IEC 61508 기능안전 흐름 정리

Chapter 1 → Chapter 2 → Chapter 5 → Chapter 3 → Chapter 4

학습 목적
본 문서는 *The Safety Critical Systems Handbook*의 Chapter 1, 2, 5, 3, 4를 기능안전의 실제 사고 흐름에 맞추어 재배열한 것이다. 전체 흐름은 **위험 인식** → **목표 SIL 결정** → **정량적 신뢰도 계산** → **하드웨어 · 시스템 구현** → **소프트웨어 개발 · 검증**이다.

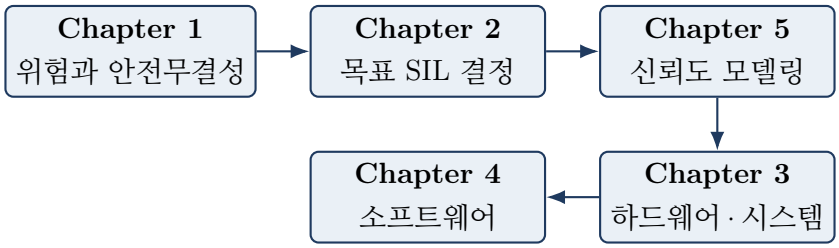
Contents

1 전체 흐름

1.1 학습 순서와 핵심 질문

순서	Chapter	핵심 질문
1	Chapter 1	왜 안전무결성과 SIL이 필요한가?
2	Chapter 2	어떤 안전기능에 어느 SIL을 부여할 것인가?
3	Chapter 5	설계한 시스템이 정량적 목표를 만족하는가?
4	Chapter 3	하드웨어와 전체 시스템을 어떻게 설계 · 검증할 것인가?
5	Chapter 4	소프트웨어의 체계적 고장을 어떻게 방지할 것인가?

1.2 기능안전 전체 관계도



핵심
Chapter 1과 2가 "얼마나 안전해야 하는가"를 정한다. Chapter 5는 "수치적으로 가능한가"를 검토한다. Chapter 3과 4는 각각 하드웨어와 소프트웨어가 그 목표를 실제로 만족하도록 구현하고 입증한다.

2 Chapter 1: 안전무결성이 왜 필요한가

2.1 Zero Risk는 존재하지 않는다

기능안전의 출발점은 **위험을 완전히 제거할 수 없다는 사실**이다.

- 하드웨어에는 물리적 고장이 발생한다.
- 사람은 실수한다.
- 소프트웨어는 모든 운전상황을 완벽히 예측할 수 없다.

따라서 목표는 위험을 0으로 만드는 것이 아니라 **허용 가능한 수준까지 저감하는 것**이다.

현재 위험 → 위험 저감 → 허용 가능한 잔여위험

2.2 랜덤 하드웨어 고장과 체계적 고장

	랜덤 하드웨어 고장	체계적 고장
발생 특성	부품의 열화 · 파손 · 물리적 결함으로 발생	요구사항 · 설계 · 소프트웨어 · 절차의 오류로 발생
예시	센서 단선, 릴레이 접점불량, 밸브 고착	잘못된 Trip 조건, 논리 누락, 변경 영향 분석 누락
평가 방법	고장률과 확률을 이용한 정량적 평가	설계 · 검토 · 시험 · 관리 절차에 의한 정성적 통제

랜덤 하드웨어 고장 → 정량적 평가

체계적 고장 → 정성적 설계 · 관리 규율

2.3 SIL의 두 가지 의미

SIL(Safety Integrity Level)은 단순한 확률 등급이 아니다.

1. **정량적 의미**: 위험고장확률 또는 위험고장빈도의 목표
2. **정성적 의미**: 요구사항, 설계, 검토, 시험 및 관리의 엄격도

즉, SIL 3을 주장하려면 계산값뿐 아니라 SIL 3 수준의 생명주기, 아키텍처, 검증 및 관리 요구도 충족해야 한다.

2.4 안전 생명주기

개념 · 범위 → 위험원 식별 → 안전기능 · SIL 결정 → SRS → 설계 · 제작

→ 설치 · 시운전 → Validation → 운전 · 정비 → 변경 → 폐기

Chapter 1의 결론
위험을 식별하고 허용 가능한 목표를 정한 뒤, 생명주기 전체에서 랜덤 하드웨어 고장과 체계적 고장을 함께 관리해야 한다.

