로 TÜV SÜD로부터 인증을 취득했으며, DO-178B/C를 준수합니다.

Polyspace 정적 분석 및 정형 기법을 사용하여 임베디드 소프트웨어의 안전 및 보안을 향상하는 7가지 방법

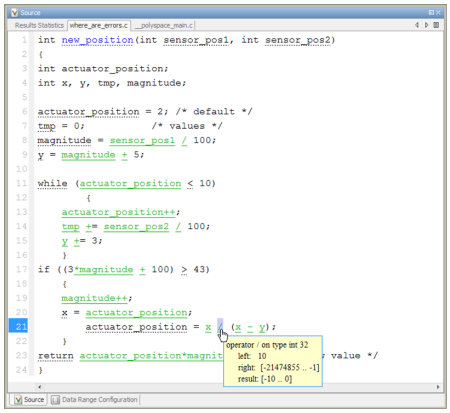
Polyspace를 사용하는 Nissan, Airbus, Delphi를 비롯한 기업의 개발 팀들은 7가지 장점이 있다고 전합니다.

1. **개발자를 위한 즉각적인 피드백.** Polyspace는 *코드의 모든 포인트에 대한 런타임 변수 범위 정보,* 잠재적인 오버플로/언더플로 버그 조건, 데드 코드와 같은 세부 정보를 제공하며 MISRA®와 같은 코딩 지침을 적용하는 데 도움이 됩니다.
2. **집중적인 단위 테스트 전략.** Polyspace는 ***모든 가능한 입력에*** *심각한 결함이 없음*을 증명함으로써 단위 테스트 개발 노력을 줄여 주고 가이드를 제공합니다.
3. **동시 발생 결함 검출.** Polyspace는 멀티스레드 응용 프로그램에서 *희귀 조건이 없음을 증명*하고 소스에서 결함을 추적합니다.
4. **문서화된 흐름 정보.** Polyspace는 철저하고 견고한 의미론적 분석을 통해 *상세한 제어 및 데이터 흐름 정보*를 제공합니다.
5. **보안 준수.** Polyspace는 버그 찾기, 코드 증명 및 표준 검사를 사용하여 *보안 취약성을 식별 및 방지*하고 CERT C, ISO 17961, CWE와 같은 표준을 준수합니다. 버퍼 오버플로, 부적절한 포인터 역참조, 초기화되지 않은 변수와 같이 해커들이 악용할 수 있는 *보안 취약성이 없음을* ***증명***할 수 있습니다.
6. **인증 표준을 위한 아티팩트.** Polyspace는 ISO 26262, IEC 61508, IEC 62304와 같은 기능적인 안전 표준을 준수하는 데 필요한 개발 프로세스에서 사용하는 용도로 TÜV SÜD로부터 인증을 취득했으며, DO-178B/C를 준수합니다. 다른 툴로는 얻을 수 없는 정형 기법, 제어/데이터 흐름, 범위 검사용 인증 크레딧을 얻을 수 있습니다.
7. **모델 기반 설계와의 통합.** Polyspace는 모델 기반 설계 툴체인의 일부로 기능하며, *Simulink® 및 Stateflow® 모델로의 추적 기능을 제공합니다.*

테스트와 정형 기법: 간단한 비교

```
1  int new_position(int sensor_pos1, int sensor_pos
2  {
3  int actuator_position;
4  int x, y, tmp, magnitude;
5
6  actuator_position = 2; /* default */
7  tmp = 0;              /* values */
8  magnitude = sensor_pos1 / 100;
9  y = magnitude + 5;
10
11 while (actuator_position < 10)
12     {
13     actuator_position++;
14     tmp += sensor_pos2 / 100;
15     y += 3;
16     }
17 if ((3*magnitude + 100) > 43)
18     {
19     magnitude++;
20     x = actuator_position;
21     actuator_position = x / (x - y);
22     }
23 return actuator_position*magnitude + tmp; /* new
24 }
25
```

예제 소스 코드.



