* 간사이 사업부 姫電 개발 프로젝트실

자동차 기능안전 규격 ISO 26262의 소개

ISO26262 Automotive Functional Safety Standard

글: 茂野一彦* • Kazuhiko Shigeno | 출처: MSS기보 • Vol.23 (8장~11장)

번역: 이채원 • 카이젠컨설팅

8. 기능안전의 소프트웨어 개발

기능안전 대응에 따라 소프트웨어의 개발은 종래(從來)의 방법과 비교하여 무엇이 바뀐 것인가? 본 장에서는 이 점에 대해서 설명한다.

8.1 소프트웨어 안전 요구사항의 Traceability

ISO 26262에서는 각 Hazard의 리스크 저감을 안전 목표로 설정하고 그들이 확실하게 구축/시험 되었는지를 개발 공정 전체에 걸쳐 추적할 수 있게 하는 것을 요구하고 있다. 이것을 안전 요구사항의 Traceability라고 부른다. Traceability는 요구사항에 ID 번호를 붙여서 식별한다.

소프트웨어 개발에서는 ID 번호를 사용하여 소프트웨어 안전 요구사항(소프트웨어 기능 사양에 해당), 소프트웨어 아키텍처 설계 사양의 소프트웨어 컴포넌트, 소프트웨어 단위 설계 사양의 함수를 연결한다(그림 4). 이에 따라 안전 요구사항의 추적이 명시적으로 이루어지고 상위 요구사항에 대한 하위 요구사항의 누락이나 함수의 오류가 미치는 요구사항으로의 영향 등을 검증이나 시험에서 효과적으로 발견하여 대처하는 것이 가능하다.

안전 요구사항의 Traceability 관리를 위해서는 소프트웨어 개발 등에서 사용되는 요구사항 관리 툴*(주 10)*을 사용한다.

(주 10) 미국 IBM사의 "Rational DOORS"나 미국 PTC사의 "MKS Integrity" 등이 있다.

8.2 설계 방법

ISO 26262에서는 높은 ASIL이 요구되는 소프트웨어에 대해서 보다 엄격한 제약의 설계 방법이 요구된다.

소프트웨어 아키텍처 설계에서는 ASIL D의 경우, 소프트웨어 컴포넌트의 계층화, 소프트웨어 컴포넌트

* 간사이 사업부 姫電 개발 프로젝트실

사이즈의 제한, 적절한 스케쥴링, 소프트웨어 컴포넌트의 결속성, 결합 제한, Interrupt 제한이 필수 요건이다.

이러한 소프트웨어 설계를 위한 각종 방법은 구조 설계 방법 등에서 이야기하고 있는 사고 방식으로써 새로운 내용은 아니지만, ISO 26262에서는 ASIL 레벨에 대응하기 위해서는 필수적으로 적용할 것을 명시하고 있다.

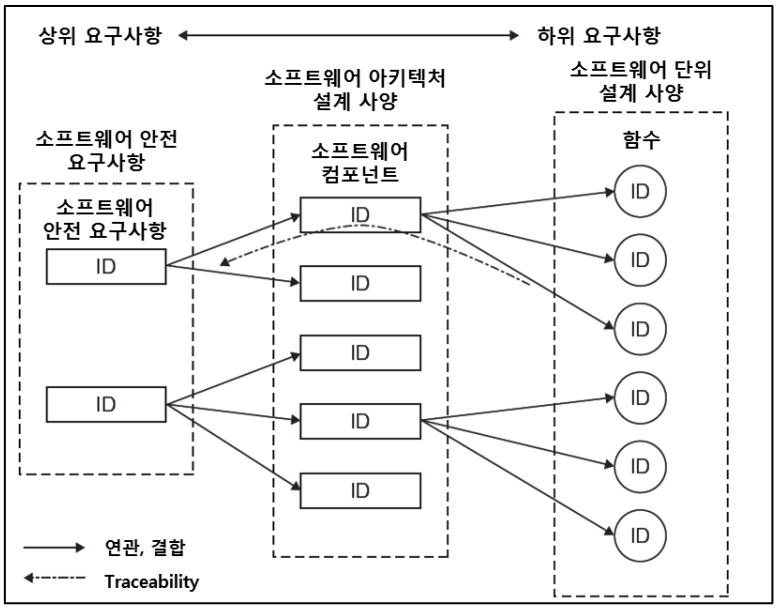

그림4 소프트웨어 안전 요구사항의 연결 및 추적

소프트웨어 아키텍처 레벨에서의 에러 검증 활동으로는 ASIL D의 경우, 입출력 데이터의 범위 체크, 데이터 타당성의 체크, 외부 모니터링, 제어 Flow 감시, 소프트웨어 용량 설계가 필수 요건이다.