3 Chapter 2: 목표 SIL을 어떻게 결정하는가

3.1 위험 시나리오 정의

목표 SIL 결정은 구체적인 사고 시나리오에서 시작한다.

압력 상승 → 용기 파열 → 유해물질 누출 → 사망 가능성

평가 요소는 사고 결과의 심각도, 발생 가능성, 노출 가능성, 회피 가능성, 기존 보호계층의 성능, 개인위험과 사회적 위험이다.

3.2 필요한 위험저감량

$$RRF_{\text{required}} = \frac{\text{보호 전 위험}}{\text{허용 가능한 위험}}$$

예를 들어 보호 전 사고빈도가 연간 10^{-2}이고 허용 가능한 목표가 연간 10^{-5}라면 $RRF_{\text{required}} = 10^{3}$이다.

3.3 보호계층의 분담

$$\text{기본 공정제어} + \text{경보} \cdot \text{운전자 조치} + \text{기계적 보호장치} + \text{SIF} + \text{비상대응}$$

각 보호계층은 독립적이어야 하며, SIF가 담당해야 할 잔여 위험저감량을 기준으로 목표 SIL을 정한다.

3.4 목표 SIL 결정 방법

방법	개념	특징
정량적 방법	초기사건빈도, 사고전파확률, 보호계층 실패확률을 계산	명확한 수치목표를 제시할 수 있으나 데이터가 필요함
LOPA	사고 시나리오별 독립보호계층을 단계적으로 평가	현장 적용성이 높고 보호계층의 독립성 판단이 중요함
Risk Graph	심각도, 노출빈도, 회피 가능성, 발생 가능성을 범주화	간편하지만 판단의 주관성이 커질 수 있음

3.5 SIL은 안전기능별로 결정한다

SIL은 전체 PLC나 SIS에 일괄 부여하는 것이 아니라 각 **SIF**별로 결정한다. 예를 들어 고압 차단은 SIL 2, 고온 차단은 SIL 1, 화염감시는 SIL 3일 수 있다.

3.6 ALARP와 기능안전관리

목표 SIL을 만족하더라도 추가 위험저감이 합리적으로 실행 가능하다면 검토해야 한다. 역할과 책임, 역량, 독립 검토, 문서체계, 변경관리 및 기능안전평가도 필요하다.

Chapter 2의 결론
위험분석을 통해 각 안전기능이 담당해야 할 위험저감량을 정하고, 이를 SIL 또는 개별 정량목표로 표현한다.

4 Chapter 5: 목표 SIL을 수치로 만족하는가

4.1 신뢰도 모델의 구성

안전기능은 일반적으로 다음의 세 부분으로 구성된다.

$$\text{센서} \rightarrow \text{로직솔버} \rightarrow \text{최종요소}$$

정량평가에는 위험고장률, 진단범위, 자동진단 주기, Proof Test 주기, 수리시간, 중복 구성, 공통원인고장 및 운전자 개입이 필요하다.

4.2 Reliability Block Diagram

직렬구조: 모든 요소가 정상이어야 안전기능이 성공한다. 한 요소만 실패해도 전체 안전기능이 실패할 수 있다.

병렬구조: 여러 경로 중 필요한 수만 정상이어도 안전기능을 유지한다. 중복은 독립고장에는 효과가 있지만 공통원인고장에는 취약하다.

4.3 공통원인고장

CCF(Common Cause Failure)는 중복채널이 같은 원인으로 동시에 실패하는 현상이다. 공통 전원, 동일 환경, 동일 설계결함, 공통 통신망, 동일 정비 오류, 동일 시험장비 및 동일 소프트웨어가 대표 원인이다.

4.4 Fault Tree Analysis

$$\text{안전차단 실패} = \text{센서 실패} \vee \text{로직솔버 실패} \vee \text{밸브 실패}$$

RBD는 성공경로를 표현하고, FTA는 실패경로와 복잡한 원인조합을 표현하는 데 유리하다.