Polyspace 분석 결과.

입력값을 두 개 갖는 함수를 살펴보겠습니다. 이 짧은 코드에 단 하나의 결함도 없도록 하려면 수동으로 어떻게 검토하시겠습니까?

테스트 방식. 테스트를 고려 중이라면, MCDC(수정 조건/결정 실행률)를 위한 두 개의 테스트 케이스만 필요합니다. 하지만 강인성을 보장하는 데 이것만으로 충분할까요? *철저한 테스트라면* 모든 입력값의 가능한 모든 값을 고려해야 할 것입니다.

정적 분석. 정적 코드 분석은 테스트 케이스를 보완하는 용도로 널리 사용됩니다. 코딩 표준 및 스타일 가이드 준수와 같은 수동 검증 작업을 자동화해 주는 정적 분석은 휴리스틱을 기반으로 결함을 검출할 수도 있습니다.

이 예제에서는 정적 분석 툴을 사용하면 변수의 여러 가능한 값을 검사하여 테스트 실행률을 늘릴 수 있습니다. 하지만 어떤 경우에도 발생하지 않을 값에 대한 거짓 경고가 많이 발생한다는 부수적인 효과를 감수해야 합니다.

정형 기법. Polyspace 정적 분석은 기본적인 기법을 포함할 뿐 아니라 정형 기법을 사용하여 런타임 동작을 확인하고 오류가 없음을 증명합니다.

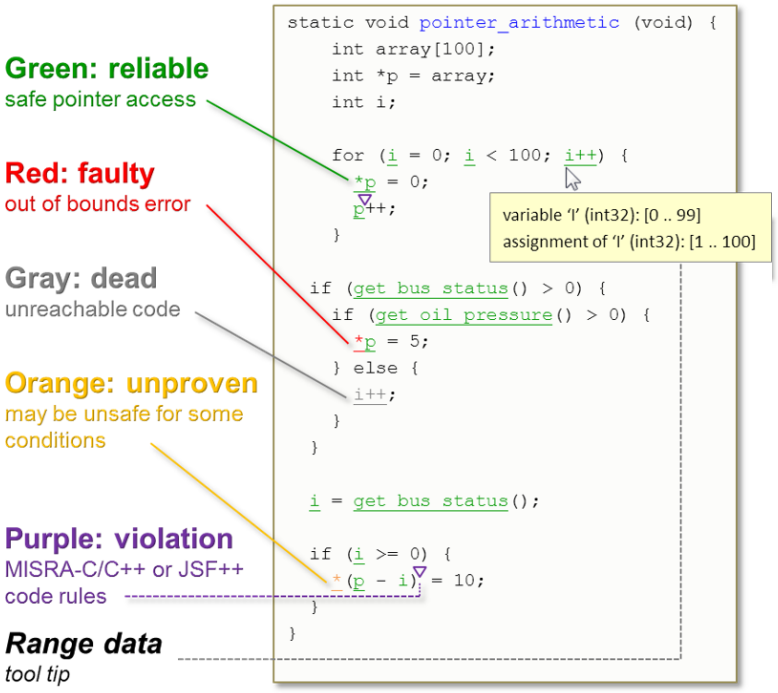
Polyspace는 추상적 해석과 같은 증명 기반 기법을 사용하여 소프트웨어가 모든 런타임 조건으로 안전함을 증명합니다. 그뿐만 아니라 코드의 모든 포인트에 대한 변수 범위 정보를 식별하고, 제어 및 데이터 흐름을 매핑합니다.

예제의 해. 예제 코드에서, Polyspace는 코드에 심각한 런타임 오류가 없음을 증명했습니다. 0에 의한 나눗셈이 의심되는 21행을 살펴보겠습니다. Polyspace는 정형 기법을 사용하여 변수 x와 y의 범위가 어떤 경우에도 동일하지 않음을 확인했으며, 따라서 0에 의한 나눗셈은 일어나지 않음을 증명했습니다.

