ISO 26262 인증의 효과적 테스트 방법

BTC EmbeddedTester®를 이용한 테스트

모델 기반 백투백 테스팅 접근법은 개발 중인 제품의 품질 향상과 전체적인 테스팅에 대한 수고를 최소화해준다. 코드 수준에서의 정형 기법 기술의 재사용으로 인해 ISO 26262, IEC 61508 또는 DO-178b와 같은 산업 표준을 기준으로 설정된 품질 레벨을 충족시킬 수 있다. 진행된 테스팅에 대한 품질을 모든 면에서 평가하기 위하여 테스팅과 개발을 진행하는 동안에 테스트 실행과 coverage 측정 기술의 견고한 결합은 필수적이다.

글 | **한스 홀베르크**(Hans J. Holberg) 〈holberg@btc-es.de〉
SVP 마케팅 및 영업
BTC Embedded Systems AG

모델 주도 개발(Model Driven Development)이 더 짧은 시간, 더 적은 비용으로 더 많은 임베디드 기능을 개발할 수 있도록 한다는 사실은 그 동안 인정돼 온 사실이다. 임베디드 소프트웨어 개발에 있어서 모델 주도 방법의 추가적인 중요 장점은 MiL (Model in the loop) 시뮬레이션과 같은 개발 초기 검증 수단으로 인해 개발된 소프트웨어의 품질이 더욱 향상되었다는 점이다. 그러나 일반적으로 임베디드 소프트웨어 테스팅은 여전히 80~90년대의 소프트웨어 개발 과정에 맞는 조금 더 오래된 과정과 방법을 사용하고 있다.

사실 소프트웨어 개발에 모델을 사용하는 소프트웨어 엔지니어들의 높은 수준의 생산성과 테스팅 및 품질보증을 맡고 있는 엔지니어들의 낮은 수준의 생산성 사이에는 상당한 차이가 있다. 특히 걱정스러운 부분은 개발 시간이 촉박하여 임베디드 소프트웨어의 품질이 저하된다는 것이다. 게다가 자동차 기능에서 안전성이 점점 더 중요해지고 있기 때문에, 자동차 분야의 OEM들과 부품 공급업체들이 느끼는 압박은 더욱 커지고 있다. 이런 이유로 자동차 분야로 새롭게 다가오고 있는 ISO 26262 표준에 의해서 제기된 기능안전성이라는 관점은 소프트웨어 분야에서 기존에 존재하는 개발 과정들을 위협하고 있다.

이 글에서는 모델 기반 백투백(back-to-back) 테스트 접근법을 가지고 어떻게 모델 주도 개발을 보완하는지, 그리고 이것이 어떻게 상당히 향상된 품질과 테스팅 효율을 만들어 내는지 제시할 것이다. 이 모든 것은 기술된 테스트 기능과 검증 환경을 지닌 BTC EmbeddedTester®가 ISO 26262 표준의 모든 ASIL(Automotive Software Integrity Levels)에서 "목적에 알맞다"라고 독일 인증기관 TÜV SÜD로부터 인증 받은 것처럼, ISO 26262의 기준을 충족시키면서 진행된다. 완전한 모델 기반 소프트웨어 검증 접근법은 TargetLink를 사용한 모델 주도 개발의 맥락에서 설명된다. 이 접근법은 MiL, SiL, PiL 테스팅 작업을 매끄럽게 통합하고, 그렇게 함으로써 일상적이고 많은 테스트 활동들을 자동화할 수 있다. 심지어 이런 접근법은 어떻게 필수요소인 테스트 벡터(test vector)들의 생성이 완전히 자동화되는지, 또 자동화함으로써 어떻게 임베디드 소프트웨어의 완전한 검증이 현재보다 더 짧은 시간 내에 행해질 수 있는지 보여준다.

