How Automotive Functional Safety really works: *a practical, ISO 26262:2018-Oriented Big Picture*

Version 1.0 ©Copyright by Yingtao WANG, Feb. 2026, Germany contact: Yingtao.Wang.de@gmail.com

Mental Level (describes the abstraction perspective) / FuSa Lifecycle (describes the development sequence)

Mental Level	Step Name	Core Question/Content	Output of the Tasks	ISO26262 Scope
Vehicle Behavior / Hazard Analysis	Step 01 **Item Definition**	What system we talk about	• Item boundary • Operational assumptions (i.e. driver present, speed range etc.)	part03, part04
	Step 02 **HARA & ASIL**	What can go wrong	• Hazard list • Operational situations	part03
Safety Intent	Step 03 **Safety Goals**	What must never happen	• Safety goals • ASIL classification	part03
Functional Concept	Step 04 **Safe States & FTTI**	What 'safe' means within which 'timing'	• Defined safe states and transition conditions • FTTI per safety goal • Detection & Reaction Time Budget	part04
	Step 05 **Functional Safety Concept (FSC)**	How do we know safety is lost and what do we do when it is? / Detection & reaction strategy (architecture independent)	• Detection strategy • Reaction strategy • Safe state transition logic	part04, part09
Technical Realization	Step 06 **Technical Safety Concept (TSC)**	What concrete mechanisms realize the strategy? / Responsibilities, architecture, mechanisms	• Safety responsibilities who should do what? • Architectural constraints • Safety mechanisms	part05 (HW), part06 (SW)
	Step 07 **Implementation**	HW Creation / SW Coding, Configuration	• Real Implementation in HW • Real Implementation in SW	part05 (HW), part06 (SW)
Evidence, Argument & Confidence	Step 08 **Verification & Validation**	How do we know it actually work? / Evidence	• Test Reports • Coverage arguments	part05 (HW), part06 (SW), part04
	Step 09 **Safety Case**	Why should an assessor believe this is safe? / Argument(s)	• Safety Case • GSN-Style arguments	part02
	Step 10 **Assessment (Part 10 view)**	Confidence	• report from checking tasks on Independence plausibility, completeness & confidence	part02, part10

Clarify the 'FuSa Lifecycle' with an example (SW): Brake + Steering + ADAS (Vehicle Level)

Step Name	Brake System	Steering System	ADAS
Step 01 **Item Definition**	Item: *Vehicle longitudinal and lateral motion control influenced by electronic systems.* Included Brake system (ESC / iBooster / EBB) Steering system (EPS / Steer-by-Wire) ADAS functions influencing brake/steer (AEB, LKA, ACC) Excluded Mechanical fallback (assumed available) Key assumptions Driver present Vehicle speed > 0 ADAS is **assistive**, not autonomous **Important insight:** ADAS is *inside the item*, but **not trusted for safety execution**		
Step 02 **HARA & ASIL**	Unintended braking Loss of braking Excessive braking	Unintended steering Loss of steering assist Steering in wrong direction	Braking or steering without driver intent Driver misled about system availability Late or missing takeover request
Step 03 **Safety Goals**	SG-B1: *Unintended braking shall be prevented* (ASIL D) SG-B2: *Loss of braking shall be detected and mitigated* (ASIL D)	SG-S1: *Unintended steering shall be prevented* (ASIL D) SG-S2: *Loss of steering assist shall be detected and mitigated* (ASIL D)	SG-A1: *ADAS shall not cause unintended brake or steering actuation* (ASIL B–C) SG-A2: *Driver shall be informed when ADAS control is no longer reliable* (ASIL B)
Step 04 **Safe States & FTTI**	Brake safe states Controlled deceleration Brake torque limited to driver input only Brake FTTI **FTTI ≈ 10–100 ms** Physics-limited No human compensation possible	Steering safe states Steering torque limited or disabled Mechanical fallback to driver Steering FTTI FTTI ≈ **10–50 ms** Lane departure happens fast	ADAS safe states Function deactivation Clear driver takeover request No autonomous actuation ADAS FTTI FTTI ≈ **hundreds of ms to seconds** Driver can compensate
Step 05 **Functional Safety Concept (FSC)**	Brake / Steering detection Loss of execution Loss of timing Signal corruption Plausibility violations Brake / Steering reaction Local safe state activation Torque limitation Redundant path usage		ADAS detection Software health Sensor inconsistency Deadline misses ADAS reaction Degrade Disable Inform driver
Step 06 **Technical Safety Concept (TSC)**	Execution monitoring Timing supervision Output plausibility Independent watchdog paths		Health supervision Deadline supervision Semantic checks Driver monitoring
Step 07 **Implementation**	Illustrated with two different mechanisms: AUTOSAR and Non-AUTOSAR, pls. refer to the diagram on the right side		
Step 08 **Verification & Validation**	Fault injection Timing violation tests Independence verification		Failure injection Degradation tests Driver warning latency
Step 09 **Safety Case**	GSN-Style Arguments on vehicle level (Brake+Steering+ADAS), pls. refer to the GSN-Tree on the right-bottom		
Step 10 **Assessment**	Checking the Independence plausibility, completeness & confidence on vehicle level (Brake+Steering+ADAS)		