자세히 살펴보기: 임베디드 코드의 안전 및 보안을 향상하는 7가지 방법

1. 개발자를 위한 즉각적인 피드백.

임베디드 시스템에서는 코딩 단계에서 런타임 결함이 다수 발생합니다. 이러한 오류는 코드 검토 단계에서 간과되는 경우가 많으며, 일반적인 정적 분석으로는 검출할 수 없습니다. 오류가 개발 프로세스의 다운스트림으로 전파되어 나중에 하드웨어 테스트 단계에서 발견되면 재작업이 필요하게 되며 지연이 발생하게 됩니다.



런타임 오류 특성은 색으로 구분됩니다.

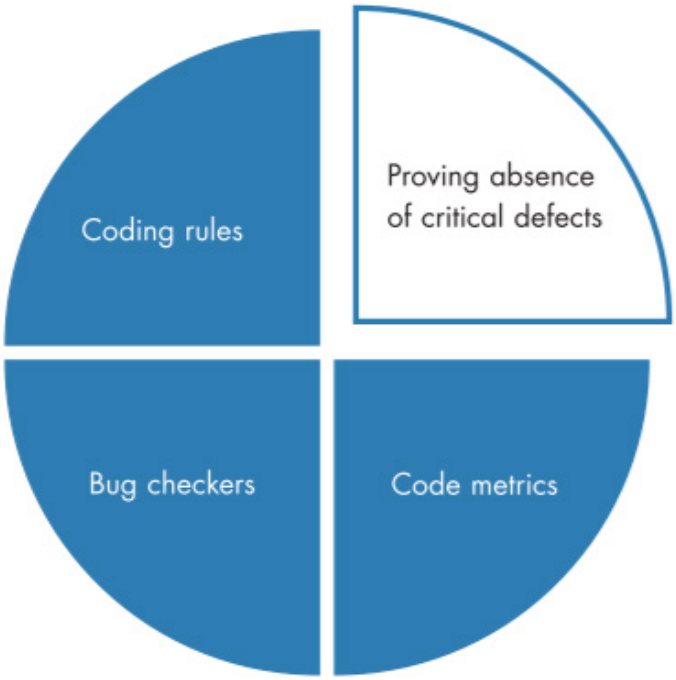
Polyspace는 ***코드의 모든 포인트에 대한 변수 범위 정보***, 잠재적인 오버플로/언더플로 버그 조건, 데드 코드와 같은 런타임 세부 정보를 포함하는 정적 분석을 제공함으로써 이 문제를 해결하며, MISRA와 같은 코딩 지침을 적용하는 데 도움이 됩니다. 개발자는 정적 분석 및 정형 기법을 적용함으로써 런타임 문제가 다음 단계로 전파되기 전에 코딩 중에 발견하고 수정할 수 있습니다.

> *"동적 테스트는 단순히 오류의 증상을 감지할 수만 있습니다. Polyspace 코드 검증은 근본 원인을 찾아내므로 우리는 디버그에 큰 노력을 할 필요가 없습니다."*
>
> —**프레데릭 리테유**(Frédéric Retailleau), Delphi Diesel Systems

2. 집중적인 단위 테스트 전략.

단위 테스트를 실시하면 구성요소의 강인성을 확인할 수는 있으나, 이를 위해서는 테스트 케이스를 작성해야 하는데 테스트 케이스는 런타임 오류를 집중 조명하는 경우가 많습니다. 단위 테스트를 수동으로 작성하기란 까다로운 일이며, 실행률을 늘리기 위해 테스트 케이스를 추가하면 심각한 오류를 놓치거나 코드에서 중복된 요소가 발생할 수 있습니다.

