IEC 61508 Part 3 기반
안전 소프트웨어 개발 정리

The Safety Critical Systems Handbook, Chapter 4 학습 노트

Chapter 4는 IEC 61508 Part 3의 소프트웨어 요구사항을 설명한다. 핵심 대상은 랜덤 하드웨어 고장이 아니라 요구사항, 설계, 코딩, 시험 및 변경 과정에서 발생하는 **체계적 고장**이다. 따라서 소프트웨어 안전무결성은 고장률을 직접 예측하여 입증하지 않는다. 목표 SIL에 맞는 생명주기, 설계기법, 검토, 시험, 형상관리 및 적합성 입증을 적용하여 체계적 고장을 예방하고 발견한다.

이 문서의 구성

1. 핵심 키워드를 정의한다.
2. 키워드 간 관계를 V-Model과 생명주기 관점에서 정리한다.
3. 원문의 흐름에 따라 소프트웨어 요구사항부터 적합성 입증까지 설명한다.
4. 형상관리를 별도로 정리하여 기준선, 추적성, 변경통제, Release의 관계를 명확히 한다.

Contents

1 Chapter 4의 본질

1.1 적용 범위

IEC 61508 Part 3은 안전관련 소프트웨어의 개발을 다룬다. 적용 대상은 크게 다음 두 경우로 구분된다.

- **Embedded software design**: 하드웨어와 밀접하게 결합된 임베디드 소프트웨어를 개발하는 경우
- **Application software**: 검증된 PLC 플랫폼 등에 응용 로직을 구성하는 경우

두 경우 모두 소프트웨어 생명주기가 필요하다. 다만 생명주기의 단계 수와 검증 강도는 목표 SIL과 시스템 복잡도에 따라 달라진다.

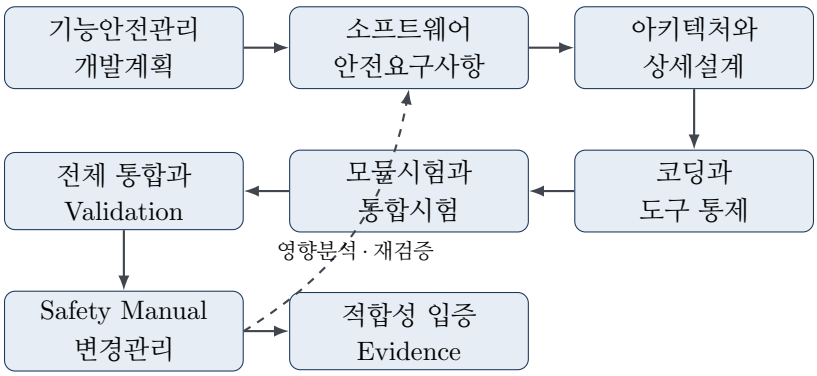
1.2 소프트웨어 안전무결성의 의미

하드웨어 신뢰도 예측은 예상 고장률을 계산한다. 그러나 소프트웨어에 정성적 기법을 적용했다고 해서 체계적 고장의 고장률이 계산되는 것은 아니다. Chapter 4가 주장할 수 있는 범위는 다음과 같다.

정량적 오해 방지

목표 SIL에 맞는 기법을 적용했다는 사실은 소프트웨어 고장률을 수치로 직접 입증한 것이 아니다. 현재의 기술수준에서 해당 SIL에 적절한 생명주기와 검증수단을 적용하여 체계적 고장을 충분히 억제했다고 판단하는 것이다.

1.3 전체 흐름



2 키워드 정의

2.1 기본 개념

키워드	정의
IEC 61508 Part 3	안전관련 소프트웨어의 요구사항, 설계, 코딩, 시험, Validation, 변경 및 적합성 입증을 다루는 규정이다.
체계적 고장	요구사항 오류, 설계결함, 코딩오류, 도구오류 또는 변경오류처럼 특정 조건에서 반복되는 고장이다. 일반적인 부품 고장률 방식으로 직접 예측하지 않는다.
SIL	Safety Integrity Level의 약어이다. 소프트웨어에서는 SIL이 높아질수록 요구사항 표현, 설계기법, 시험범위, 독립성 및 문서화가 강화된다.
응용 소프트웨어	검증된 플랫폼 위에 특정 공정이나 안전기능을 구현하는 로직이다. PLC 래더, Function Block 등이 대표적이다.
임베디드 소프트웨어	하드웨어 기능과 밀접하게 결합되어 개발되는 소프트웨어이다. 고급언어와 어셈블러가 사용될 수 있으며 더 상세한 설계와 분석이 필요하다.

