Software Project Impact Assessment

A Practical Application of Lattice-Theoretic Safety Integrity Classification

Conceptual Framework for Educational Purposes

Based on IEC 61508, ISO 26262, DO-178C, ECSS-E-ST-40C, and IEC 62304

Generated: November 30, 2025

Abstract

This document presents a practical self-assessment methodology for determining appropriate software development constraints based on impact analysis. Using a lattice-theoretic framework that maps harm potential, exposure, and controllability to safety integrity levels, we derive concrete recommendations for architecture, testing, process, and documentation requirements. The methodology constructs explicit functors from heterogeneous safety standards to a universal ordinal space, enabling cross-domain communication while explicitly acknowledging the lossy nature of such mappings.

1. Introduction

The proliferation of software in safety-relevant domains necessitates systematic approaches to determining appropriate development rigor. Industrial standards such as IEC 61508, ISO 26262, and DO-178C provide domain-specific guidance, yet practitioners often require cross-domain comparison and initial scoping before formal hazard analysis.

This assessment framework addresses two fundamental questions:

> **Assessment Questions**
>
> **Q1:** Given the potential impact of software failure, what integrity level is required?
>
> **Q2:** What concrete constraints on architecture, testing, and process follow from that level?

2. Assessment Methodology

The assessment proceeds through four dimensions, each contributing to the overall risk profile. The method is inspired by the ISO 26262 ASIL determination matrix (Severity × Exposure × Controllability) generalized to arbitrary domains.

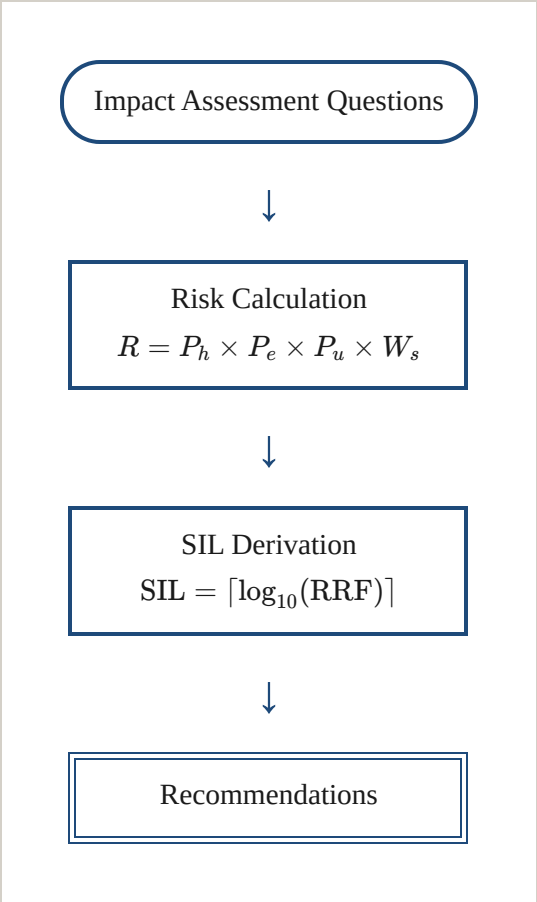

Figure 1: *Assessment workflow from questions to recommendations*

2.1 Input Dimensions

Dimension	Question	Scale
Harm Potential	What is the worst-case outcome if the software fails completely?	None → Catastrophic (10 levels)
Exposure Frequency	How often are users/systems exposed to the hazard?	Rare → Continuous (5 levels)
Controllability	Can users prevent harm if the software fails?	Fully controllable → Uncontrollable (5 levels)
Operational Context	What is the deployment context?	Personal → Critical infrastructure (6 levels)

Table 1: The four assessment dimensions

3. Risk Classification Matrix

The combination of severity and controllability determines the base integrity requirement, modulated by exposure frequency and operational context.