여기에 기술된 방법은 자동 코드 생성 환경인 dSPACE TargetLink에 기반을 두고 있다. 자동화된 테스트와 검증 환경을 지닌 BTC EmbeddedTester®는 자동 코드 생성기의 모든 모델링 블록 세트(modelling block-set)를 지원하고 있다. 추가적으로 여기서 설명하는 방법은 다른 경로로 생성된, 심지어는 손으로 직접 작성한 추가 레거시 코드(external legacy code)도 지원한다. 현재 이용 가능한 솔루션은 계층 구조로 개발된 어떤 고정소수점과 부동소수점 애플리케이션(application)이라도 지원함으로써 매우 높은 모델 및 코드 커버리지(code coverage) 수준에 다다를 수 있다. 이러한 사실은 지난 5년 동안 독일, 프랑스, 일본의 자동차 분야에서 일련의 제품을 통해 성공적으로 증명됐다.

산업 기능안전 표준의 지원

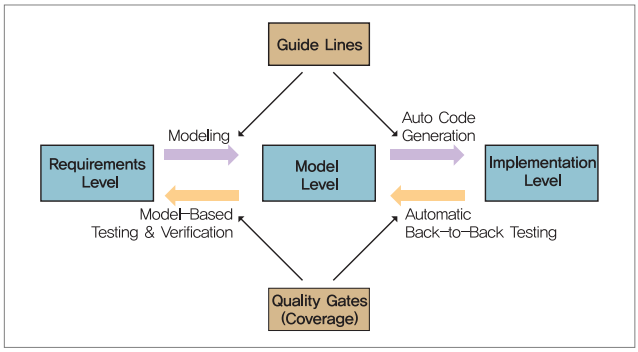
기술된 방법은 거의 자동차 분야에서 사용되기 때문에, 품질 측면을 관련된 안전기준을 사용해서 평가할 수 있다. 기술된 방법은 현재 기능안전 표준인 IEC 61508을 특별히 자동차의 기능안전에 맞춰서 각색하고 확장한 ISO 26262 표준에 포커스 돼 있다. 이 국제 표준의 최종 본 배포는 2011년으로

계획되어 있다(2009년 중반 이후 Draft International Standard 버전으로 이용 가능). ISO 26262에서는 모델 기반 개발(Model Based Development) 과정을 고려하고 있다. 때문에 모델 기반 테스팅과 각각 다른 개발 단계 사이의 백투백 테스팅은 최신의 기술이 될 것이다. ISO 26262는 ASIL A, ASIL B, ASIL C, ASIL D라고 하는 4개의 안전수준을 정의하고 있다. 수준A는 가장 낮고 D는 가장 높은 안전수준이다. 모든 레벨에서, "model과 코드 사이의 back-to-back 테스트"를 권장하고 있으며, 수준C와 D에서는 적극 권장하고 있다. 백투백 테스트의 품질은 소위 coverage criteria라고 불리는 기준에 의하여 결정된다. 특히 statement coverage, branch coverage, 그리고 MC/DC coverage는 서로 다른 ASIL 수준에서 요구된다.

효율성 측면

품질 측면에서 백투백 테스트가 최신 기술인 것과 같이, 다음으로 매우 중요한 관점은 개발과 테스팅에 대한 효율성이다. 규격에 관련된 ASIL 수준들에 따라 각각 다른 coverage criteria는 그 테스팅 활동 과정에 대해서 추가적인 많은 노력을 초래하고 있다. 이러한 테스팅 복잡성을 다루기 위해서, 그리고 이런 도전을 극복하기 위해서는 자동화 접근법이 절대적으로 필요하다.