2.2 생명주기와 관리

키워드	정의
Software Development Plan	조달, 개발, 통합, Verification, Validation, 변경, 형상관리, 산출물 및 책임자를 정의하는 계획이다.
V-Model	요구사항에서 코드로 구체화하고, 코드에서 모듈 · 통합 · 전체 시스템 시험으로 올라가며 각 설계단계와 시험단계를 대응시키는 생명주기 모델이다.
Verification	각 단계의 산출물이 이전 단계의 입력과 요구사항을 올바르게 만족하는지 검토하고 시험하는 활동이다.
Validation	최종 시스템이 전체 안전목표를 만족하며 필요한 절차가 수행되었는지를 최종 확인하는 활동이다.
형상관리	형상항목을 식별하고 기준선을 설정하며 추적성, 변경통제, 영향평가, Release 및 폐기를 관리하는 활동이다.
Baseline	검토와 승인이 완료되어 이후 변경이 공식 변경절차를 통해서만 허용되는 형상의 기준점이다.
추적성	안전요구사항이 설계, 코드, 시험항목 및 시험결과와 양방향으로 연결되는 성질이다.

2.3 요구사항과 설계

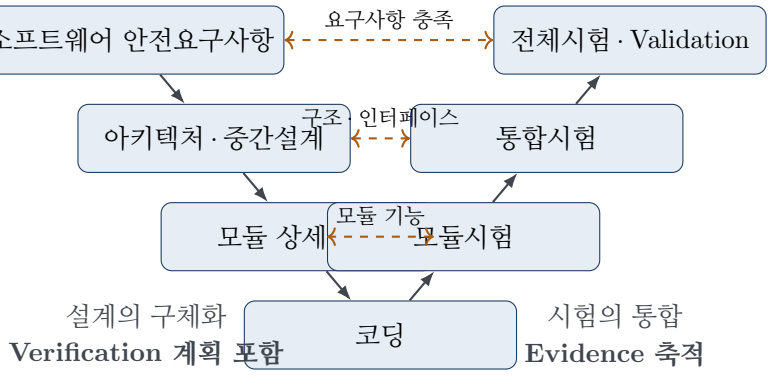
키워드	정의
Software Safety Requirements Specification	안전기능과 소프트웨어 안전무결성 요구사항을 명확하고 시험 가능한 형태로 정의하는 문서이다.
반정형 기법	Logic Diagram, Cause and Effect Chart, State Transition Diagram, Truth Table, Data Flow Diagram 등 자연어보다 구조화된 표현방법이다.
모듈화	기능을 작고 관리 가능한 단위로 분리하여 복잡도를 낮추고 시험과 변경을 용이하게 하는 설계원칙이다.
비간섭성	비안전 기능의 오류나 자원사용이 안전관련 기능의 수행을 방해하지 않는 성질이다. 공유 RAM, 주변장치, CPU 시간, 통신 및 오류전파를 검토한다.
방어적 프로그래밍	입력 범위, 타당성, 상태전이 및 오류조건을 검사하여 비정상 상황을 안전하게 처리하는 코딩방법이다.
제한된 언어 부분집합	동적 객체, 무조건 분기, 과도한 인터럽트 · 포인터 등 검증을 어렵게 하는 기능을 제한한 프로그래밍 규칙이다.
지원도구	Compiler, PLC Engineering Tool, Code Generator, 정적분석 도구, PFD 계산 프로그램 등 생명주기 결과에 영향을 주는 소프트웨어 도구이다.