Polyspace는 ***코드가 런타임 실패로부터 안전함을 증명***함으로써 강인성 테스트의 필요를 없애고 추가 분석이나 테스트가 필요한 검증되지 않은 연산을 식별합니다. Polyspace는 ***모든 가능한 입력에*** 심각한 결함이 없음을 증명함으로써 단위 테스트 개발 노력을 줄여 주고 가이드를 제공합니다.



"Polyspace Code Prover는 손으로 작성한 코드에서 문제를 식별했습니다. 오류가 없는 코드와 주의가 필요한 코드도 식별했습니다. 이 결과를 바탕으로 정형 검사 프로세스에서 코드의 타겟팅된 평가를 수행할 수 있었습니다."

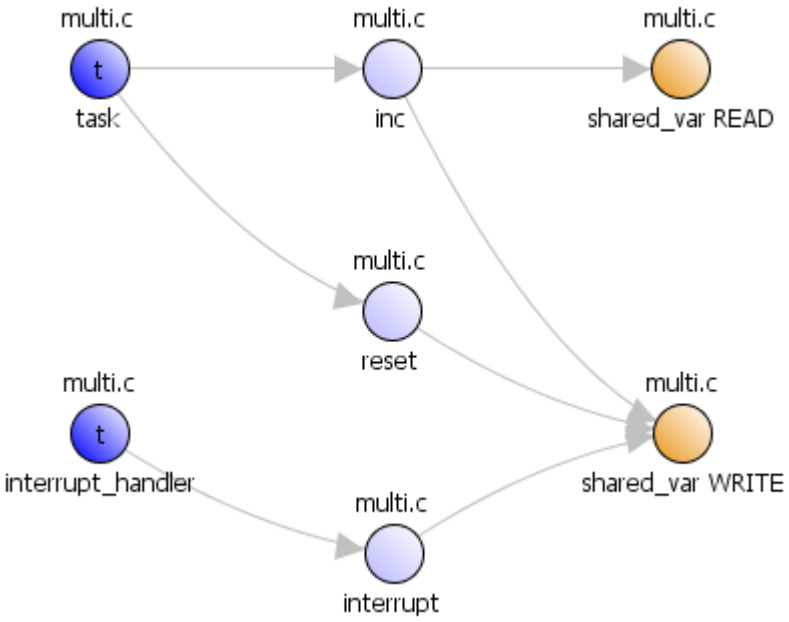
— **캐런 건디-벌렛 박사**(Dr. Karen Gundy-Burlet), NASA Ames Research Center

3. 동시 발생 결함 검출.

Polyspace는 멀티스레드 응용 프로그램에서 ***희귀 조건 및 기타 동시 발생 문제가 없음을 증명***하고 소스에서 결함을 추적합니다.

정적 메모리 문제, 동시 발생 문제, 세밀한 런타임 오류와 같은 복잡한 결함은 검출하기가 힘들며 프로덕션 단계로 그대로 넘어갈 수 있습니다. 이를 검출하기 위해서는 꼭 맞는 테스트 케이스가 필요하며, 재현하기도 어렵습니다.

Polyspace는 코드를 소스 코드 리포지토리에 체크인하기 전에 이러한 결함을 검출할 수 있도록 도움을 주므로 테스트와 숨겨진 버그의 디버그 과정이 필요하지 않게 됩니다. Polyspace는 디버그 프로세스를 순차적으로 실행하는 이벤트 추적 기능을 통해 복잡한 결함의 디버깅 노력을 줄여 줍니다.