복잡성에 대한 첫 번째 노력은 테스트 생성 단계에 있다. 모델과 코드 커버리지를 만족시키기 위해 테스트 벡터 생성을 자동화하는 방법으로 이 문제에 대응할 수 있다. 또 다른 종류의 문제점은 매우 많은 테스트들을 효율적으로 실행하고 분석하는 것이다. 테스팅 작업 흐름에 관련된 수많은 수작업을 방지하기 위해서, 그리고 수동 테스트를 실행하는 동안의 테스트 에러를 막기 위해서 완전히 자동화된 솔루션이 반드시 필요하다. 마지막으로 품질 매트릭스(커버리지 통계)는 자동으로 생성된 테스트 보고서로부터 결정되어야만 한다. 소프트웨어 생성 툴을 가진 개발과정과 테스트 환경 안에서 이런 기술들이 잘 통합되어야만 테스팅을 자동화하는 이러한 모든 방식들이 효율적으로 동작할 수 있다. 이렇게 완전히 자동화되고 통합된 솔루션은 이 글의 뒤 부분에서 소개할 것이다.



【그림 1】 Reference Work Flow

Reference work flow

모델 기반 개발과 모델 기반 테스팅 사이의 관계를 정의하기 위해서, reference work flow가 개발되었다. 이 work flow는 코드 구현 수준 상의 수동 테스팅으로부터, 모델과 코드 레벨 사이에서 자동화된 백투백 테스트와 결합된 모델 기반 테스팅으로 이동하는 패러다임 변화를 보여준다. 새로운 접근법은 주요 개발과 테스팅 작업을 모델 수준에 초점을 맞추고 있고, 자동 코드 생성에 의해서, 그리고 자동화된 구조적 백투백 테스트와 결합하여 올바른 테스트 방식으로의 변화를 보증한다. 모델 수준에서의 테스팅과 디버깅이 훨씬 더 쉽고 비용이 적게 든다는 사실은 널리 인정돼 왔다. **그림 1**은 위에 언급된 reference work flow를 간단히 보여주고 있다.

특정 비정형(informal) 또는 정형(formal) 요구 명세들을 기반으로, 실행 가능한 모델들은 특정한 가이드라인을 충족시키는 모델링에 의해서 개발될 수 있다. 이것이 모델 시뮬레이터를 사용해서 모델 기반 테스팅을 가능하게 하고, 심지어는 모델 checker를 이용해서 정형 검증(formal verification)을 가능하게 한다. 모델 기반 테스팅은 거의 ISO/DIS 26262와 같은 산업 안전표준에 의해서 잘 정의된 품질 게이트 환경에서 실행된다. 만약 이 품질 게이트를 통과한다면, 모델 수준을 구현 수준으로 전환하기 위해 자동 코드 생성 기능을 사용할 수 있다. 이런 단계들은 자동 테스트 생성, 자동 테스트 실행, 그리고 테스트 평가로 구성된 자동화된 백투백 테스트 접근법을 사용해서 검증되어야 한다.

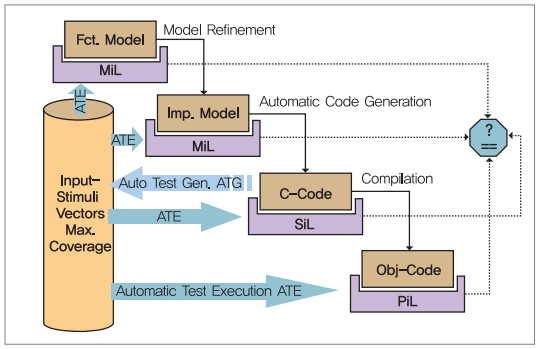
모델 기반 소프트웨어 검증 접근법

자동차나 항공 등 몇몇 산업 영역에서 개발과 테스트 과정은 갈수록 모델 기반으로 바뀌어 왔다. 이는 모든 소프트웨어와 하드웨어 디자인 결정이 미리 이루어지는 것이 아니기 때문이다. 기계 전자적 구성요소와 그 구성요소의 네트워크도 포함한 완전한 기능 시스템에 대한 초기 실행 가능 명세(specification)는 좀 더 효율적인 개발과 테스팅을 가능하게 한다. IP-Protection, 재활용성, 그리고 더 나은 OEM-Supplier-Interfaces와 같은 매우 다양한 장점을 지니면서도, 기능들은 최종 타깃에 의존하지 않고 개발될 수 있다. 이런 접근법을 사용한 디자인은 개발 초기에 검증될 수 있다. 또한 어떤 개별 기능이라도 그것의 특정한 요구사항에 의해서 검증될 수 있다. 이런 점을 통해서 개발하고 있는 전체 제어 시스템이 원하는 시스템 명세들을 만족하고 있다는 것을 보장한다. 모델링 환경을 위한 자동 코드 생성기의 사용이 구현 단계에서의 수고를 크게