Vehicle-Level Safety (Invariant Truth)

```
                 Vehicle-Level Safety (Invariant Truth)
                                ||
        ________________________________________________
        ||                      ||                     ||
 Brake ECU (ASIL D)    Steering ECU (ASIL D)       ADAS ECU
        ||                      ||                     ||
 Owns brake safety     Owns steering safety       Requests only
  local safe state       local safe state     may fail arbitrarily
```

Brake & Steering ASIL-D AUTOSAR CLASSIC (CP)

Architecture
- Dedicated Brake ECU
- Dedicated Steering ECU
- AUTOSAR OS (OSEK-like)
- ASIL D configuration

Safety mechanisms
- WdgM
  - Alive supervision
  - Deadline supervision
  - Logical supervision
- Internal HW watchdog
- End-to-end protection
- Output plausibility

Reaction
- Local safe state:
  - Torque limitation
  - Controlled deceleration
- No dependency on ADAS

Brake & Steering ASIL-D Non-AUTOSAR

Architecture
- Dedicated Brake ECU
- Dedicated Steering ECU
- Small RTOS or bare metal
- ASIL-D certified

Safety mechanisms
- HW watchdog
- Time-triggered main loop
- Cross-monitoring tasks
- Output range & rate checks

Reaction
- Local safe state
- Same vehicle behavior as AUTOSAR

ADAS ECU (QM / ASIL B-C) AUTOSAR ADAPTIVE (AP)

Architecture
- POSIX OS
- Adaptive applications
- Service-oriented

Safety supervision
- PHM (Platform Health Management)
- SHM (Software Health Management)
- EM (Execution Manager)
- SM (State Manager)

What is monitored
- Process alive
- Deadline
- Memory
- Resource usage
- Functional correctness

Reaction
- Restart app
- Degrade function
- Notify driver
- Inform vehicle safety state

Not allowed
- Trigger brake or steering safe state

ADAS ECU (QM / ASIL B-C) Non-AUTOSAR

Architecture
- Linux or QNX
- Containers / processes
- Central compute

Safety supervision
- Safety Supervisor daemon
- Process heartbeats
- Deadline monitoring
- Semantic checks
- Watchdog to chassis ECUs

Reaction
- Kill / restart process
- Disable ADAS feature
- Driver takeover request
- Maintain chassis independence

Not allowed
- Trigger brake or steering safe state

ADAS → Chassis Interaction AUTOSAR

ADAS sends requests

Requests are:
- Plausibility-checked
- Time-limited
- Ignorable

Brake/steer ECU decides

Implemented via:
- RTE
- Interfaces with E2E protection
- Safety constraints in Classic ECU