소프트웨어 아키텍처 설계의 에러 처리에서는 ASIL D의 경우, 에러 발생시의 중첩 기능, 병렬의 용량성이 필수가 된다.

소프트웨어 단위 설계에서는 ASIL D의 경우, 함수에 1개의 입구, 1개의 출구, 히프 영역 등 동적으로 확보되는 오브젝트의 제한, 변수 초기화의 실시, 동일 변수명의 금지, 글로벌 변수의 사용 제한, 포인터의 사용 제한, 암시형 변환의 금지, 숨은 데이터 Flow/제어 Flow의 금지*(주 11)*, 무조건 점프 금지, 재귀 호출의 금지가 있다.

(각 설계 방법의 상세한 내용에 대해서는 소프트웨어 설계 기법 등의 서적을 참조하길 바란다.)

ISO 26262에서 모델링 설계에 대해서는 깊게 언급되어 있지 않다. 자동차의 모델링 설계(모델베이스

개발)에서는 MATLAB/Similink가 대표적인 방법이지만 모델링 설계를 실시했다고 해서 그것이 곧 기능 안전에 기여하는 것은 아니다.

모델링 설계뿐만 아니라 코드의 품질과 함께 모델의 품질을 높이는 것이 만들어지는 제품(Item)의 안전성에 기여한다.

따라서 ISO 26262에서 모델을 설계하는데 필요한 가이드라인에 대해서는 코딩 가이드라인과 함께 사용하는 형태로 요구사항을 정의하고 있다.

(주 11) 숨은 데이터 Flow/제어 Flow의 금지란, 제어 Flow가 복수의 의미를 갖는 것을 금지하는 것이다. 설계자가 인식하고 있지 않은 제어가 구축되는 것에 의해 문제가 발생할 리스크가 증가한다. 예를 들어 다음의 예가 숨은 제어 Flow에 해당한다.

[예] 설계자 A가 함수 A의 제어 Flow에서 글로벌 변수 X를 1개씩 가산해 가고, X가 10이 될 때 변수 Y에 1을 대입한다 라는 설계를 실시하게 된다. 한편, 다른 설계자 B가 함수 B에서 원래라면 의존 관계가 없는 함수 A의 변수 X를 변수 Z에 1을 대입하기 위해 판정 조건으로 참조하는 제어 Flow를 포함하면 변수 X는 설계자 A가 인식하고 있지 않은 숨겨진 Flow에 대해서도 영향을 미친다.

즉 이 상황에서 설계자 A가 변수 X에 대해 제어를 변경하면 설계자에게 인식되는 것이 아니라 함수 B에 대해서 영향이 미치고 만다. 이러한 영향을 회피하기 위해 숨은 데이터 Flow/제어 Flow는 금지되고 있다.

8.3 검증

검증(Verification)이란 “요구사항에 대해 ‘올바르게’ 성과물이 작성되는 것”의 확인이며 리뷰 등에 따라 상위 요구사항이 누락 없이 하위 요구사항에 전개되고 있는가 라는 관점에서 실시한다.

ISO 26262에서는 소프트웨어 아키텍처 설계, 소프트웨어 단위 설계와 구축(코딩)에 대해 ASIL 레벨에 의한 검증이 규정되어 있다.

소프트웨어 아키텍처 설계 검증은 ASIL A의 경우 Walk-through만 실시하여도 되지만, ASIL D에서는 Inspection, 동적 분석(실행 가능 모델)에 근거한 시뮬레이션, 프로토타이핑, 제어 Flow/데이터 Flow 해석*(주 12)*을 필수 요건으로 하고 있다.

소프트웨어 단위 설계와 구축(코딩) 검증은 ASIL A의 경우, Walk-through만 실시하여도 되지만, ASIL D에서는 Inspection, 반정형검증(모델에 근거한 시뮬레이션 등을 이용한 검증), 제어 Flow/데이터 Flow의 해석*(주 13)*, 정적 분석이 필수이다.