감소시키지만, 자동 코드 생성을 사용하더라도 구현 측면에서 테스트에 대한 노력은 여전히 높다. 반면에 자동 코드 검증 기능을 도입하게 되면, 테스팅은 모델 수준까지 다다를 수 있다. 자동 코드 검증의 아이디어는 계층적 모델에 기반을 한다. 이 모델은 그에 상응하는 코드 구현 위에서 더욱 자동화된 구조적 테스팅을 위한 동작 Reference(소위 "golden device", "reference model"라고 불림)로서 작동한다. 이러한 테스팅 방법의 기본 요소는 자동 테스트 생성, 실행, 분석, 그리고 reachability 분석이다. BTC Embedded Tester®의 테스트 생성과 reachability 분석은 자동 코드 생성기에 의해서 생성된 타깃 코드 그 자체에서 수행된다. 첫 번째 분석 단계에서 필수적인 모든 테스트 케이스는 자동적으로 생성될 것이다. 게다가 이러한 테스트 케이스는 자세한 테스트 정보를 포함한 하이퍼링크로 연결된 보고서와 함께 사용자에게 보고될 것이다. 테스팅 과정 동안에 이 보고서는 테스트 활동을 유발시키는 테스트 센터로서 사용된다. Condition coverage, decision coverage, CDC 및 MC/DC와 같은 coverage criteria와 함께 scaling, division-by-zero, saturation, type casting처럼 구현과 연관된 실패원(failure source)들이 검토 대상이다. 이는 참조 수준과 구현 수준들(SiL, PiL, 적용 가능하다면 HiL까지)을 대표하는 모델 수준들 사이에서 동적 테스트와 분석을 통해 자동화된 구조적 비교를 하기 위해서다.

자동 테스트 생성과 코드 검증

자동화된 테스트 환경은 특정 coverage criteria를 다루기 위하여 어떠한 입력 stimuli sequence라도 찾아낼 수 있다. 여기에 소개된 기술은 테스트 케이스 생성과 코드 검증 분석에 필요한 소프트웨어 동작을 표현하기 위해서, 자동으로 생성된 C-code를 사용하고 있다. Test sequence 생성 외에도 임의로 정의된 분석 깊이까지 도달할 수 없는 코드 분기점들 또한 파악될 수 있다. 이러한 능력은 15년이 넘게 성공적으로 사용된 정형 기법(formal method) 분야의 특정 알고리즘들이 사용되어 이용 가능하다.



【그림 2】 코드 검증 개념

그림 2는 타깃에 독립적인 기능 모델들부터 시작해서 타깃 관련된 구현 모델들에서 C-code 생성, 마지막으로 evaluation target 위에서 동작하는 컴파일된 object code까지의 V-프로세스 좌측을 보여주고 있다. 검증 환경 분석을 위한 밑바탕은 C-code이다. 검증 환경의 자동 테스트 벡터 생성("ATG") 기능은 코드와 레퍼런스 모델을 빠짐없이 다루기 위해서 검증 환경이 가지는 올바른 입력 Stimuli 집합을 찾기 위해 타깃 C-code를 직접 사용하고 있다. 서로 다른 coverage criteria들은 바람직한 특정 coverage rate를 최대화하기 위해서 테스트 벡터를 생성하는 동안에 측정된다. 생성된 그리고/또는 import된 stimuli vector들은 내부 data bank에 저장된다. 이러한 벡터들은 필요한 비교 레퍼런스 데이터를 얻기 위해 서로 다른 개발 수준(Functional Model, Implementation Model, Code and Object Code Levels)에서 실행(ATE)되기 위해 사용된다. 기록되고 관측 가능한 변수들주 10)은 마지막으로 데이터 스트림 비교 알고리즘을 사용함으로써 자동으로 비교된다. 허용할 수 없는 차이점들은 검증 환경에 의해서 보고되는데, 이를 통해 특정한 signal들에 대해 사용자가 정의한 허용 범위를 고려한다. 계층적 테스트 접근법을 사용해서 자동 코드 생성기와 함께 테스트 및 검증 환경을 긴밀하게 툴 통합함으로써, 심지어 매우 거대한 산업의 애플리케이션들에 이르기까지 확장성이 보증된다.