ADAS → Chassis Interaction Non-AUTOSAR

ADAS sends requests over:
- CAN / Ethernet

Chassis ECUs:
- Validate
- Rate-limit
- Ignore if invalid

Same rule:
- Chassis ECUs are sovereign

AUTOSAR Coding Example Brake-Modeling (partial)

```
Application (ASIL-D)
  ├─ BrakeCalc
  ├─ SlipControl
  ├─ ActuatorControl
  ↓
AUTOSAR WdgM
  ↓
WdgIf
  ↓
Hardware Watchdog (MCU)

SupervisedEntity Def.:
  SupervisedEntity: SE_BrakeControl
  ASIL: D
  InitialMode: WDGM_NORMAL
AliveSupervision:
  SupervisedEntity: SE_BrakeControl
  Checkpoint: CP_BC
  ExpectedIndications: 1
  MinMargin: 0
  MaxMargin: 0
  ReferenceCycle: 1 ms
Meaning:
  BrakeCalc must run exactly once per 1 ms
  Missing or extra execution → fault

DeadlineSupervision:
  StartCheckpoint: CP_WS_START
  EndCheckpoint:   CP_WS_END
  MinDeadline: 50 µs
  MaxDeadline: 200 µs
Detects:
  CPU overload
  Endless loop
  Cache/memory stalls

LogicalSupervision:
AllowedSequence:
  CP_WS_END →         //WheelSpeedTask end
  CP_BC →             //BrakeCalcRunnable
  CP_SC →             //SlipControlRunnable
  CP_ACT              //ActuatorControlTask
Detects:
  Skipped calculations
  Corrupted control flow
  Memory overwrite effects

Global Reaction Config (ASIL-D):
If violation:
  First failure → internal error counter
  Repeated failure → STOP watchdog servicing
  → Hardware watchdog reset within 5 ms
```

Non-AUTOSAR Coding Example Supervisor Implementation (partial)

```
Brake Tasks (ASIL-D)
  ├─ WheelSpeedTask (1 ms)
  ├─ BrakeCalcTask (1 ms)
  ├─ SlipControlTask (1 ms)
  ├─ ActuatorTask (1 ms)
  ↓
Safety Supervisor (ASIL-D, trusted)
  ├─ Execution monitoring
  ├─ Timing monitoring
  ├─ Sequence monitoring
  ├─ Health state machine
  ↓
Hardware Watchdog

typedef struct {
   uint32_t lastExecTick;
   uint32_t minPeriod;
   uint32_t maxPeriod;
   uint32_t maxExecTime;
   uint32_t startTick;
   bool     execActive;
   uint8_t  expectedSequenceId;
} SafetyTaskMonitor;

// Alive Monitoring
void Safety_ReportAlive(TaskId id)
{
   monitors[id].lastExecTick = GetSystemTick();
}

// Deadline Monitoring
void Safety_ReportStart(TaskId id)
{
   monitors[id].startTick = GetSystemTick();
   monitors[id].execActive = true;
}
void Safety_ReportEnd(TaskId id)
{
   uint32_t execTime = GetSystemTick() - monitors[id].startTick;
   monitors[id].execActive = false;

   if (execTime > monitors[id].maxExecTime)
      RaiseFault(DEADLINE_VIOLATION);
}

// Logical Monitoring
void Safety_ReportSequence(TaskId id, uint8_t sequenceId)
{
   if (sequenceId != monitors[id].expectedSequenceId)
      RaiseFault(SEQUENCE_ERROR);

   monitors[id].expectedSequenceId++;
}

//Global Health Decision
void SafetySupervisor_Main(void)
{
   if (anyCriticalFaultDetected)
   {
      // Do NOT kick watchdog
      RequestSafeState();
   }
   else
   {
      KickHardwareWatchdog();
   }
}
```

How to Read This Big Picture

Purpose
This document explains how Automotive Functional Safety really works, from hazards to safe vehicle behavior, based on ISO 26262 principles.

1. Mental Model (What & Why)
Start here.
It defines hazards, safety goals, safe states, and timing — independent of implementation.

2. FuSa Lifecycle (When)
Follow the 10 steps top-down.
Each step answers one safety question and maps directly to ISO 26262 work products.

3. System Example (How)
The Brake + Steering + ADAS example shows how concepts are applied in real systems.
AUTOSAR and non-AUTOSAR architectures realize the same safety intent.

4. GSN (Why believable)
The GSN tree structures the safety argument and supporting evidence on vehicle level.

5. Vocabulary & Roles
Terms and roles are used as defined here to avoid misunderstandings.

What this is NOT
Not a checklist, not a process manual, not a standard replacement.

Key Question to Remember
What is the hazard, what is the safe state, and who enforces it — within the allowed time?

Must-know FuSa Vocabulary

ISO 26262
Automotive functional safety standard addressing risks caused by systematic and random hardware failures in E/E systems.

Safety Item
A vehicle-level function or system under safety consideration, including its interactions and boundaries.

Hazard
A potential source of harm caused by malfunctioning behavior of the item.

