본 내용은 기사 원문을 ㈜카이젠컨설팅이 번역한 자료임을 알려 드립니다. 본 내용에 대한 저작권은 일본 IAR SYSTEMS에 있으며 내용의 개편 및 수정이 불가합니다.

기능안전 대응을 위해 소프트웨어 개발 단계에서는 무엇을 해야 할까?

최근 IEC61508, ISO26262 등의 기능안전에 관한 이슈가 늘어나고 있습니다. 하지만 기능안전에 관하여 상세하고 엄밀하게 이야기를 하면 소프트웨어 개발자는 알아듣기 힘들고, 들으면 들을수록 이해하기 어려울 겁니다. 따라서 여기에서는 ISO26262에 대한 엄밀한 이야기는 하지 않고, 소프트웨어의 개발 영역에서 기능안전 규격에 대응하기 위해서는 무엇을 생각해야 하는가에 대해서 집중하여 설명을 하고자 합니다.

기능안전을 위한 소프트웨어 개발에서 필요한 포인트

기능안전을 위한 소프트웨어 개발에서 필요한 포인트를 크게 3가지로 정리해 보았습니다.

1. 소프트웨어 안전 요구사항이 반영된 설계의 검증

2. 개발 프로세스의 명확한 정의

3. 사용하는 툴의 검증

① 소프트웨어 안전 요구사항이 반영된 설계의 검증

자주 듣는 질문 중 “지금까지 임베디드 소프트웨어 개발을 해왔는데, 그것만으로는 충분하지 않습니까?”라는 것이 있습니다만, 확인해 보면 거의 불충분한 케이스뿐이었습니다. 그 이유는 IEC61508이나 ISO26262 등의 기능안전 규격에서는 소프트웨어의 안전 요구사항을 제대로 명확히 반영하는 것을 요구하기 때문입니다. 지금부터 개발 Phase 마다 확인해 보고자 합니다.

소프트웨어 안전 요구사항

시스템의 안전과 관련된 소프트웨어 기능에 대해서는 소프트웨어의 안전 요구사항을 명확하게 정의해야 합니다. 안전 요구사항이 문서화 되어 있지 않으면 규격을 준수하는 것은 불가능합니다.

소프트웨어 레벨에서의 기능안전 요구사항은 안전 컴포넌트나 시스템 설계 등에 반영되어야 하고 아래와 같은 내용으로 명확히 해야 합니다.

본 내용은 기사 원문을 ㈜카이젠컨설팅이 번역한 자료임을 알려 드립니다. 본 내용에 대한 저작권은 일본 IAR SYSTEMS에 있으며 내용의 개편 및 수정이 불가합니다.

- 시스템의 안전 상태를 유지하는 기능
- 안전과 관련된 부품의 고장 검출 및 대응
- 소프트웨어 자체의 고장 검출, 표시, 제어
- 성능 및 시간 제약의 엄격한 처리

또한 아래와 같은 내용도 고려하여야 합니다.

- 시스템과 하드웨어의 Configuration
- 소프트웨어와 관련된 하드웨어 사양
- 하드웨어와 소프트웨어의 인터페이스 사양
- 타이밍 제약
- 외부 신호
- 동작 모드 (평소, 테스트 시 등)

이 내용만 보더라도 단순하게 소프트웨어 자체만 검토하면 되는 것이 아니라, 시스템이나 하드웨어의 요구사항과의 일관성도 명확히 해야 하는 것을 알 수 있습니다.

소프트웨어 아키텍처 설계

소프트웨어에 대한 안전 요구사항이 정의되면 다음은 소프트웨어 아키텍처 설계를 실시하게 됩니다.

먼저 규격에서는 정형 기법이나 준정형 기법을 사용한 방법을 권고하고 있습니다. 또 현장에서도 준정형 기법인 UML 사용이 증가하고 있습니다. 설계 시에도 명확한 설계 원칙을 적용하는 것을 요구하고 있으며, 테스트의 용이성 보수성 등도 고려해야 합니다. 예를 들어

- 소프트웨어 컴포넌트 사이즈 제한
- 소프트웨어 부품의 응집도가 높고 결합도가 적은 것
- Interrupt의 이용 제한

등과 같은 점들을 설계 시에 고려해야 합니다. 그러기 위해서는 실제 설계 시 위의 내용을 어떻게 확인할 것인가? 판정 기준은 무엇인가? 준수할 수 없는 경우에는 어떻게 할 것인가?와 같은 것에 대한 검토가 필요합니다.

이러한 사항들은 제대로 된 소프트웨어를 만들기 위한 개발 프로세스 등을 명확히 수립하고, 업무 순서 및 툴, 판단 기준 등을 정의함으로써 해결하여야 합니다.

본 내용은 기사 원문을 ㈜카이젠컨설팅이 번역한 자료임을 알려 드립니다. 본 내용에 대한 저작권은 일본 IAR SYSTEMS에 있으며 내용의 개편 및 수정이 불가합니다.