4.5 자동진단과 Proof Test

- 자동진단으로 검출되는 고장: 짧은 시간 내 발견
- Proof Test로 검출되는 고장: 시험주기 동안 잠복
- 실제 Demand에서만 검출되는 고장: 더 긴 시간 잠복 가능

진단범위가 높고 진단주기가 짧을수록 비가용도가 감소한다. 자동진단으로 검출하지 못하는 위험고장은 Proof Test 주기의 영향을 크게 받는다.

4.6 인적요인

잘못된 Bypass, Proof Test 후 복구 누락, 설정값 오입력, 비상조작 실패 및 오정비도 실패경로가 될 수 있다.

> **Chapter 5의 결론**
> 고장률뿐 아니라 중복구조, 잠복시간, 진단, 시험, 수리, 공통원인 및 인적요인을 하나의 모델에 포함하여 PFD 또는 PFH를 계산한다.

5 Chapter 3: 하드웨어와 시스템을 어떻게 설계할 것인가

5.1 안전요구사항명세서

SRS에는 안전기능, 안전상태, Trip 조건, 응답시간, Reset 조건, 운전모드, 전원 상실 시 동작, 외부 인터페이스, 환경조건, Proof Test 요구 및 각 기능의 SIL을 시험 가능한 형태로 정의해야 한다.

5.2 독립성과 분리

EUC, 일반 제어시스템 및 안전시스템 사이에는 전원, 배선, 통신 및 물리적 분리가 필요하다. 독립성을 입증하지 못하면 일반 제어시스템까지 안전관련 시스템으로 평가해야 할 수 있다.

5.3 구조화 · 모듈화 · 진단

검증된 부품, 구조화 · 모듈화, 고장 검출 시 안전상태 전환, Watchdog, 메모리 · I/O 진단, EMC, 온라인 변경 방지 및 운전자 오류 방지를 적용한다.

5.4 SFF와 HFT

SFF는 전체 고장 중 안전상태를 유발하거나 자동진단으로 검출되는 위험고장의 비율이다.

$$SFF = \frac{\text{안전고장} + \text{검출된 위험고장}}{\text{전체 고장}}$$

HFT는 하드웨어 고장을 허용하면서 안전기능을 유지하는 능력이다.

HFT	의미
0	고장 허용 없음
1	한 개의 고장을 허용
2	두 개의 고장을 허용

Type A 또는 Type B, SFF 및 HFT의 조합은 주장 가능한 최대 SIL을 제한한다.

PFD/PFH 정량목표 충족 및 SFF/HFT 아키텍처 제약 충족

5.5 Verification과 Validation

	Verification	Validation
핵심 질문	각 단계의 산출물이 앞 단계 요구를 만족하는가?	완성된 시스템이 최초 안전요구사항을 만족하는가?
주요 수단	검토, 계산, 단위시험, 통합시험	전체 기능시험, 고장삽입, 환경시험, 최종 인수

시험에는 정상조건뿐 아니라 경계값, 비정상 입력, 명세되지 않은 입력, 고장삽입, 환경조건 및 간섭조건을 포함해야 한다.

Chapter 3의 결론
정량적 고장확률뿐 아니라 사양, 아키텍처, 진단, 시험, 운전, 정비 및 변경까지 포함하여 하드웨어와 전체 시스템의 SIL 적합성을 입증한다.

6 Chapter 4: 소프트웨어를 어떻게 개발 · 검증할 것인가

6.1 소프트웨어 고장의 특성

소프트웨어 고장은 일반 부품처럼 마모되어 발생하지 않는다. 대부분 요구사항, 설계, 코딩 또는 변경 과정에 존재하는 체계적 오류이다. 따라서 명확한 요구사항, 구조화 설계, 제한된 언어, 방어적 코딩, 단계적 검증, 형상관리 및 Validation을 적용한다.

6.2 소프트웨어 안전 생명주기

소프트웨어 안전요구사항 → 아키텍처 → 상세설계 → 코딩

→ 모듈시험 → 통합시험 → 전체 시스템시험 → Validation

V-Model은 왼쪽의 요구사항 · 설계단계와 오른쪽의 시험단계를 대응시킨다.

6.3 소프트웨어 안전요구사항

처리용량, 응답시간, 입력값 허용범위, 정상 · 기동 · 정지 · 고장 운전모드, 안전상태 전환, Overflow · Underflow, 데이터 손상, 자기진단, 운전자 인터페이스 및 하드웨어 인터페이스를 명확히 정의해야 한다.