2.4 시험, 재사용 및 적합성 입증

키워드	정의
모듈시험	개별 소프트웨어 모듈이 의도한 기능을 수행하고 의도하지 않은 기능을 수행하지 않는지 확인하는 시험이다.
모듈 통합시험	모듈을 단계적으로 결합하여 기능, 인터페이스, Black Box 동작 및 성능을 확인하는 시험이다.
시험 커버리지	요구사항, 코드경로, 경계조건 또는 상태조합이 시험으로 얼마나 확인되었는지를 나타내는 지표이다.
Safety Manual	재사용 소프트웨어의 기능, 구성, 가정, 설치조건, 도구버전, 미해결 결함, 호환성, 인터페이스 제약 및 변경요청 방법을 제공하는 문서이다.
Trusted/Verified Module	이전에 설계 · 시험되었거나 유사한 용도에서 사용된 소프트웨어 모듈이다. 현재 환경에 대한 적합성 검토는 별도로 필요하다.
Data-driven System	기본 시스템의 기능을 데이터, 파라미터 또는 제한된 프로그램으로 구성하여 응용기능을 만드는 시스템이다.
정적분석	코드를 실행하지 않고 제어흐름, 데이터흐름, 변수사용 및 연산관계를 분석하는 기법이다.
형식기법	수학적 표기와 논리를 이용하여 요구사항과 설계를 엄밀하게 표현하고 검증하는 기법이다.
적합성 입증	표준의 각 요구사항과 이를 충족하는 계획서, 사양서, 설계서, 검토서 및 시험성적서를 연결하는 문서화 활동이다.

3 키워드 간 관계

3.1 V-Model 관계



V-Model의 핵심

V-Model은 코딩 후 시험을 시작하는 방식이 아니다. 요구사항과 설계를 작성할 때 대응 시험방법과 합격기준을 함께 정하고, 각 개발단계의 결과를 대응 시험단계에서 확인하는 방식이다.

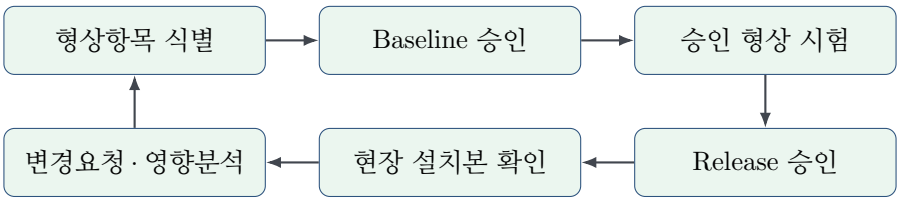
3.2 Verification과 Validation의 차이

구분	Verification	Validation
핵심 질문	각 산출물을 올바르게 만들었는가?	완성된 시스템이 전체 안전목적을 만족하는가?
적용 시점	요구사항, 설계, 코딩, 모듈통합 등 각 단계	전체 시스템 통합 후 최종 확인
주요 방법	Review, Code Review, 모듈시험, 통합시험	Acceptance Test, 요구사항별 종합 확인, 절차 준수 확인
주요 증거	검토기록, 모듈시험성적서, 통합시험성적서	Validation Plan, 최종 시험결과, 불일치 종결기록

3.3 SIL 상승에 따른 요구수준

항목	SIL 1–2	SIL 3	SIL 4
요구사항 표현	핵심 안전부분에 반정형 기법	모든 요구사항에 반정형 기법	모든 요구사항과 핵심부분의 컴퓨터 지원
코딩	구조화 · 제한된 언어, SIL 2 이상 동적 객체와 무조건 분기 금지	방어적 프로그래밍, 인터럽트 · 포인터 · 재귀 제한 강화	SIL 3 수준에 더 엄격한 적용
시험	경계값시험, 입력영역 분할시험	중요 사건의 Cause-Consequence 기반 시험 추가	SIL 3 수준에 더 높은 엄격성
Validation	SIL 2 이상 커버리지 지표 제시	정확성 · 일관성 · 코딩 규칙 준수의 강화된 확인	형식기법 등을 포함한 최고 수준의 확인
변경 후	SIL 1 변경 모듈, SIL 2 영향 모듈 재검증	전체 시스템 재Validation	전체 시스템 재Validation과 더 엄격한 증거

3.4 형상관리와 시험증거의 관계



시험결과는 시험한 형상의 버전이 명확할 때만 유효한 증거가 된다. 형상관리는 설계한 소프트웨어, 시험한 소프트웨어, Release한 소프트웨어 및 현장 설치본이 동일하다는 것을 보증한다.

