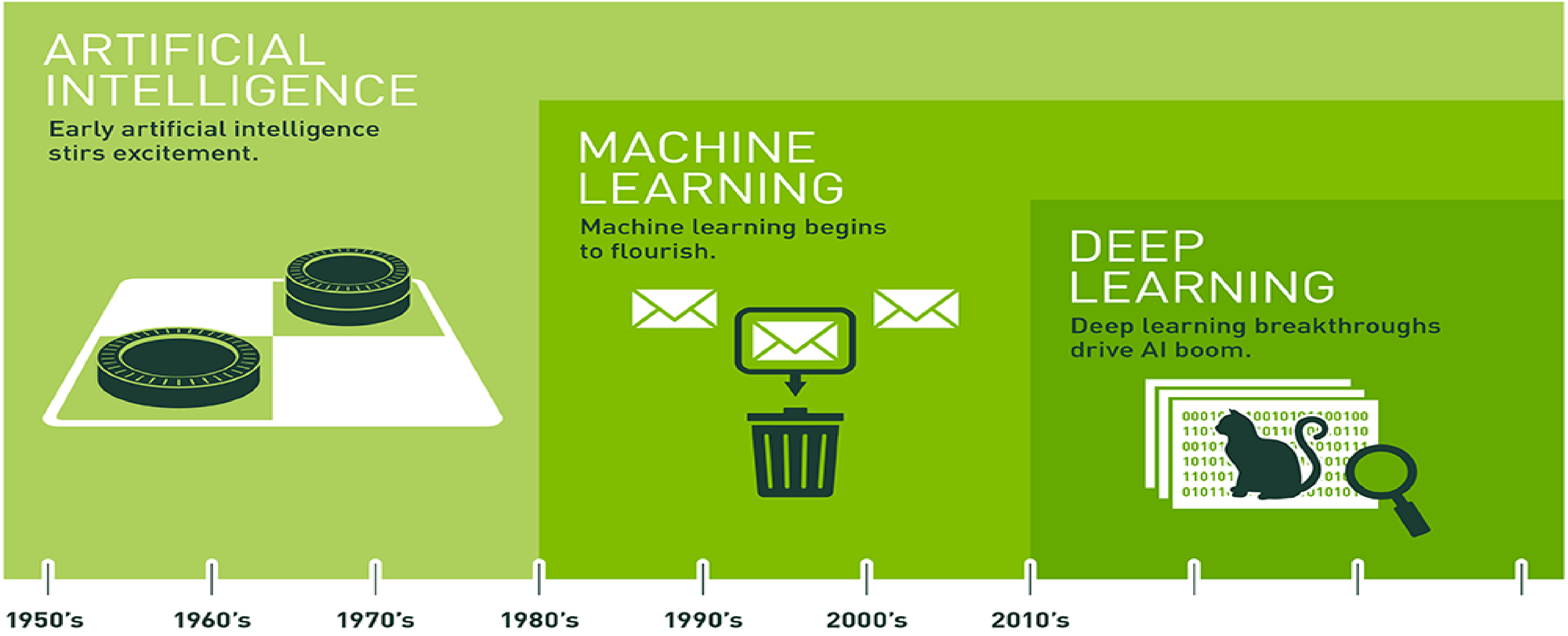
CHALLENGES IN AI/ML FOR SAFETY CRITICAL SYSTEMS