(주 12) 소프트웨어 아키텍처 설계 검증에 사용하는 제어 Flow/데이터 Flow 해석은 예를 들어 에러

* 간사이 사업부 姫電 개발 프로젝트실

검출 시에 시스템이 Fail-safe 상태에 전이할 수 있는지, 기능 오류의 경우 Redundancy 기능이 유효하게 되는 지 등을 확인한다. 이 해석에 의해 안전 메커니즘의 제어 Flow가 올바른 순서로 실시되고 있는지 검증하는 것이 가능하다.

(주 13) 소프트웨어 단위 설계와 구축 검증에 사용하는 제어 Flow/데이터 Flow 해석이란 제어의 흐름이나 데이터의 흐름을 확인하는 것이다. 소프트웨어 단위 설계에 근거한 제어 Flow나 데이터 Flow의 구축이 실시되고 있는지 해석을 실시한다.

8.4 타당성 확인

타당성 확인(Validation)이란 “요구사항에 대해 ‘올바르게’ 성과물이 작성되는 것”의 확인을 실시하는 것이며 시험에 따라 요구사항을 만족한 성과물이 만들어졌는가 라는 관점에서 실시한다.

ASIL D의 경우 소프트웨어 단위 시험, 소프트웨어 통합 시험과 함께 요구사항에 근거한 시험, 인터페이스 시험, 결합 주입 시험, 리소스 시험*(주 14)*, 백투백 시험*(주 15)*이 필수이다.

시험 케이스의 도출 방법은 요구사항 분석, 동등 클래스 분석 및 생성, 경계값 분석이 ASIL D의 필수 요건이다.

소프트웨어 단위 시험의 커버리지는 ASIL A의 경우 C0 커버리지(명령 커버리지)의 100% 실시로 좋은 것에 대해, ASIL D에서는 C1 커버리지(분기 커버리지) 및 MC/DC (Modified Condition/Decision Coverrage)*(주 16)*의 100% 실시가 요구되고 있다.

소프트웨어 통합 시험에서는 기능 커버리지*(주 17)* 및 콜 커버리지*(주 18)*의 100% 실시가 ASIL D의 요건이다.

(주 14) 코드가 실행 가능한지, ROM/RAM/레지스터 등의 리소스가 어느 정도 사용되고 있는지를 확인하는 시험이다.

(주 15) 모델 베이스 개발에 의해 작성된 실행 가능한 모델의 시뮬레이션과 자동 생성된 코드에 같은 시험 케이스를 부여하고, 그 실행 결과를 비교하여, 확인을 실시하는 시험이다.

(주 16) 모든 코드 중의 조건이나 판정에 대해 한 번은 코드를 실행한다. 이 때 판정의 조건은 판정의 출력이 독립하도록 시험을 실시한다. 조건의 수를 n으로 할 때, 전체 조건의 조합을 시험하는 C2 커버리지(복합 조건 커버리지)의 시험 케이스 수가 2의 n승 개가 되는 것에 대해, MC/DC의 케이스 수는 n+1 개로 된다. 즉, 조건이 많아질수록 C2 커버리지 보다 비용적으로 유리하게 된다.

(주 17) 소프트웨어 전체의 기능 수와 실행된 기능의 배합이다. 이에 따라 실행 기능의 커버리지를 확인하는 것이 가능하다.

* 간사이 사업부 姫電 개발 프로젝트실

(주 18) 전체의 함수(또는 기능)와 실행된 함수(또는 기능)의 배합이다. 이에 따라 호출된 함수의 커버리지를 확인하는 것이 가능하다.

8.5 소프트웨어 툴의 인정

ISO 26262는 개발 시 사용 목적으로 도입하는 각종 소프트웨어 툴(이하, 간단하게 툴로 표기)에 대해서 신뢰성 평가 지표를 정하고 그 지표에 적합하다고 인정되는 툴을 개발에 사용할 수 있도록 하고 있다.

우선 툴이 인정의 대상이 될지 선별을 실시한다.

선별을 위해서는 TI(Tool Impact), TD(Tool error Detection), TCL(Tool Confidence Level)이라는 지표를 사용한다.

TI란 특정 툴이 오작동할 경우에 개발 중인 안전 관련 아이템 또는 컴포넌트에 에러가 발생하거나 에러 검출에 실패할 가능성이다. TI1과 TI2 단계에서 평가한다. TI1은 그것 이외의 경우(즉 어떠한 영향이 있는 경우)에 선택한다.