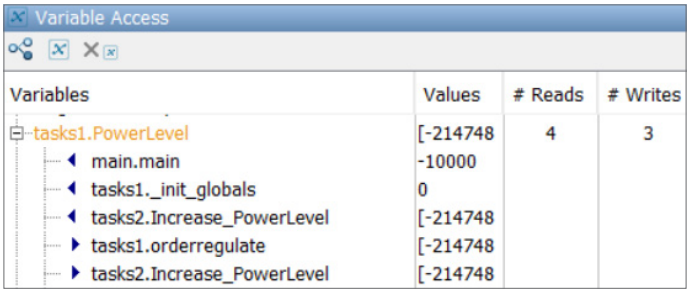
"Polyspace 제품은 어떤 경우에 런타임 오류가 발생할 수 있는지 찾고 어떤 환경에서도 오류가 절대 발생하지 않을 경우도 식별합니다. 이 작업을 단위 테스트를 수행하기 전, 코딩 중에 한다는 것이 장점입니다. 그 결과 공급업체에 엄청난 가치를 제공할 수 있게 되었습니다."

— **미츠히코 키쿠치**(Mitsuhiko Kikuchi), Nissan

4. 문서화된 흐름 정보.

Polyspace는 철저하고 견고한 의미론적 분석을 통해 ***상세한 제어 및 데이터 흐름 정보***를 제공합니다. 제어 및 데이터 흐름 정보는 코드를 설계하고 디버그할 때 무척 유용하지만, 수동적인 접근 방식으로는 추적하기가 어렵습니다.

Polyspace는 이 정보를 기능적인 호출 트리, 글로벌 데이터 사전 및 변수 데이터 범위로 제공함으로써 복잡한 런타임 에지 케이스의 디버그를 도와줍니다.



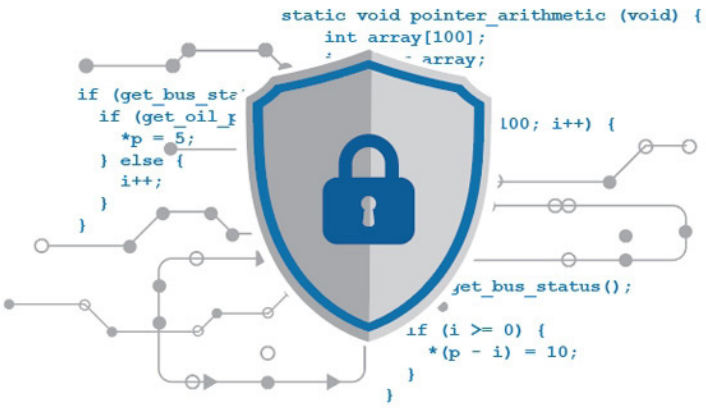
"Polyspace 솔루션은 실행 없이도 런타임 오류를 검출하며, 코드를 남김없이 철저히 확인합니다."

— Airbus

5. 보안 준수.

첨단 운전자 보조 시스템(ADAS)과 같은 임베디드 시스템이 연결되는 빈도가 늘어남에 따라 보안은 점점 더 큰 우려 사항이 되고 있으며, 안전이 중시되는 분야의 경우 특히 그렇습니다. 보안에 대한 종합적인 접근 방식은 전체 개발 사이클을 포괄하며, 따라서 추가적인 검증과 확인이 요구됩니다.

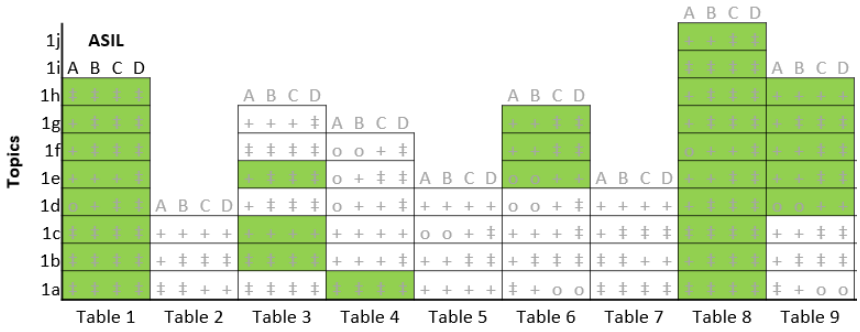
Polyspace는 버그 찾기, 코드 증명 및 표준 검사를 사용하여 *보안 취약성을 식별 및 방지*하고 CERT C, ISO 17961, CWE와 같은 표준을 준수합니다. 버퍼 오버플로, 부적절한 포인터 역참조, 초기화되지 않은 변수와 같이 해커들이 악용할 수 있는 ***보안 취약성이 없음을 증명***할 수 있습니다.