RICCARDO MARIANI | VP, INDUSTRY SAFETY | OCTOBER 3, 2019

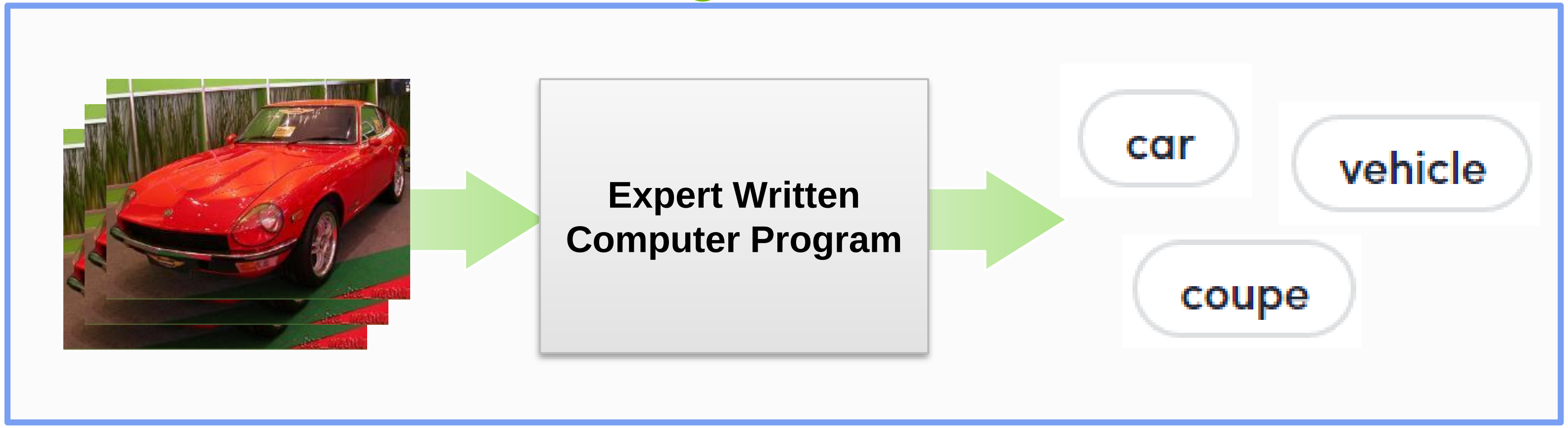
A BRIEF HISTORY OF AI



NVIDIA

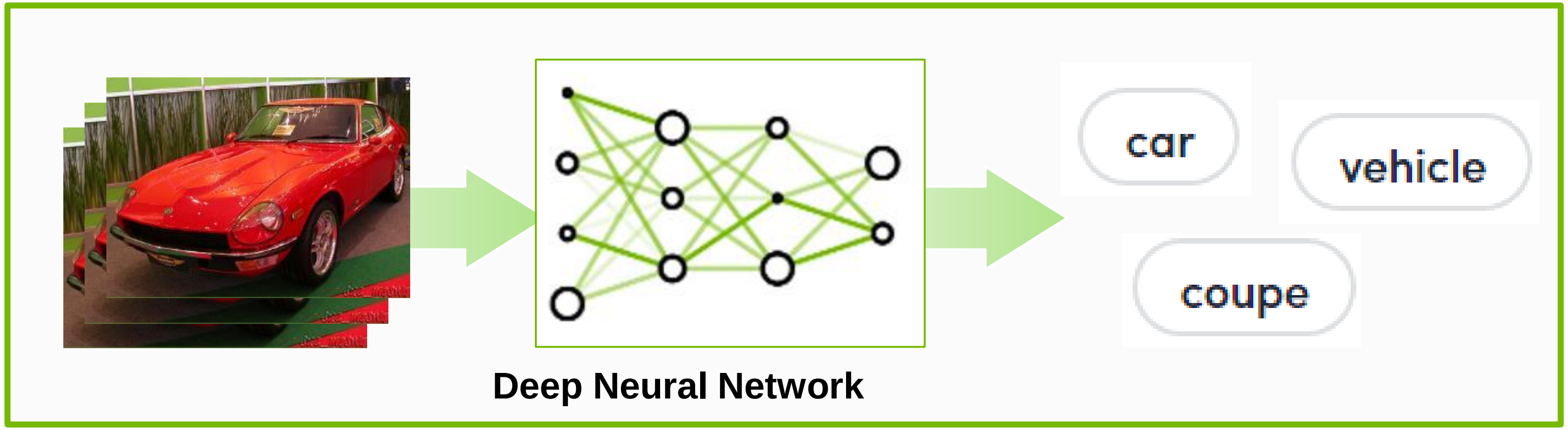
WHAT IS A DEEP NEURAL NETWORK (DNN)

An Algorithm that Learns from Data



Traditional Approach

- ➢ Requires domain experts
- ➢ Time consuming
- ➢ Error prone
- ➢ Not scalable to new problems