안전성 확보를 위해서는 먼저, 안전 분석에 따라 안전 관련 부품의 명확화, 에러 검출 방법과 제어를 위한 기구 설계 부분을 적용하고, 아키텍처 설계에 대한 정적 분석 및 동적 분석을 실시하여 안전 메커니즘을 검증 해야 합니다. 이렇게 하지 않으면 기존 소프트웨어에 개발 관행에서 결국 빙빙 돌게 됩니다.

소프트웨어 단위 설계 및 구현

아키텍처 설계에 근거하여 단위 설계 및 구현을 실시합니다. 단위 설계에서는 몇 가지의 설계 원칙을 적용하도록 규격에서는 정의하고 있으며 그 예로

- 함수에서의 하나의 입구와 하나의 출구
- 동적 메모리 사용하지 않는다.
- 변수는 초기화하여 이용한다.
- 변수의 이름을 중복으로 사용하지 않는다.
- 포인터 사용에 제한

등이 규정되어 있고 이는 MISRA-C를 통해 많은 것이 커버됩니다. 또한 단위 설계의 검증으로는 아래 항목을 들 수 있습니다.

- Walk through/Inspection
- Semi-formal/Formal 기법
- Control flow 분석
- Data flow 분석
- 정적 code 분석

이러한 규칙들을 적용하여 문서화하는 것이 가능합니까? 그렇지 않다면 먼저 이러한 설계 룰 및 가이드라인을 작성하는 것부터 시작하여야 합니다. 설계 룰 및 가이드라인에 대해서는 ②에서 설명합니다.

소프트웨어 단위 테스트 및 통합 테스트

단위 테스트를 위해서는 아래의 테스트 기법을 적용하여야 합니다.

- 요구사항 base 테스트
- Interface 테스트

본 내용은 기사 원문을 ㈜카이젠컨설팅이 번역한 자료임을 알려 드립니다. 본 내용에 대한 저작권은 일본 IAR SYSTEMS에 있으며 내용의 개편 및 수정이 불가합니다.

- Fault injection 테스트
- Resource 테스트

특히 안전 기능을 확인하기 위해서는 Fault injection 테스트와 같이 인위적으로 위험 상황을 만들어 안전성을 확인하는 것이 중요합니다. 이 테스트는 예를 들어 특정 오류로 인해 CPU 고장이 발생하더라도 정해진 시간 내에 정상 상태로 돌아오는가 등을 확인하는 케이스입니다.

테스트 케이스를 도출하는 방법으로는 요구사항 분석, 동치 분할, 경계값 분석 등이 있습니다.

테스트 케이스의 완전성 및 의도하지 않은 기능이 없다는 것을 확인하기 위해서는 커버리지를 측정하여야 합니다. 단순히 결과값을 측정하는 것이 아니라 제대로 된 커버율을 달성하고 있냐는 것을 확인해야 합니다.

소프트웨어 안전 요구사항 테스트

정의된 요구사항이 임베디드 소프트웨어에서 동작할 수 있는가를 확인하기 위해 HILS 및 실제 기기 등에서 테스트해야 합니다.

② 개발 프로세스의 명확한 정의

①에서는 소프트웨어의 안전 요구사항이 올바르게 구현되어 고장 시에도 대응할 수 있다는 것을 증명해야 한다는 것을 설명했습니다.

개발 프로세스 정의를 위해서는 개발 초기 관련 활동, 개발 중 관련 활동 등을 명확히 해야 합니다.

개발 초기에는 룰 결정을 해야 합니다. 개발 프로세스 및 프로그래밍 언어, 개발 툴, 설계 및 코딩 가이드 등이 필요합니다. 예를 들어 C언어는 정의하기 애매한 부분, 실행 시 예외 처리가 약한 부분, 구조 설계 등을 강제하지 않는 부분 (비교적 자유도가 높은 언어) 등 때문에 더욱더 설계 원칙 및 코딩 가이드를 적용하여 개발을 수행해야 합니다. 개발자 개개인의 능력에 맡기는 것이 아니라, 제대로 규정된 프로세스 기반으로 소프트웨어를 개발하는 것이 중요합니다.

구체적으로는 MISRA-C 등을 사용한 코딩 가이드 및 설계 방침, 명명 규칙 등이 필요합니다.

MISRA-C의 도입도 단순히 툴을 구입하면 끝나는 것이 아니라, 툴에서 확인할 수 없는 룰은 어떻게 확인할 것이며, 지킬 수 없는 룰은 어떻게 예외 처리를 할 것인지, 그 타당성은 어떻게 보증할 것인지를 정의하여야 합니다.

설계 방침으로는 안전 기능과 비안전 기능을 어떻게 분리할 것인가, 소프트웨어 구조에 대한 함수의 크기, 복잡한 조건 식의 제한 등을 정의하여야 하며, 또한 제한하는 것뿐만 아니라 위반 사항을 어떻게 검출할 것인가 등 여러 가지 부분을 결정하여야 합니다.

본 내용은 기사 원문을 ㈜카이젠컨설팅이 번역한 자료임을 알려 드립니다. 본 내용에 대한 저작권은 일본 IAR SYSTEMS에 있으며 내용의 개편 및 수정이 불가합니다.