MathWorks®

6. 인증 표준을 위한 아티팩트.

Polyspace는 *ISO 26262, IEC 61508, IEC 62304와 같은 기능적인 안전 표준*을 준수하는 데 필요한 개발 프로세스에서 사용하는 용도로 TÜV SÜD로부터 인증을 취득했으며, DO-178B/C를 준수합니다. 다른 툴로는 얻을 수 없는 정형 기법, 제어/데이터 흐름, 범위 검사 등의 용도로 인증 크레딧을 얻을 수 있습니다.



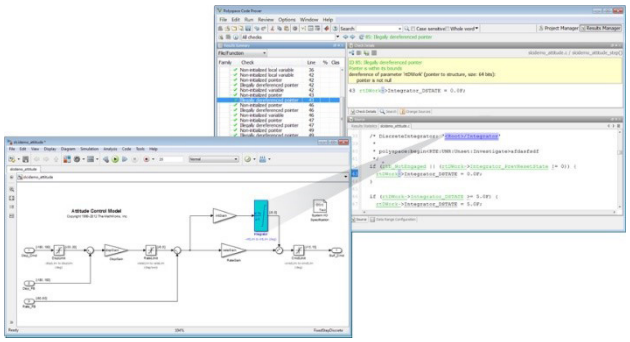
ISO 26262 크레딧

"FDA의 출시 전 승인을 받는 과정에서 Polyspace Code Prover는 코드의 적합성을 증명하고 코드 품질을 보장하기 위해 우리가 최선의 노력을 다했음을 보여주는 데 있어 없어서는 안 될 툴입니다."

— **라르스 샤이망크**(Lars Schiemanck), Miracor Medical Systems

7. 모델 기반 설계와의 통합.

Polyspace는 모델 기반 설계 툴체인의 일부로 기능하며, *Simulink 모델로의 추적 기능을 제공합니다*. Simulink 모델이나 dSPACE® TargetLink® 블록에서 생성된 코드로 정적 분석을 실행할 수 있습니다. Simulink 내부에서 분석을 실행하여 모델에서 결과를 추적할 수 있습니다.



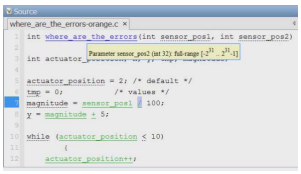
예: Alenia Aermacchi는 Polyspace를 사용하여 코드의 런타임 오류를 확인하고 MISRA C 코딩 표준의 준수 여부를 확인하였으며 인증 크레딧에 대한 아티팩트를 생성했습니다. DO-178용 DO Qualification Kit를 사용하여 Polyspace code verifier와 Simulink Verification 제품을 인증하였습니다.

MathWorks®

자세히 알아보기

Solar Impulse, 태양광 비행기에 Polyspace 정적 분석 이용

Polyspace 제품을 사용하여 문제를 조기에 빠르게 찾아서 제거한 결과, Solar Impulse는 1~2년이라는 개발 엔지니어링 시간을 단축하고 DO-178 준수라는 목표를 달성하는 데 성공했습니다.



정적 분석에 대한 잘못된 생각 바로잡기

정적 분석에 대한 잘못된 생각을 바로잡아 봅니다. 여기에는 “테스트를 충분히 하기 때문에 정적 분석은 필요하지 않다”거나 “정적 분석은 인증을 충족해야 하는 경우에만 필요하다”도 포함됩니다.

평가판 요청 | 전문가와 상담