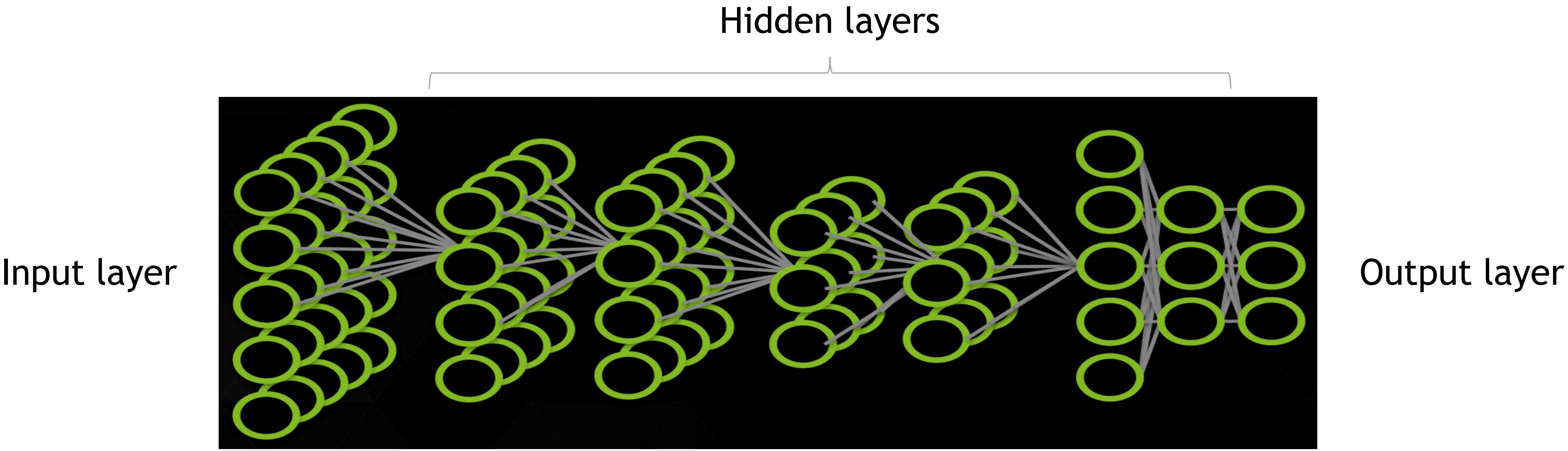


Deep Learning Approach

- ✓ Learn from data
- ✓ Easily extended
- ✓ Often better at complex problems

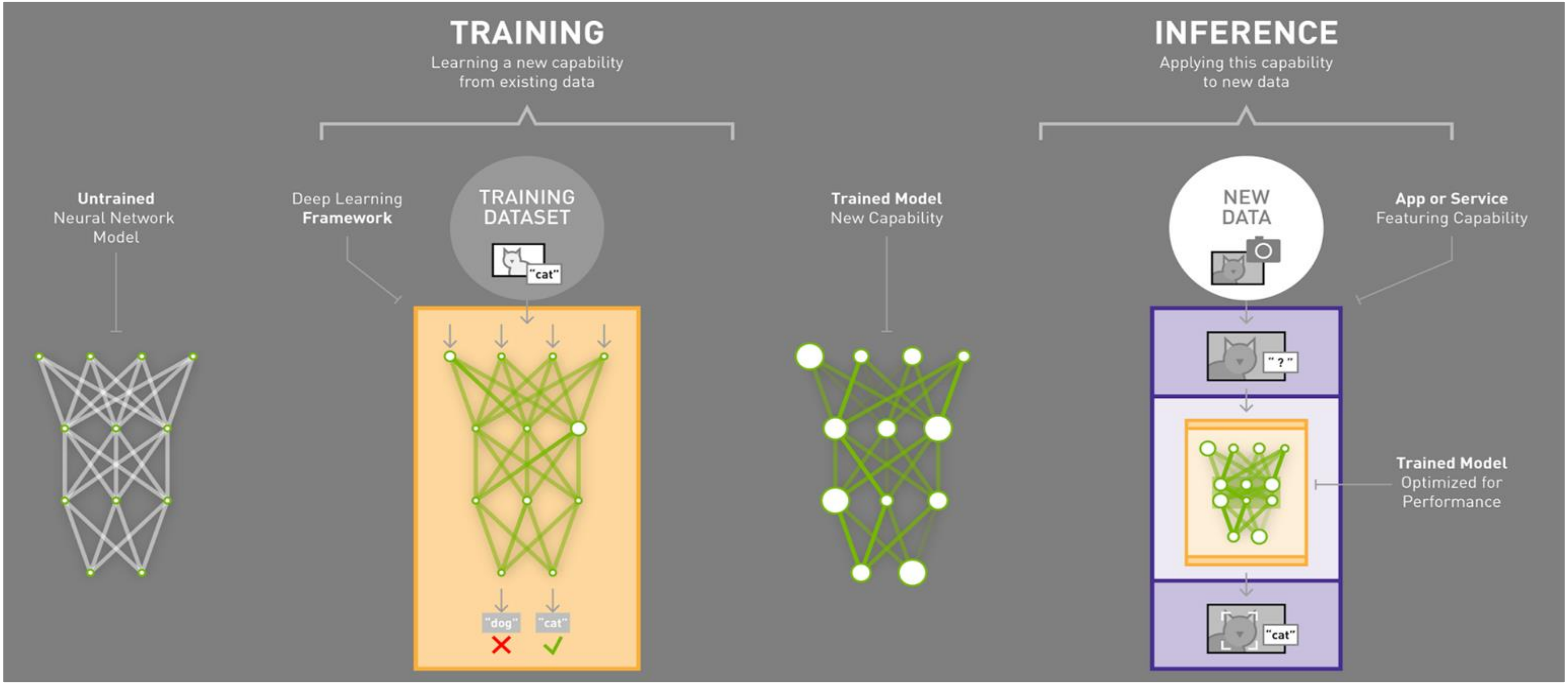
ARTIFICIAL NEURAL NETWORK

A collection of simple, trainable mathematical units that collectively learn complex functions