현재 아래의 code coverage criteria가 지원된다.

- Statement Coverage
- Condition Coverage
- Decision Coverage
- Switch-Case-Coverage
- Function-Call-Coverage
- Condition Decision Coverage ("CDC")
- Modified Condition / Decision Coverage ("MC/DC")

구현과 관련된 측면을 확인하는데 중요한 테스트 케이스들은 다음과 같다.

- Division by-Zero,
- Type Range Violations (Over- and Underflows)
- Type-Casting
- Saturation and
- Relational Operations (Fixed-Point vs. Floating-Point),

만약 테스트 벡터 생성 알고리즘들이 코드와 모델을 완벽하게 처리할 수 없다면, 검증 환경은 정형 기법 테크닉을 적용하여 빠진 coverage 특성의 reachability를 평가한다. 다른 방법들과는 대조적으로, handling율이라 불리는 것이 이 방식에 의해서 도입된다. 최우수 사례를 살펴보면,100% handling율

이 100% coverage율보다 더 나은 매트릭스를 제공하는 것을 보여줬다. 이것은 예를 들면 division 근처의 안전 코드 때문에 보통 조건 하에서 100% Coverage율은 결코 도달할 수 없기 때문이다. 이는 reachability 분석이 이러한 테스팅과 검증 접근법의 중요한 요소가 된다는 것을 보여준다.

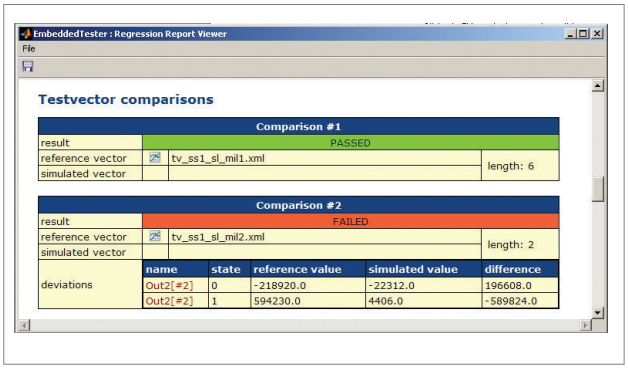
자동화된 테스트 실행

산업용 규모의 애플리케이션을 넘어서는 확장성을 보장해주는 계층적 접근법 때문에, 자동으로 생성된 Stimuli sequence는 특정 기능이나 시스템의 행위에 대한 반응을 기록하기 위해서 그에 상응하는 계층 실체(interface)의 각각 다른 실행 수준에서 실행될 수 있다. 필요한 test harness를 생성하는 것은 사용자의 어떠한 노력이나 간섭 없이도, 검증환경에 의해 완전히 자동으로 생성된다. 이에 의해 테스팅 중에 테스트를 받고 있는 시스템(target code)에 손을 대거나 수정하지 않아도 된다. 이런 접근방법은 자동으로 생성된 stimuli-vector들을, 입력 시퀀스와 계산된 이에 대응하는 출력 (관찰 가능한) 기대 값으로 구성된 실제 테스트 벡터들로 완성시킨다.