TD란 툴이 오작동하여 이에 따라 오출력을 일으키는 것을 방지하는 수단, 또는 툴이 오작동하여 이에 따른 오출력을 일으키는 것을 검출하는 수단에 대한 신뢰성이다. TD1, TD2, TD3의 3단계로 평가한다. TD1은 오작동과 이에 따른 오출력을 방지 또는 검출할 수 있는 신뢰성의 정도가 높은 경우에 선택한다. TD2는 같은 신뢰성의 정도가 중간 정도의 경우에 선택한다. TD3은 그것 이외의 모든 케이스를 선택한다.

이렇게 해서 요구된 TI와 TD에 대해서 표 5에 나타나는 매트릭스에서 TCL을 결정한다.

TCL1과 판정된 툴은 인정하지 않더라도 사용할 수 있다.

TCL2 및 TCL3과 판정된 툴은 다음의 4종류의 인정 방법을 적용한다.

표5 TCL의 결정표

	TD1	TD2	TD3
TI1	TCL1	TCL1	TCL1
TI2	TCL1	TCL2	TCL3

① 사용 실적에 의한 신뢰성의 향상

툴에 대해서 규격에서 정해진 사용 실적에 관한 근거가 제공되는 경우에 한해 사용 실적에 의한 신뢰도

의 향상이 논증되는 것으로 평가하는 등의 방법.

② 툴의 개발 프로세스 평가

툴의 개발에 적용되는 개발 프로세스가 규격에 적합하고 있는가를 평가하는 등의 방법.

③ 툴의 타당성 확인

규격에서 정해진 판정 지표를 만족하는 타당성 확인을 실시하는 방법.

④ 안전 규격에 따른 개발

툴의 개발이 예를 들면 ISO 26262, IEC 61508, RTCA DO-178(항공 시스템이나 장치의 안전 규격)이라는 규격에 만족하고 있는가를 검증하는 방법.

이들의 인정 방법의 적용은 툴에서 개발하는 아이템 또는 컴포넌트의 ASIL에 따라 다르다.

TCL2의 경우, ASIL A/B/C에서는 ①과 ②가, ASIL D에서는 ③과 ④가 필수이다.

TCL3의 경우, ASIL A/B에서는 ①과 ②가, ASIL C/D에서는 ③과 ④가 필수이다.

또 툴을 벤더로부터 구입하는 경우 툴의 이용자가 인정을 실시하는 것은 현실적으로 곤란하다. ISO 26262에 적합한 제3자 인증을 받은 툴을 도입하는 것이 현실적이라고 할 수 있다.

8.6 보호 기능에 의한 결함의 파급 방지

ISO 26262에서는 특정 소프트웨어 컴포넌트에 결함이 있을 때, 그것이 다른 소프트웨어 컴포넌트에 악영향을 미치는 것을 막기 위해 구조를 “소프트웨어 파티셔닝” 으로 규정한다. 이 기술은 ASIL의 다른 복수의 소프트웨어 컴포넌트를 동일의 MPU상에서 공존시키는 경우에 이용할 수 있다. 예를 들어 ASIL A와 ASIL B와 ASIL C의 소프트웨어가 있다고 하자. 이것을 파티션 없이 1개의 MPU와 1개의 리소스(공유 메모리)에서 작동시키는 경우, 3개의 소프트웨어를 모두 최고 ASIL인 ASIL C 등급으로 적용하지 않으면 안되고 결국은 필요 이상으로 높은 개발 비용이 들게 돼버린다. 한편 리소스로 파티션이 있으면 1개의 MPU에서도 각각의 소프트웨어 컴포넌트에 최적인 ASIL을 적용하는 것이 가능하고 쓸데없는 개발 비용을 피하는 것이 가능하다.

소프트웨어 파티셔닝 기술에는 여러 가지가 있지만 자동차용 MPU에 소프트웨어 파티셔닝 기술을 적용하는 경우 현시점에서는 비용면에서 MPU의 메모리 보호 유닛을 이용한 OS에 의한 방법이 가장 유용한 것으로 여겨지고 있다.

8.7 프로세스 개선

* 간사이 사업부 姫電 개발 프로젝트실

임베디드 소프트웨어 분야에서는 CMMI, Automotive SPICE 라는 각종 프로세스 개선 규격의 조합을 계속적으로 실시하는 것이 요구된다.