Given sufficient training data an artificial neural network can approximate very complex functions mapping raw data to output decisions

HOW IT WORKS


TRAINING
Learning a new capability from existing data
INFERENCE
Applying this capability to new data
Untrained Neural Network Model
Deep Learning Framework
TRAINING DATASET
"cat"
"dog"
"cat"
Trained Model New Capability
NEW DATA
App or Service Featuring Capability
" ? "
Trained Model Optimized for Performance
"cat"

SMART MACHINES

Definition

A **smart machine** is a device embedded with:

- **Machine-to-machine (M2M)**
- **Human-to-machine (H2M)**, and
- Cognitive computing technologies such as **artificial intelligence (AI), machine learning (ML)** or **deep learning (DL)**, implemented with **Deep Neural Networks (DNN)**

= all of which it uses to **reason**, **problem-solve**, **make decisions** and ultimately, even **take action.**

EXAMPLES OF SMART MACHINES

Cars

Robotaxis

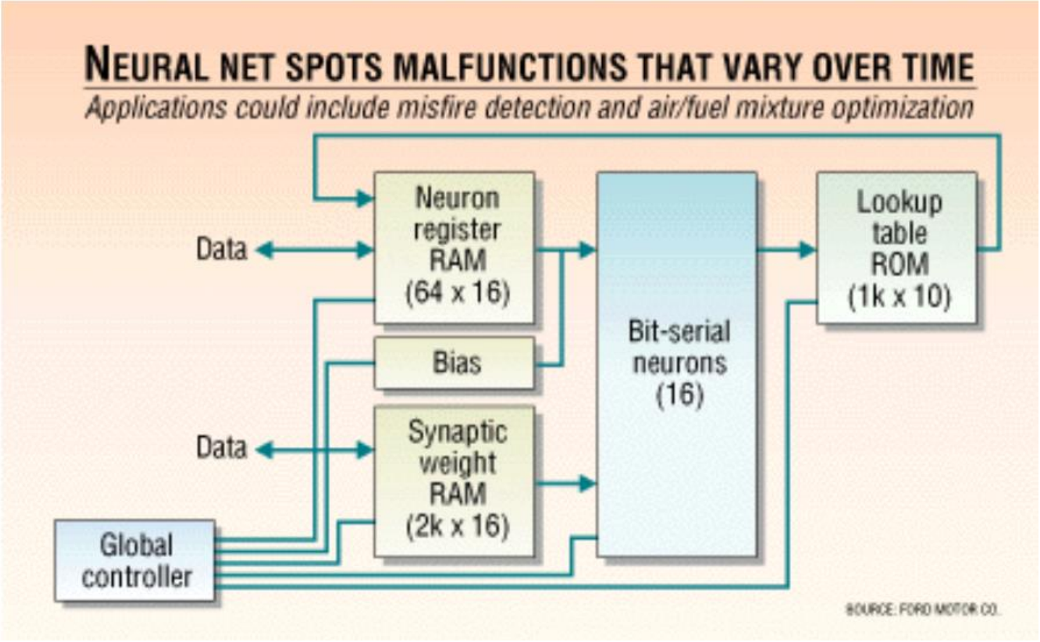
Trucks

Delivery Vans

Buses

Tractors

NEURAL NETWORKS IN AUTOMOTIVE

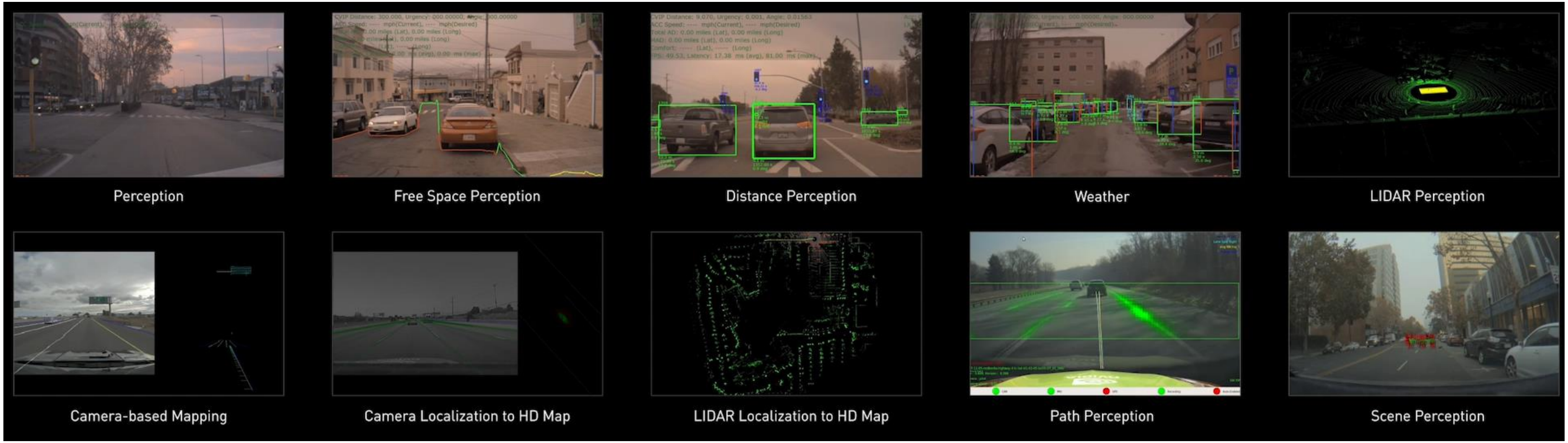


https://www.eetimes.com/document.asp?doc_id=1138005

- Research from early 1990s
- Used for:
  - Misfire detection
  - Air/fuel mixture optimization
  - Fuel canister purge
  - Dynamic suspension control
- Ford licensed neural network IP from JPL in 1998 for powertrain.