4 원문 흐름에 따른 설명

4.1 소프트웨어 엔지니어링의 조직과 관리

소프트웨어 개발은 기능안전관리체계 안에서 수행해야 한다. 개발계획에는 조달, 개발, 통합, Verification, Validation, 변경, 형상항목, 산출물 및 책임자를 포함한다. 프로젝트의 복잡도와 목표 SIL에 맞추어 생명주기를 정의하고 문서로 설명해야 한다.

표준은 V-Model을 권고하지만 Waterfall 등 다른 모델도 사용할 수 있다. 다만 V-Model과 동등한 설계단계, 검증단계, 추적성 및 검토 특성을 포함해야 한다. SIL 2 이상에서는 표준 생명주기 활동과 다른 부분을 명시적으로 정당화하고 검토해야 한다.

원문 대응: 4.1 Organizing and Managing the Software Engineering

4.2 형상관리 절차 설정

원문은 소프트웨어 형상관리 절차에 다음 내용을 명확히 포함하도록 요구한다.

1. 형상통제를 시작하는 생명주기 단계
2. 기준선을 설정하는 위치와 승인방법
3. 요구사항 추적방법
4. 변경통제 절차
5. 변경 영향평가
6. Release와 폐기 규칙

SIL 2 이상에서는 형상통제를 가장 작은 컴파일 모듈 또는 단위까지 적용해야 한다.

원문 대응: 4.1, software configuration management process

4.3 소프트웨어 안전요구사항명세서 작성

Software Safety Requirements Specification에는 안전기능과 안전무결성 요구사항을 함께 규정한다. 원문이 제시하는 주요 항목은 다음과 같다.

- 처리용량과 응답시간
- 장비 및 운전자 인터페이스와 오사용
- 소프트웨어 자기감시
- 안전상태를 강제하는 기능
- 데이터 저장공간의 Overflow와 Underflow
- 데이터 손상
- 허용범위를 벗어난 값
- 운전 중 안전기능의 주기적 시험
- 모든 운전모드, 유지보수 요구, 내부 · 외부 인터페이스

요구사항은 명확하고 정확하며 모호하지 않아야 한다. 또한 상위 안전사양과 관련 문서까지 추적할 수 있어야 한다. 명세는 형상통제 단위까지 내려가야 한다.

시험 가능한 요구사항의 예

모호한 표현: “이상압력이 발생하면 신속히 정지한다.”
시험 가능한 표현: “압력이 설정값 이상으로 지정된 시간 동안 유지되면 차단출력을 정해진 응답시간 이내에 발생시키고, 수동 Reset 전까지 재기동을 금지한다.”

SIL 1과 SIL 2에서는 핵심 안전부분에 반정형 기법을 사용한다. SIL 3과 SIL 4에서는 모든 요구사항에 반정형 기법을 적용한다. SIL 4에서는 핵심 안전부분에 컴퓨터 지원도구를 추가한다.

원문 대응: 4.2 Requirements Involving the Specification

4.4 아키텍처 설계와 비간섭성

설계방법은 모듈화를 지원하고 복잡도를 줄여야 한다. 기능, 정보흐름, 데이터구조, 실행순서, 시간제약 및 설계가정을 명확하게 표현해야 한다.

시스템 소프트웨어는 다음 진단기능을 포함해야 한다.

- 시스템 하드웨어 고장 진단
- 통신링크 오류 검출
- 표준 응용모듈의 온라인 시험
- 고장 검출 시 고장 요소를 격리하거나 안전상태로 전환하는 처리

SIL 1과 SIL 2에서는 Watchdog와 직렬통신 오류검출 같은 기본 진단이 필요하다. SIL 3과 SIL 4에서는 센서, I/O 회로, 논리처리부, 출력요소, 통신 및 메모리까지 고장검출을 확대한다.

안전기능과 비안전기능이 같은 플랫폼에 존재하면 비간섭성을 확인해야 한다. 검토대상은 공유 RAM, 주변장치, CPU 처리시간, 기능 간 통신 및 한 요소의 오류가 다른 요소로 전파되는 가능성이다.

원문 대응: 4.3.1 Features of the Design and Architecture

4.5 상세설계와 코딩

상세설계와 코딩은 작고 관리 가능한 모듈을 만들어야 한다. 모든 SIL에서 반정형 설계방법, 설계표준, 코딩표준 및 구조적 프로그래밍을 적용한다.