Safety Goal (SG)
A top-level safety requirement defined to prevent or mitigate a hazardous event.

Safe State
A system state that eliminates or sufficiently reduces the risk associated with a hazard.

FTTI (Fault Tolerant Time Interval)
Maximum allowed time between fault occurrence and reaching the safe state.

ASIL (Automotive Safety Integrity Level)
A risk classification defining the required rigor of safety measures.
Derived from the risk matrix: S x E x C
S: Severity of harm / E: Exposure probability / C: Controllability by driver

QM (Quality Managed)
Function without unreasonable risk requiring ISO 26262 safety measures.

Functional Safety Concept (FSC)
Technology-independent definition of fault detection and reaction strategies to reach safe states.

Technical Safety Concept (TSC)
Concrete realization of the FSC through architecture, responsibility allocation, and safety mechanisms.

Freedom from Interference (FFI)
Assurance that one element does not adversely affect the safety of another.

ASIL Decomposition
Structured partitioning of a safety requirement into multiple elements with lower ASIL, while preserving safety.

Safety Case
A structured argument, supported by evidence, demonstrating that safety goals are achieved.

Advanced FuSa Vocabulary

GSN (Goal Structuring Notation)
Graphical or textual notation to express safety arguments explicitly.

Item out of Context (IoC)
Development of a safety element without full knowledge of its final vehicle integration.

Confirmation Measures
Independent reviews, audits, and assessments ensuring correctness and completeness of safety activities.

ISO 21448 – SOTIF
Addresses hazards caused by functional insufficiencies or performance limitations, not failures.

ISO 21434 – Cybersecurity
Addresses risks caused by malicious attacks, not accidental failures.

FuSa Roles

Functional Safety Manager (FSM / FuSa Manager)
Owns the functional safety process and ensures ISO 26262 compliance across the lifecycle.
Key resp.: define safety plan / ensure lifecycle completeness / coordinate confirmation measures / interface with assessor

Safety Requirements Engineer
Derives, manages, and traces safety requirements from HARA downwards.
Key resp.: safety goals / functional & technical safety requirements / traceability across lifecycle

Functional Safety Concept Designer (System Level)
Defines the detection and reaction strategies to reach safe states.
Key resp.: safe states / FTTI / FSC definition

Technical Safety Architect (HW / SW)
Realizes the FSC through architecture and responsibility allocation.
Key resp.: TSC / safety mechanisms / ASIL decomposition / freedom from interference

Functional Safety Developer (HW / SW)
Implements safety mechanisms according to the TSC.
Key resp.: watchdogs, monitors, diagnostics / fault reactions / timing guarantees

Base Software / Feature Developer
Implements functional behavior not directly related to safety mechanisms.

Safety Verification Engineer / Tester
Verifies that safety requirements and mechanisms behave as intended.
Key resp: fault injection / timing verification / requirement-based testing

Safety Case Engineer
Builds and maintains the structured safety argument and evidence mapping.
Key resp: GSN development / evidence consistency / argument completeness

Functional Safety Auditor
Verifies compliance of processes and work products with ISO 26262. Focus: process adherence

Functional Safety Assessor
Judges whether the safety concept and evidence are sufficient to claim safety.
Focus: safety sufficiency, not checklist compliance; Typically independent

GSN for the example Brake+Steering+ADAS

C0 – Top Claim
Vehicle motion control (braking and steering) is acceptably safe in the presence of ADAS functions.

S1 – Decomposition Strategy
Argue safety by decomposing vehicle motion control into independent functional safety goals for:
braking
steering
ADAS command arbitration
(ISO 26262 principle: responsibility separation & freedom from interference)

C1 – Braking Function is Safe (ASIL D)
No unintended braking
No loss of braking capability
Defined brake safe state reached within FTTI
Brake ECU has final authority

C2 – Steering Function is Safe (ASIL D)
No unintended steering torque
Driver override always possible
Steering torque reduced to safe state within FTTI
Steering ECU has final authority

C3 – ADAS Arbitration is Safe
ADAS never has final actuation authority
Brake and Steering ECUs arbitrate and validate ADAS requests
ADAS failures are detected, isolated, and lead to command suppression

Context (implicit, but important)
Operational Design Domain defined
Driver available (not fully autonomous)
Mechanical fallback exists
This textual GSN is what assessors love during reviews.