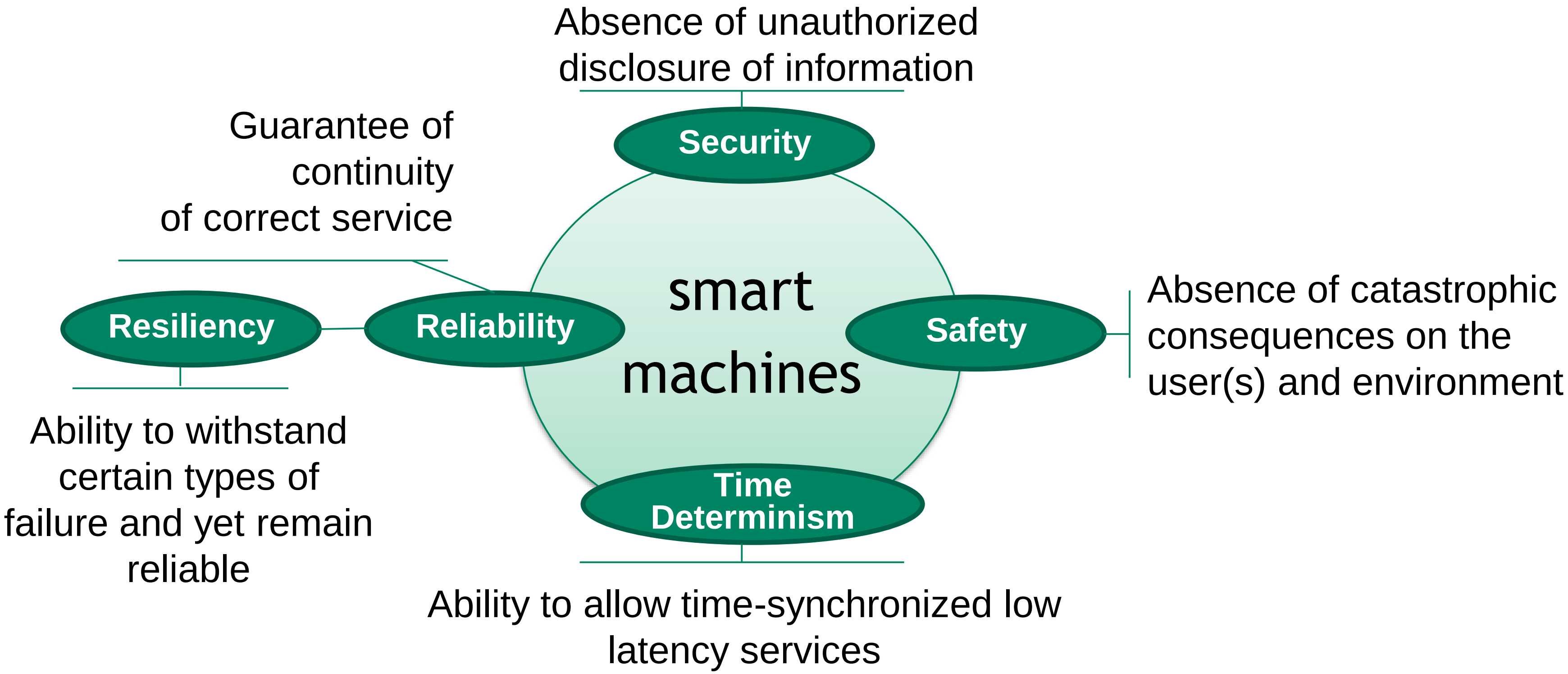
MANY THINGS TO LEARN

SIMULTANEOUS DEEP NEURAL NETWORKS



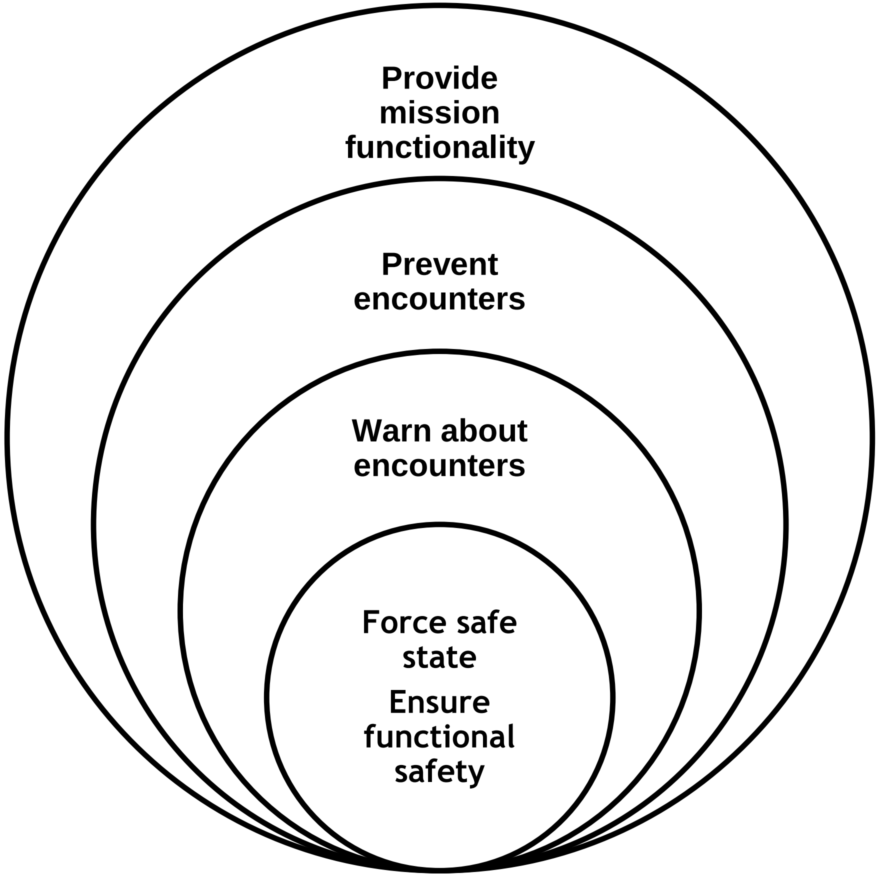
DEPENDABILITY OF SMART MACHINES

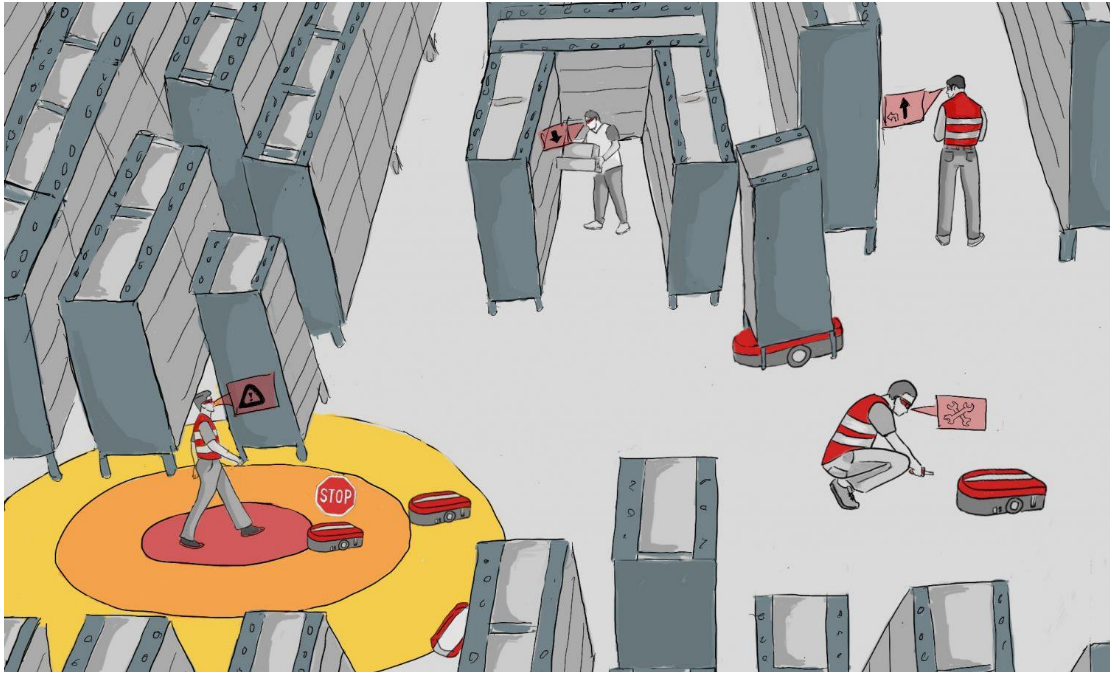
Definitions



SAFETY OF SMART MACHINES

Abstracting Safety in Layers





from SafeLog project [1]

SAFETY OF SMART MACHINES

What we need to avoid or mitigate….

- successful attack exploiting vulnerabilities — Cybersecurity
- impact from surroundings
- reasonably foreseeable misuse, incorrect HMI
- performance limitations

  } Safety of Intended Functionality (SOTIF)
- HW random failures
- systematic failures

  } FuSa

FUNCTIONAL SAFETY (FUSA)

Definition

The absence of ***unreasonable*** risk due to hazards caused by malfunctioning behavior of electric/electronic (E/E) systems