ISO 26262에서도 프로세스 개선이 요구되고 있지만 이것은 기본적으로 기존의 프로세스 개선의 연장선상에 있으며, CMMI의 레벨2, Automotive SPICE의 레벨 3의 인증을 취득하고 있으면 좋다고 여겨지고 있다.

ISO 26262에서는 Part2 기능안전의 관리 중 기술적 관점부터 성과물을 검증하는 Confirmation Review, 프로세스적 관점부터 기능안전 활동의 구축 상황을 확인하는 기능안전 Audit, 아이템의 안전성에 대해서 기능안전의 적합성을 평가하는 기능안전 Assessment의 3가지 세트를 정리한 Confirmation Measurement에 대해서 ASIL에 대응한 독립성을 갖고 실시하도록 규정하고 있다.

독립성은 4단계로 정의되어 있으며, 가장 낮은 레벨은 “Confirmation Measurement을 다른 인물에게 실시하는 것이 좋다”라는 정도이지만, 가장 높은 레벨은 “다른 부서 혹은 조직의 인물(즉, 제3자)에 의해 실행되지 않으면 안 된다”로 규정되어 있다. Confirmation Measurement의 내용에 따라 ASIL 레벨이 높으면 독립성도 높은 레벨이 요구된다.

또 제 3자 조직에 대해서는 품질보증부 라는 사내의 독립된 부서라도 좋고, 사외의 제3자 인증 기관이라도 좋다. 보통은 영업 전략상 이름이 알려진 제3자 인증 기관에 감사를 의뢰하는 공급자나 툴 벤더가 많다. 대표적인 제3자 인증 기관의 하나로 독일 TUV SUD사가 있다.

9. 안전 문화

ISO 26262에서는 조직 차원에서 안전 활동을 보증할 수 있도록 별도 조직의 구축을 요구하고 있으며, 따라서 안전한 제품 개발을 계속할 수 있는 프로세스 인프라의 구축이 요구된다. 프로세스 인프라의 구축에 있어서는 “사람은 누구라도 잘못을 저지를 가능성이 있다”라는 성악설의 입장에 서서 조직의 입장을 생각할 필요가 있다.

거기에서 등장하는 개념이 “안전 문화(Safety Culture)”이며 ISO 26262는 조직(및 개인)이 안전을 중시하는 기업의 풍토를 구축해 가는 것이다.

안전 문화란, 조직 차원에서 안전에 관하여 마땅히 있어야 할 형태(목적 달성을 위해 충분한 효과가 있고 이해관계자가 내용을 충분히 이해하여 합의를 얻을 수 있는 상태)의 규칙이나 프로세스가 구축되고, 적절한 교육을 통해 개발 활동에 따라 각각의 담당자에게 “안전에 관한 공통의 상식”으로써 스며들게 해야 하는 것이다. 이것은 하루 아침에 정착할 수 있는 것은 아니다. 그렇기 때문에 ISO 26262에서는 조직의 표준 프로세스를 정의하고 각 프로젝트의 특성에 맞춘 테일러링(표준 프로세스의 개조)을 실시하는 계획과 계속적인 프로세스 개선 활동이 요구되고 있다.

* 간사이 사업부 姫電 개발 프로젝트실

ISO 26262는 프로세스의 구축이 최종적인 목표가 아니다. 구축한 프로세스로 조직이 요구하는 목표를 달성했을 때 프로세스의 구축이 의미를 갖게 되는 것이다.

10. ISO 26262 추진에 필요한 조직 체제

ISO 26262 추진에 필요한 조직 체제의 예를 그림 5에 나타냈다.

이는 CMMI나 Automotive SPICE의 체제에 Safety Management Team을 더한 체제의 모양이다.

이 체제는 규격 내에서 직접 요구되고 있는 것은 아니지만 ISO 26262 대응을 추진하기 위해서는 추진 체제의 구축이 불가결하다.

CMMI나 Automotive SPICE에 의한 프로세스 개선에서는 소프트웨어 개선 추진 조직인 SEPG (Software Engineering Process Group)을 둔다. 또는 소프트웨어 개발의 프로세스에 한정하지 않고 프로세스 개선을 추진하는 조직을 EPC (Engineering Process Group)이라고도 부른다. 이 EPG가 ISO 26262의 추진 그룹으로서 역할을 완수해 내기 위해서는 기능안전에 관한 활동을 포함한 개발 활동 전체에 대해 적절한 조직 표준 프로세스를 구축하고 그것을 실시하는 각 담당자에게 필요한 교육을 실시하는 것이다. 또 조직 표준 프로세스를 유지하기 위해 필요한 데이터나 정보를 수집하여 프로세스를 개선하는 것이다.