SIL 2 이상에서는 가능한 한 유사한 응용에서 사용되고 시험된 Trusted/Verified Module을 사용한다. 그러나 재사용 사실만으로 현재 시스템 적합성이 자동 보장되지는 않는다. 적용환경과 인터페이스를 확인해야 한다.

오프라인 도구로 충분히 확인하기 어려운 동적 객체는 사용하지 않아야 한다. SIL 3과 SIL 4에서는 변수의 범위와 가능하면 타당성을 함께 검사하는 방어적 프로그래밍을 적용하며, 인터럽트, 포인터 및 재귀의 사용을 제한한다.

원문 대응: 4.3.2 Detailed Design and Coding

4.6 프로그래밍 언어와 지원도구

프로그래밍 언어는 완전하고 모호하지 않게 정의할 수 있어야 한다. 모든 SIL에서 코딩표준과 제한된 언어 부분집합을 사용한다.

- SIL 2 이상: 동적 객체와 무조건 분기를 금지한다.
- SIL 3과 SIL 4: 인터럽트와 포인터 제한을 강화하고, 도구 오류에 대비한 다양성 기능을 검토한다.

지원도구는 충분한 사용실적이 있고 알려진 오류가 해결되었거나 안전시스템 용도에 적합한 인증을 받아야 한다. SIL 3과 SIL 4에서는 인증된 도구가 더 강하게 권고된다.

이 요구는 Compiler나 PLC 도구에만 적용되지 않는다. 안전루프 PFD 또는 고장률을 계산하는 오프라인 패키지도 완전성과 정확성을 평가하고 명확한 사용설명서를 제공해야 한다.

원문 대응: 4.3.3 Programming Language and Support Tools

4.7 소프트웨어 모듈시험과 통합시험

각 모듈은 Code Review와 시험을 받아야 한다. 시험목표는 다음 두 가지이다.

1. 의도한 기능을 정확히 수행하는지 확인한다.
2. 제한된 비정상 데이터로 의도하지 않은 기능을 수행하지 않는지 확인한다.

모듈시험 후에는 사전에 정의한 시험사례와 시험데이터로 모듈 통합시험을 수행한다. 기능시험, Black Box 시험 및 성능시험을 포함한다.

시험결과는 시간순으로 기록한다. 모듈버전과 시험절차서 버전을 명시하고 예상결과와 실제결과의 불일치를 명확하게 표시한다. 시험 이후 변경이 발생하면 영향분석으로 재시험 범위를 결정한다.

SIL 1과 SIL 2에서는 경계값시험과 입력영역 분할시험을 포함한다. SIL 3과 SIL 4에서는 중요한 사건의 Cause-Consequence 분석에서 도출한 시험을 추가한다.

원문 대응: 4.4.1 Software Module Testing and Integration

4.8 전체 통합시험

전체 통합시험은 하드웨어와 소프트웨어를 포함하는 통합 시스템을 시험한다. 이 단계는 Factory Acceptance Test까지 이어질 수 있다. 시험 Harness도 시험장비의 일부이므로 설계문서와 적합성 확인이 필요하다.

원문은 시험기록을 특히 강조한다. 시험기록은 시험결과를 확인할 수 있는 유일한 가시적 증거이기 때문이다.

원문 대응: 4.4.2 Overall Integration Testing

4.9 Validation

Validation은 최종 시스템이 모든 목표를 만족하고 설계절차가 준수되었음을 확인한다. Validation Plan은 전체 안전요구사항이 어떻게 확인되는지를 보여야 한다.

- 생명주기 전체와 감사시점

- 구체적인 합격·불합격 기준
- 선정한 Validation 방법과 선정 근거
- 부적합 사항의 처리방법

SIL 2 이상에서는 시험 커버리지 지표가 제시되어야 한다. SIL 3과 SIL 4에서는 정확성, 일관성 및 코딩 규칙 등 표준 준수를 더욱 엄격하게 확인한다.

원문 대응: 4.5 Validation

4.10 재사용 요소의 Safety Manual

재사용 소프트웨어 요소에는 Safety Manual이 필요하다. 주요 내용은 다음과 같다.