기존 개발 활동 중에도 소프트웨어와 관련된 프로세스를 수행할 때에는 당연히 실시하는 부분이겠지만, 특히 기능안전과 관련된 부분을 관리하기 위해서는 각 Phase 마다 무엇을 실시해야 하는지, 성과물은 무엇인지 등을 명확히 정의해 두어야 합니다. 또한 기능안전에서 중요한 포인트가 Traceability입니다. 소프트웨어가 구현되면 안전 요구사항의 반영 여부를 제대로 확인하여야 합니다. 요구사항대로 구현되었는지를 확인하기 위해 안전 요구사항이 단위 소프트웨어 어디에 구현되어 있는가? 등을 추적해야 합니다. 이를 이용하여 안전 요구사항이 제대로 반영되어 구축되고 있는가에 대한 검증, 안전 요구사항의 변경 시 어디에 영향을 끼치는가를 추적할 수 있게 됩니다. 이러한 관리를 위해 어떻게 해야 한다 까지는 규격에 규정되어 있지 않지만, 이는 툴을 사용하지 않으면 힘든 작업입니다. 또한 요구사항이나 설계가 변경되었을 때 변경이 제대로 반영되었는지, 변경에 따른 영향 범위는 어떤지 등에 대한 변경 관리도 해야 합니다.

③ 사용하는 툴의 검증

ISO26262에서는 툴이나 외부 소프트웨어(자사 개발이 아닌 구매)에 대해서도 규정이 있습니다. 소프트웨어 개발에 이용하는 툴은 먼저 툴의 Use-case를 명확히 하고 툴에 버그 발생 시 영향도를 판단하는 것부터 시작합니다. 툴에 결함이 발생한 경우 해당 버그가 안전 기능에 영향을 끼치지 않으면 툴 임팩트(TI)가 1, 영향을 끼치면 TI가 2가 됩니다.

또한 결함이 발생했을 때에 해당 결함을 검출할 수 있는 가능성이 높음, 중간, 낮음에 따라 툴 에러 검출성(TD)을 1,2,3으로 정의합니다. 이 결과를 통해 툴에 요구되는 신뢰 레벨(TCL)을 결정합니다.

	TD1	TD2	TD3
TI1	TCL1	TCL1	TCL1
TI2	TCL1	TCL2	TCL3

툴에 대한 인증은 TCL 레벨(숫자)이 커질수록 엄격한 인증을 요구합니다. 구체적으로 인증을 위해서는 ISO26262 규격에서는 ・이용 실적, ・개발 프로세스 평가, ・소프트웨어 툴 검증, ・안전 규격 준수 등에 따라 인증하도록 되어 있습니다. 그러나 소프트웨어 개발을 위한 대표적인 툴인 컴파일러를 예를 들면, 이용 실적 및 소프트웨어 툴 검증은 유저 측에서 확인이 가능하지만, 개발 프로세스 평가나 안전 규격 준수 여부 등은 유저 측에서 확인하기 어려운 부분이 있습니다.

툴의 검증에 대해서는 벤더 측에서 검증 작업을 완료하여 릴리즈 하고 있는 툴도 때문에, 벤더 측에서 검증한 툴을 구입할 것인지, 일반 툴 구입 후 자사에서 검증을 할 것인지 등을 판단할 필요가 있습니다. 툴 선정을 위해서는 이러한 부분도 판단하는데 있어서 중요한 요소입니다.

결론

기능안전 규격을 준수하는 소프트웨어를 개발하기 위해서는 안전 요구사항에 기반한 아키텍처/단위 설계를 실시하고 기존에 실시하던 테스트뿐만 아니라 안전 요구사항에 대한 테스트를 제대로 실시하여야

본 내용은 기사 원문을 ㈜카이젠컨설팅이 번역한 자료임을 알려 드립니다. 본 내용에 대한 저작권은 일본 IAR SYSTEMS에 있으며 내용의 개편 및 수정이 불가합니다.

합니다. 그러기 위해서 중요한 것은 개인의 능력에 의존하지 않고 설계 원리・코딩 가이드 등을 준비하고 각 개발 Phase에서 해야 하는 일을 정의한 개발 프로세스를 수립하여야 합니다.

툴에 대해서도 Use case에 따라 결함의 영향을 분석하고 영향 정도에 따라 툴의 검증을 실시하여야 합니다. (혹은 검증된 툴 이용)

또 자주 있는 오해 중 하나가 「기능안전을 준수한 툴을 사용하면 이를 통해 개발한 소프트웨어도 기능안전을 준수한다.」라는 것인데, 여기까지 읽으신 분은 이미 아시겠지만, “기능안전을 준수한 툴을 사용하는 것”은 기능안전 소프트웨어 개발을 위해 만족해야 할 1개 항목에 불과합니다. 또한 툴의 검증 작업은 힘든 작업이기 때문에 검증된 툴을 사용하는 것이 유저의 부담을 줄이는데 효과적일 것입니다.