6.4 설계와 코딩

- 구조화 · 모듈화
- 안전기능과 비안전기능의 비간섭성
- 제한된 프로그래밍 언어
- 방어적 프로그래밍
- 범위 및 타당성 검사
- Pointer, Recursion, 동적 메모리의 제한
- 검증된 Library와 Function Block 사용
- Compiler와 개발도구의 적합성 관리

6.5 시험

모듈시험 → 모듈 통합시험 → HW/SW 통합시험 → 전체 시스템시험 → Validation

정상기능뿐 아니라 경계값, 범위 밖 입력, Timeout, 상충 명령, 통신중단, 비정상 상태전이 및 의도하지 않은 기능을 시험해야 한다.

6.6 형상관리

형상관리는 요구사항, 설계문서, PLC 프로그램, Function Block, Parameter, Compiler, Engineering Tool, Firmware, 시험절차 및 시험결과를 하나의 승인 형상으로 관리하는 활동이다. 목적은 **설계한 시스템, 시험한 시스템 및 실제 설치된 시스템의 동일성을 보장하는 것**이다.

6.7 변경관리

변경 요청 → 영향분석 → 승인 → 구현 → 재검증 → 새 기준선

SIL이 높을수록 재검증 범위와 독립성이 강화된다.

Chapter 4의 결론
안전소프트웨어에 오류가 없다고 선언하는 것이 아니라, 오류를 예방 · 발견 · 통제하기 위한 생명주기를 적용하고 그 수행 증거로 목표 SIL을 입증한다.

7 다섯 Chapter의 통합 관계

1. **위험 인식** — **Chapter 1**: 사고 시나리오를 식별하고 랜덤 고장과 체계적 고장을 구분한다.
2. **목표 결정** — **Chapter 2**: 필요한 위험저감량을 계산하고 각 안전기능에 목표 SIL을 부여한다.
3. **정량평가** — **Chapter 5**: 신뢰도 모델을 만들고 PFD 또는 PFH가 목표를 만족하는지 확인한다.
4. **하드웨어 · 시스템 구현** — **Chapter 3**: SRS, 독립성, SFF, HFT, 진단, 시험 및 Validation으로 하드웨어 적합성을 입증한다.
5. **소프트웨어 구현** — **Chapter 4**: V-Model, 구조화 설계, 제한된 언어, 시험 및 형상관리로 체계적 고장을 통제한다.

Chapter 1 : 왜 안전무결성이 필요한가
Chapter 2 : 어느 수준의 SIL이 필요한가
Chapter 5 : 고장확률이 목표를 만족하는가
Chapter 3 : 하드웨어와 시스템이 적절하게 설계되었는가
Chapter 4 : 소프트웨어 체계적 고장이 통제되었는가

최종 정리
위험을 식별하고 목표 SIL을 결정한 뒤, 신뢰도 모델로 랜덤 하드웨어 고장을 계산한다. 이후 하드웨어 아키텍처와 소프트웨어 생명주기를 통해 정량적 · 정성적 요구를 모두 충족하고, 시험 · 운전 · 변경의 객관적 증거로 적합성을 입증한다.

8 기술사 답안용 핵심 요약

구분	핵심 표현
Chapter 1	기능안전은 허용 가능한 위험을 설정하고 생명주기 전반에서 랜덤 고장과 체계적 고장을 관리하는 활동이다.
Chapter 2	위험분석과 보호계층 평가를 통해 각 안전기능의 목표 SIL을 결정한다.
Chapter 5	고장률, 진단, 시험주기, 수리시간, 중복 및 CCF를 반영하여 PFD 또는 PFH를 계산한다.
Chapter 3	SRS, 독립성, SFF, HFT, 통합시험 및 Validation으로 하드웨어와 시스템 적합성을 입증한다.
Chapter 4	V-Model, 제한된 언어, 방어적 코딩, 단계별 시험 및 형상관리로 소프트웨어 체계적 고장을 통제한다.

본 문서는 학습용 요약이며, 세부 적용 시 해당 IEC 61508 원문과 프로젝트 절차를 함께 확인해야 한다.