- 요소의 기능과 속성
- 구성과 설계가정
- 통합자에게 요구되는 최소 지식
- 요소에 의존하는 정도
- 설치방법과 Release 사유
- 미해결 결함과 추가된 기능
- 이전 버전 및 다른 시스템과의 호환성
- 운영체제 의존성
- Compiler와 도구의 식별정보와 버전
- 요소의 명칭, 버전, 개정상태
- 변경통제와 변경요청 방법
- 인터페이스와 사용자 인터페이스 제약
- Safety Manual 주장에 대한 정당화

Safety Manual은 단순한 인증서가 아니다. 재사용 요소를 어떤 조건과 제한 안에서 사용할 수 있는지 알려주는 통합 지침이다.

원문 대응: 4.6 Safety Manuals

4.11 소프트웨어 변경

모든 변경에는 변경로그, 개정관리, 변경사유, 영향분석 및 재시험이 필요하다. 변경절차는 최초 설계 때 적용한 방법과 최소한 동등해야 한다. 변경기록에는 변경된 문서와 변경내용을 명확히 표시한다.

목표 SIL	변경 후 요구되는 확인범위
SIL 1	변경된 모듈을 재Verification한다.
SIL 2	변경으로 영향을 받는 모든 모듈을 재Verification한다.
SIL 3 이상	전체 시스템을 재Validation한다.

소프트웨어가 포함된 SIL 3 시스템에서는 전체 재Validation이 상당한 비용을 발생시킬 수 있다. 따라서 초기부터 변경영향을 제한할 수 있도록 모듈화와 추적성을 확보해야 한다.

원문 대응: 4.7 Modifications

4.12 대체기법과 절차

표준에 직접 나열되지 않은 대체기법도 사용할 수 있다. 다만 다음 속성을 평가해야 한다.

- 안전요구에 대한 완전성
- 안전요구에 대한 정확성
- 사양 오류와 모호성의 부재
- 안전요구사항의 이해 용이성
- 비안전 소프트웨어의 부정적 간섭 방지
- Verification과 Validation의 기반 제공

원문은 평가수준을 다음과 같이 설명한다.

적용 수준	등급	의미
SIL 1–2	R1	제한된 객관적 합격기준. 예: Black Box Test, Field Trial
SIL 3	R2	높은 신뢰도를 갖는 객관적 기준. 예: 커버리지 지표를 갖는 시험
SIL 4	R3	객관적이고 체계적인 논증. 예: Formal Proof
해당 없음	–	제안 기법에 관련되지 않는 속성

원문 대응: 4.8 Alternative Techniques and Procedures

4.13 Data-driven System의 네 가지 유형

Data-driven System은 기본 시스템에 입력되는 데이터, 파라미터 또는 제한된 프로그램으로 응용기능을 구성한다. 복잡도와 사용자 자유도에 따라 생명주기를 Tailoring한다.

유형	대표 예	핵심 확인사항
Limited variability configuration / Limited application configurability	Smart Sensor, Smart Actuator의 파라미터 설정	입력 파라미터 명세, 정확한 적용, 모든 조합의 Validation, 특수 운전모드, 인적요인, Interlock 유지, 임의 재설정 방지
Limited variability configuration / Full application configurability	대규모 정적 데이터로 시스템을 구성하는 경우	데이터 생성 자동화도구, 데이터 일관성, 규칙 준수, 데이터 준비시스템 인터페이스의 유효성
Limited variability programming / Limited application configurability	Function Block, Ladder Logic, Spreadsheet 기반 시스템	응용요구사항, 허용 언어 부분집합, 부분집합 결합 설계방법, 가능한 시스템 상태 조합을 다루는 Verification 커버리지
Limited variability programming / Full application configurability	그래픽 시스템, SCADA 기반 Batch Control	응용 아키텍처, Template 제공, 개별 Template Verification, 완성된 응용의 Verification과 Validation

원문 대응: 4.9 Data-Driven Systems

4.14 정적분석

정적분석은 코드를 실행하지 않고 소스코드를 대수적으로 분석한다. 제어흐름, 데이터흐름, 변수사용 및 알고리즘의 연산관계를 검토한다. Part 3의 Table B8은 Data Flow와 Control Flow를 SIL 3과 SIL 4에서 Highly Recommended로 제시한다.