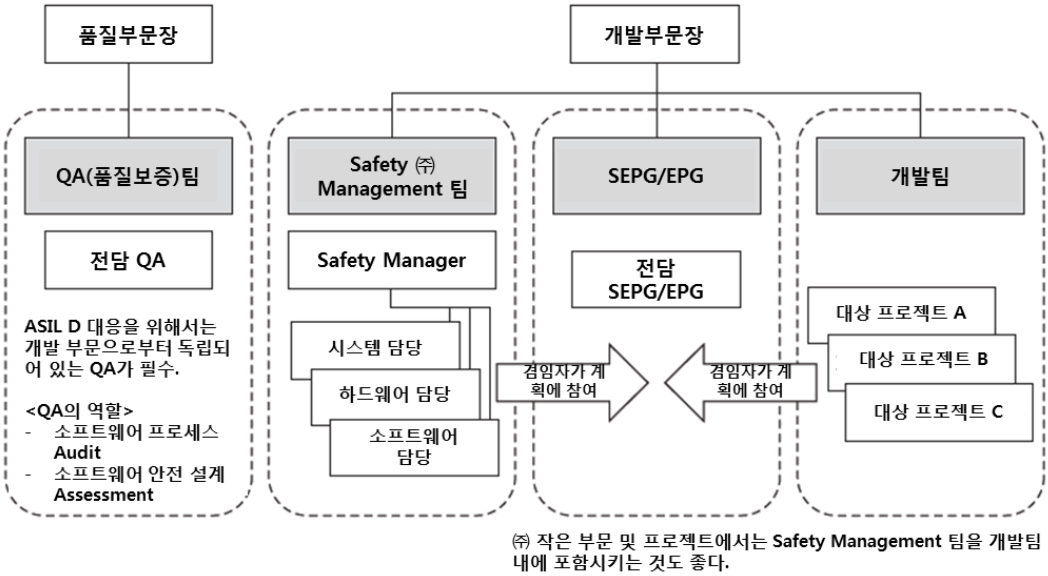

그림5 ISO 26262 추진에 필요한 조직 체제

* 간사이 사업부 姫電 개발 프로젝트실

11. 분산 개발(개발 위탁)의 요건

기능안전을 적용하는 아이템이나 컴포넌트(소프트웨어를 포함)를 개발 위탁할 경우 수탁하는 공급자(소프트웨어 수탁 회사를 포함)도 ISO 26262가 요구하는 요건을 만족하지 않으면 안 된다.

개발을 위탁하기 전에는 개발 위탁자와 공급자 간에 개발 인터페이스 합의서(DIA-Development Interface Agreement)를 주고 받아야 한다. DIA에는 개발 위탁자와 공급자 쌍방의 Safety Manager, 안전 라이프사이클의 테일러링*(주 19)*, 개발 위탁자와 공급자 각각의 활동과 프로세스, 쌍방에서 교환하는 정보나 활동 등이 포함된다.

개발 활동 내에 공급자는 진척, 리스크 관리, 결함을 개발 위탁자에게 보고하지 않으면 안 된다. 공급자가 개발이 불가능한 안전 요구사항이 있는 경우 안전 컨셉을 재검토하고 안전 요구사항을 다시 점검할 필요가 있다.

아이템의 안전성에 영향을 미칠 가능성이 있는 변경이 발생 시 영향 분석을 실시하기 위해 개발 위탁자와 공급자가 서로 통보할 필요성이 있다. 이 때문에 사전에 안전의 타당성 확인을 개발 수탁자가 실시할 것인지, 공급자가 실시할 것인지를 결정해 두지 않으면 안 된다.

(주 19) ISO 26262에서의 안전 라이프사이클이란 제품의 컨셉부터 개발, 생산, 운용, 폐기에 이르는 제품 라이프사이클 전체를 아우른 안전관리의 라이프사이클이다.

안전 라이프사이클의 테일러링이란 ISO 26262에서 정해진 안전 라이프사이클을 조직의 특성에 맞춰 새로 조정하는 행위이다. 조직에 따라서 각 Phase의 활동을 통합하거나 분할하거나 반복하거나 또는 다른 Phase에서 실시할 수 있다. 〈끝〉