자동화된 테스트 평가

백투백 테스팅 모드에서, 검증 환경은 모든 수준(MiL, SiL, PiL)에서 reference (output) 값을 포함하여 실행된 테스트 케이스들을 완전히 자동으로 비교하며 차이점들을 자동으로 생성된 보고서로 보여준다. 데이터 스트림들의 자동화된 비교를 세밀하게 조정하는데 있어서 허용 범위 또한 정의될 수 있다. 자동화된 테스트 평가에서 고정소수점 대비 부동소수점 측면은 특별히 다루어진다.

그림 3은 모든 테스트 결과를 요약한 테스트 평가 보고서를 보여주고 있다. 이 보고서는 사용자가 정의한 허용 범위에 대해서 기대했던 결과를 내지 않은 테스트가 있었는지를 나타낸다. 또한 이 보고서는 매우 자동화되고 효율적인 디버깅이 가능하도록 관련된 테스트 벡터들을 직접 가리키며 test manage에게 하이퍼링크된다.



【그림 3】 테스트 평가 보고서

디버깅 지원

각 실행 수준들 사이에서 차이점이 발견된다면, 오류의 원인을 찾고 고치는 것이 이슈가 될 것이다. BTC EmbeddedTester®는 링크된 커버리지 보고서와 전용의 디버깅 기능들을 사용자에게 지원한다. 사용자가 정의한 허용 범위를 벗어나면, 자동으로 생성되고 하이퍼링크로 연결된 보고서들이 타깃 코드와 그에 대응하는 모델들 간의 차이점을 보여준다. 허용할 수 없는 차이에 대한 이유를 분석하기 위해서, 한 번의 마우스 클릭으로 사용자는 그에 대응되는 관련 코드나 모델 부분으로 이동할 수 있다. 이런 장점은 모델링, 코드 생성, 그리고 검증 환경이 긴밀하게 통합되어야만 달성될 수 있다. 마지막으로 테스트 벡터들은 MiL, SiL 수준(예를 들면, Visual C-Debugging-Project의 automatic export)에서 단계적으로 디버깅될 수 있다.

개방적인 Import 및 Export 인터페이스

BTC EmbeddedTester®는 테스트 벡터들과 XML, MAT, XLS, CSV, CTC 등의 많은 파일 포맷들 사이의 import 및 export를 지원한다. 이 기능은 새로 만들거나 다양한 소스들로부터 가져온 기존에 존재하는 테스트 세트들을 쉽게 사용할 수 있게 한다. 테스트 케이스를 import 한 후, 검증 환경은 이에 대해 도달한 code coverage 비율을 보여준다. 또한 요구사항 기반 테스트나 모델 checker와 같은 검증 환경 등을 통해서 자동으로 만들어진 테스트 케이스들은 재사용될 수 있다. 더욱이 테스트 벡터들은 통합되거나 plugged-in 된 테스트 제작 시스템을 사용함으로써 요구사항을 기반으로 사용자에 의해 대화식으로 정의될 수 있다. 모든 관리되고 있는 테스트 케이스들 역시 HiL 테스팅 환경과 같은 다음 테스팅 단계에서 재사용되도록 export 될 수 있다.

실용적인 경험들

소비자들로부터 얻은 경험을 통해 자동 테스트 벡터 생성을 사용하여 향상된 자동화된 백투백 테스팅 방법이 테스트 생성, 실행 그리고 분석에 대한 노력을 전통적인 수동 방식에 비해 80~90% 정도 감소시킨다는 것이 입증됐다. 추가로, 달성한 품질 수준이 ISO DIS 26262 표준에서 요구하는 것만큼 상당히 향상된다는 것이 증명됐다. 특히 이런 경험이 branch coverage나 MC/DC coverage와 같은 잘 알려진 covererage criteria를 사용해서 테스트 품질을 측정하는 과정에 매우 도움이 된다. 자동화 방법을 사용하면, coverage율을 25% 이상 쉽게 높일 수 있다. 이렇게 잘 통합된 기술의 또 다른 장점은 거의 완벽히 자동화된 디버깅을 지원한다는 점이다. 최초 사용자의 반응에 따르면 완전 수동 접근법과 비교해서 최소 50%의 시간이 절약된다.