정적분석의 한계도 분명하다.

- 언어별 Translator와 분석도구가 필요하다.
- 주로 절차형 고급언어에 적용된다.
- PLC 코드는 자동분석 대신 수동 Walkthrough에 의존할 수 있다.
- 논리기능은 분석하지만 타이밍 특성은 직접 다루지 못한다.
- 정적분석만으로 코드품질 전체를 보증하지 못한다.

원문 대응: 4.10.1 Static Analysis

4.15 형식기법

형식기법은 수학적 표기와 논리를 이용하여 요구사항과 설계를 엄밀하게 정의한다. 장점은 생명주기 초기에 잠재오류를 예방하거나 발견할 수 있다는 점이다. 오류를 시운전이나 현장사용 단계에서 발견하는 것보다 수정범위와 비용이 작다.

그러나 훈련된 인력과 적절한 도구가 필요하며, 모든 응용에 보편적으로 적합한 하나의 방법은 없다. 원문은 Formal Method와 Formal Proof를 SIL 4에서 Highly Recommended, SIL 2와 SIL 3에서 Recommended로 설명한다.

원문 대응: 4.10.2 Use of Formal Methods

4.16 PLC 언어

초기의 Ladder Logic은 분기문이 거의 없는 Limited Variability Language였으므로 모든 SIL에서 비교적 쉽게 적용할 수 있었다. 현대 PLC는 명령집합이 확대되고 분기와 복잡한 기능이 추가되었다. 따라서 높은 SIL에서는 명령어 부분집합을 제한해야 한다.

IEC 61131-3 언어를 사용할 때도 안전관련 응용에 적합한 Restricted Subset을 정의해야 한다. Cause and Effect Diagram으로 Shutdown System을 직접 구성하는 전용언어는 안전응용을 위해 제한된 부분집합의 사례이다.

원문 대응: 4.10.3 PLCs and their Languages

4.17 소프트웨어 재사용

재사용은 이미 설계 · 시험된 사양, 알고리즘 또는 코드를 활용한다. 기대효과는 다음과 같다.

- 기존 코드나 사양에 사용증거가 있다.
- 평균보다 많은 시험을 받았을 수 있다.
- 절감한 시간을 추가 개발과 시험에 사용할 수 있다.
- 실제 응용환경에서 시험되었을 수 있다.
- 재사용을 목적으로 설계한 요소는 결합도가 낮을 가능성이 있다.

반대로 새로운 환경에서 잠재결함이 나타날 수 있고, 불필요한 기능 때문에 수정이 필요할 수 있으며, 내부동작을 충분히 이해하지 못할 수도 있다. 따라서 재사용의 이점은 적절한 절차와 통제가 있을 때만 얻을 수 있다.

원문 대응: 4.10.4 Software Reuse

4.18 Software Metrics

Software Metrics는 코드의 크기, 복잡도 및 구조를 측정한다. 분기문 수가 대표적이다. 시험 커버리지 통계도 Metrics라는 용어로 사용된다.

원문은 Metrics의 가치에 대해 상반된 견해가 있다고 설명한다. 장기간 특정 산업이나 제품군에서 일관되게 수집하면 현장고장 성능과의 상관관계를 연구할 수 있지만, Metrics만으로 안전무결성을 직접 입증하기에는 아직 제한이 있다.

원문 대응: 4.10.5 Software Metrics

4.19 적합성 입증 템플릿

Chapter 4의 최종목표는 IEC 61508 Part 3의 요구사항을 실제 프로젝트 증거와 연결하는 것이다. Evidence란에는 다음 문서를 연결할 수 있다.

- Software Development Plan
- Software Safety Requirements Specification
- 아키텍처 및 모듈 설계서
- 코딩표준과 Code Review 기록

- 형상항목 목록, Baseline 및 변경로그
- 모듈시험 · 통합시험 · Acceptance Test 결과
- Validation Plan과 Validation Report
- Safety Manual
- 영향분석과 재시험 기록

"Not Applicable"은 정당한 근거가 있을 때만 사용할 수 있다. 상대적으로 단순한 응용 소프트웨어는 요약 템플릿으로 평가할 수 있지만, 새로운 하드웨어와 임베디드 소프트웨어를 함께 개발하는 경우 표준의 상세표를 직접 적용할 필요가 있다.