	C1	C2	C3	C4	C5
S1 (Minor)	QM	QM	SIL 1	SIL 1	SIL 1
S2 (Moderate)	QM	SIL 1	SIL 1	SIL 2	SIL 2
S3 (Serious)	SIL 1	SIL 1	SIL 2	SIL 3	SIL 3
S4 (Severe)	SIL 1	SIL 2	SIL 3	SIL 3	SIL 4
S5 (Catastrophic)	SIL 2	SIL 3	SIL 3	SIL 4	SIL 4

S = Severity (rows), C = Controllability (columns, C1=fully controllable, C5=uncontrollable)

Figure 2: *Simplified risk classification matrix (cf. ISO 26262 ASIL determination)*

4. Example Assessments

We present six representative project types spanning the integrity level spectrum, demonstrating how the assessment framework derives concrete constraints.

📋 Internal Business Tool

Harm Potential: None
Exposure: Frequent
Controllability: Fully controllable
Context: Enterprise
Result: `SIL 1`

🌐 Public Web Application

Harm Potential: Financial loss
Exposure: Frequent
Controllability: Mostly controllable
Context: Public-facing
Result: `SIL 2`

🏠 Smart Home Door Lock

Harm Potential: Property damage
Exposure: Frequent
Controllability: Mostly controllable
Context: Personal use
Result: `SIL 2`

🔌 IoT Actuator Device

Harm Potential: Minor injury
Exposure: Common
Controllability: Sometimes controllable
Context: Personal use
Result: `SIL 3`

🏥 Medical Monitoring Device

Harm Potential: Serious injury
Exposure: Continuous
Controllability: Rarely controllable
Context: Life-sustaining
Result: `SIL 4` → IEC 62304 Class C

🚗 Automotive ADAS System

Harm Potential: Multiple fatalities
Exposure: Frequent
Controllability: Sometimes controllable
Context: Public-facing
Result: `SIL 4` → ISO 26262 ASIL D

5. Derived Constraints by Integrity Level

Each integrity level implies specific constraints across four domains: architecture, testing, development process, and documentation. The constraints form a monotonically increasing sequence.

5.1 Architecture Constraints

Level	HFT	Pattern	Key Techniques
SIL 0	0	Standard architecture	Modular design
SIL 1	0	Defensive programming	Input validation, error handling, watchdogs
SIL 2	0	Fail-safe with diagnostics	Assertions, memory protection, plausibility checks
SIL 3	1	Redundant (1oo2 or 2oo3)	Hot standby, diverse algorithms, safety kernel
SIL 4	2	Diverse redundant with voting	N-version programming, formal verification, hardware interlocks

Table 2: Architecture constraints by integrity level (HFT = Hardware Fault Tolerance)

5.2 Testing Constraints

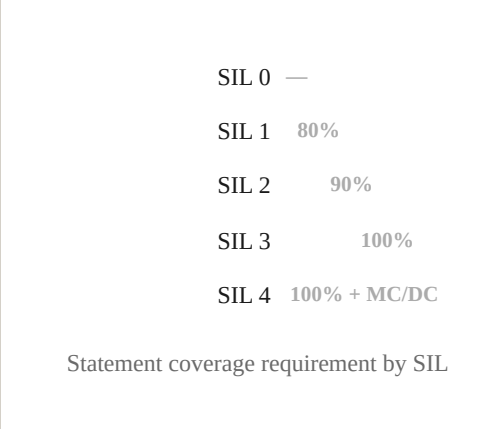

Figure 3: *Code coverage requirements scale with integrity level*

Level	Coverage	Independence	Additional Test Types
SIL 1	Statement ≥80%	Self	Boundary value, equivalence partitioning
SIL 2	Stmt ≥90%, Branch ≥80%	Team	Static analysis, code review, mutation testing

`SIL 3`	Stmt 100%, Branch 100%, MC/DC ≥80%	Independent	Formal inspection, fault injection, stress testing
`SIL 4`	All metrics 100%	Third-party	Formal verification, exhaustive testing, theorem proving

Table 3: Testing requirements by integrity level

6. Detailed Case Study: Industrial Robot Controller

Consider a software controller for an industrial welding robot operating in a factory environment. We perform a complete assessment:

Input Parameters

- **Harm Potential:** Serious injury (robot arm can injure workers)
- **Exposure:** Common (workers near robot 10-50% of operating time)
- **Controllability:** Sometimes controllable (E-stop available but not always reachable)
- **Context:** Critical infrastructure (factory production line)