원문 대응: 4.11 Conformance Demonstration Template

5 형상관리의 정확한 정리

5.1 정의

형상관리의 정확한 정의

형상관리는 소프트웨어와 관련 문서 · 데이터 · 도구를 형상항목으로 식별하고, 승인된 버전 조합을 기준선으로 설정하며, 요구사항 추적, 변경통제, 영향평가, Release 및 폐기를 관리하여 설계한 시스템, 시험한 시스템 및 설치된 시스템의 동일성을 보장하는 활동이다.

5.2 형상항목의 범위

구분	형상항목 예시
요구사항	Software Safety Requirements Specification, 인터페이스 요구사항
설계	아키텍처, Cause and Effect, State Diagram, 모듈 설계서
구현	PLC Project, 소스코드, Function Block, Library, 실행파일
설정	파라미터, Trip 값, 통신설정, I/O Mapping
도구 · 환경	Compiler, PLC Engineering Tool, 운영체제, Firmware
시험	시험절차, 시험데이터, Test Harness, 시험결과, 결함목록
운영자료	Safety Manual, Release Note, 변경로그, 폐기상태

5.3 형상관리 절차

1. **식별**: 형상항목마다 고유 ID, 명칭, 버전, 상태 및 책임자를 부여한다.
2. **기준선**: 검토와 승인 후 요구사항 · 설계 · 시험 또는 Release 기준선을 설정한다.
3. **추적**: 요구사항과 설계, 코드, 시험결과를 양방향으로 연결한다.
4. **변경통제**: 변경사유, 영향, 승인, 구현, 재시험 및 새로운 버전을 기록한다.
5. **상태기록**: 현재 승인상태, 시험상태, Release 상태 및 폐기상태를 유지한다.

6. **감사**: 문서, 저장소, 빌드결과 및 현장 설치본이 승인기준선과 일치하는지 확인한다.

5.4 버전관리와의 차이

개념	의미
버전관리	파일의 수정이력, 차이 및 버전을 저장한다.
변경관리	변경의 필요성, 영향, 승인 및 재시험을 통제한다.
Release 관리	승인된 형상조합을 설치 · 사용 가능한 버전으로 공식 발행한다.
형상관리	위 활동을 통합하여 시스템 전체 구성과 증거의 일관성을 보장한다.

Git과 같은 도구는 형상관리를 지원하지만, Git만으로 요구사항 추적, 안전영향분석, 변경승인, 도구버전, 현장 설치본 및 Release 승인을 자동 보증하지는 않는다.

6 최종 요약

1. Chapter 4는 소프트웨어의 체계적 고장을 관리한다.
2. 소프트웨어 고장률을 직접 계산하는 대신 SIL에 맞는 생명주기와 검증 강도를 적용한다.
3. V-Model은 설계단계와 시험단계를 대응시킨다.
4. 안전요구사항은 명확하고 시험 가능하며 양방향 추적이 가능해야 한다.
5. 아키텍처는 모듈화, 진단 및 비간섭성을 확보해야 한다.
6. 코딩은 제한된 언어, 구조적 프로그래밍 및 방어적 프로그래밍을 적용한다.
7. 도구도 사용실적, 오류관리 또는 인증을 통해 적합성을 확인해야 한다.
8. 모듈시험, 통합시험 및 Validation의 결과에는 버전과 불일치 처리기록이 필요하다.
9. 변경은 영향분석과 SIL별 재검증 범위를 적용한다.
10. 최종적으로 표준 요구사항과 프로젝트 Evidence를 연결하여 적합성을 입증한다.

한 문장 정리

Chapter 4는 안전소프트웨어에 오류가 없다고 선언하는 방법이 아니라, 요구사항부터 변경까지 체계적인 생명주기를 적용하여 오류를 예방 · 발견 · 통제하고, 그 수행증거로 목표 SIL 적합성을 입증하는 방법을 설명한다.

참고문헌

David J. Smith and Kenneth G. L. Simpson, *The Safety Critical Systems Handbook: A Straightforward Guide to Functional Safety, IEC 61508, IEC 61511 and Related Guidance*, 4th ed., Butterworth-Heinemann, 2016, Chapter 4, pp. 79–100.