Derived Risk Metrics

- **Unreduced Risk:** 1.5×10^{0}
- **Tolerable Risk:** 1.0×10^{-6}
- **Risk Reduction Factor:** 1.5×10^{6}
- **Required Level:** `SIL 4` (IEC 61508)

Recommended Constraints

Architecture:

- Diverse redundant architecture with voting (2oo3)
- Hardware Fault Tolerance: 2
- N-version programming
- Hardware interlocks for safety-critical functions
- Formal verification of safety kernel

Testing:

- 100% statement, branch, and MC/DC coverage
- Third-party independent V&V
- Formal verification with theorem provers
- Model checking of state machines
- Certified tool chains

Process:

- V-model with formal methods
- Third-party assessment
- Formal certification review
- Independent V&V throughout lifecycle

Documentation:

- Safety Case with formal safety argument
- Independent Assessment Report
- Certification Evidence Package
- Complete forward/backward traceability

7. Mathematical Foundation

The assessment framework is grounded in lattice theory. Each safety standard's integrity levels form a bounded total order (chain), enabling systematic comparison.

Definition (Safety Integrity Lattice)

A Safety Integrity Lattice is a structure $\mathbb{L} = (L, \bot, \top, \sqcup, \sqcap, \leq)$ where L is the carrier set, $\bot$ is the bottom (no requirements), $\top$ is the top (maximum integrity), and operations satisfy:

$$a \sqcup b = \max(a, b) \quad \text{(join = maximum integrity)}$$
$$a \sqcap b = \min(a, b) \quad \text{(meet = minimum integrity)}$$

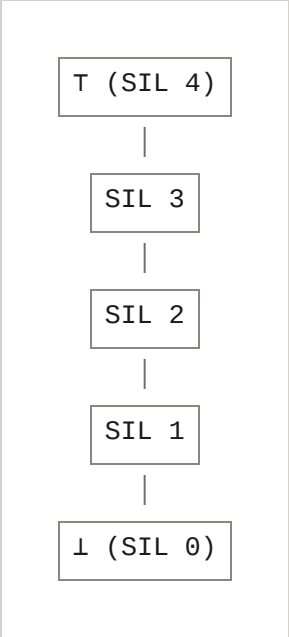

Figure 4: *The IEC 61508 SIL lattice (a total order / chain)*

Proposition (Universal Ordinal Functor)

For each standard $\mathcal{S}_i$ with $|L_i| = k_i$ levels, define $\phi_i : L_i \to [0, 1]$ by:

$$\phi_i(\ell_j) = \frac{j}{k_i - 1}$$

This functor preserves order and lattice operations, enabling cross-standard comparison via ordinal equivalence.

8. Caveats and Limitations

⚠️ **Important Disclaimers**

- This is an **initial scoping tool**, not a substitute for formal hazard analysis.

- Cross-standard mappings are **ordinal only**, not probability equivalence.
- Systems with potential for fatalities require **formal assessment by qualified professionals**.
- Always consult the **actual standards** for compliance requirements.
- Domain-specific standards (ISO 26262, DO-178C, etc.) have their own assessment methods.

The Quotient Semantics

The universal ordinal mapping is *deliberately lossy*. Two hazards with the same ASIL may have arrived via different S×E×C paths with genuinely incomparable risk profiles. This information is quotient'd away in favor of ordinal comparability. The mapping enables cross-standard *communication*, not *substitution*.

9. References

1. IEC 61508:2010 — Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems
2. ISO 26262:2018 — Road Vehicles — Functional Safety
3. DO-178C (RTCA, 2011) — Software Considerations in Airborne Systems and Equipment Certification
4. ECSS-E-ST-40C (ESA, 2009) — Space Engineering: Software
5. IEC 62304:2006/Amd.1:2015 — Medical Device Software — Software Life Cycle Processes
6. AC 25.1309-1A (FAA) — System Design and Analysis
7. Goguen, J. A. & Burstall, R. M. (1992). Institutions: Abstract Model Theory for Specification and Programming. *JACM*, 39(1), 95-146.

This document was generated from the ts-audit safety classification framework.
For interactive assessment, see `examples/assess-project.ts`.