Systematic failures

Bugs in S/W, H/W design and Tools

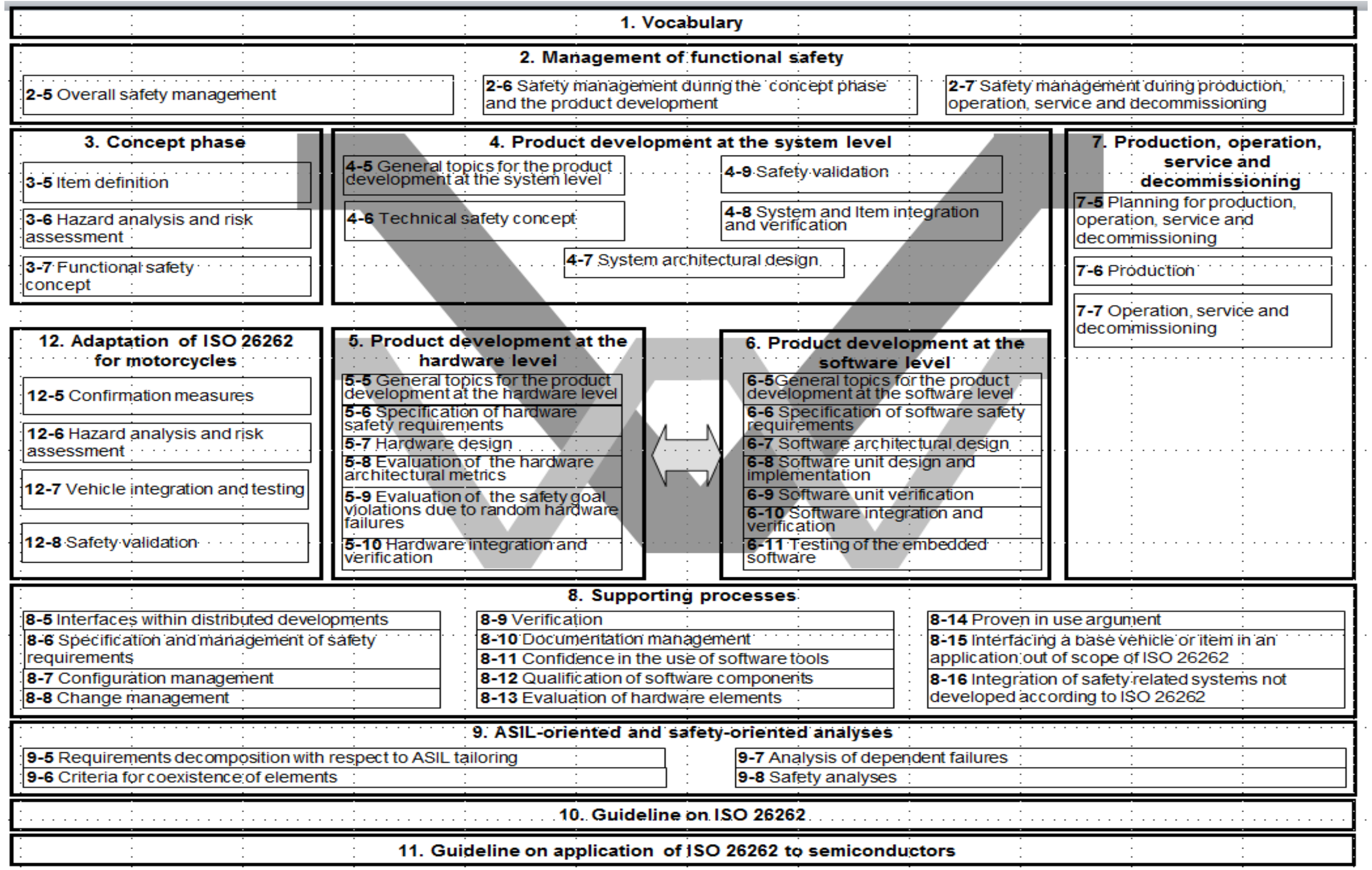
Random H/W failures

Permanent and transient faults occurring while using the system due to aging effects, electromigration, soft errors, …

FUSA INTERNATIONAL STANDARDS

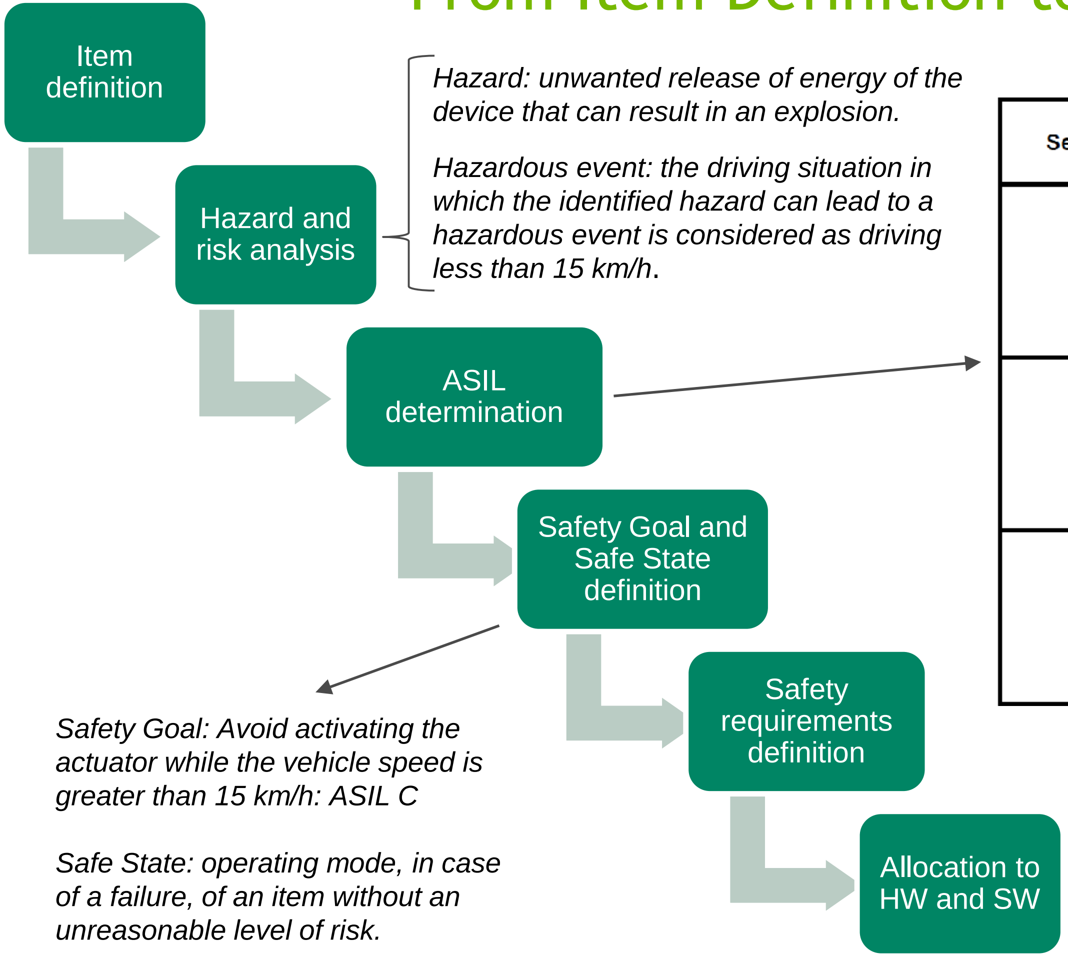
ISO 26262

Source: ISO 26262 2nd edition



FUSA WORKFLOW

From Item Definition to H/W, S/W Requirements



Hazard: unwanted release of energy of the device that can result in an explosion.

Hazardous event: the driving situation in which the identified hazard can lead to a hazardous event is considered as driving less than 15 km/h.

Safety Goal: Avoid activating the actuator while the vehicle speed is greater than 15 km/h: ASIL C

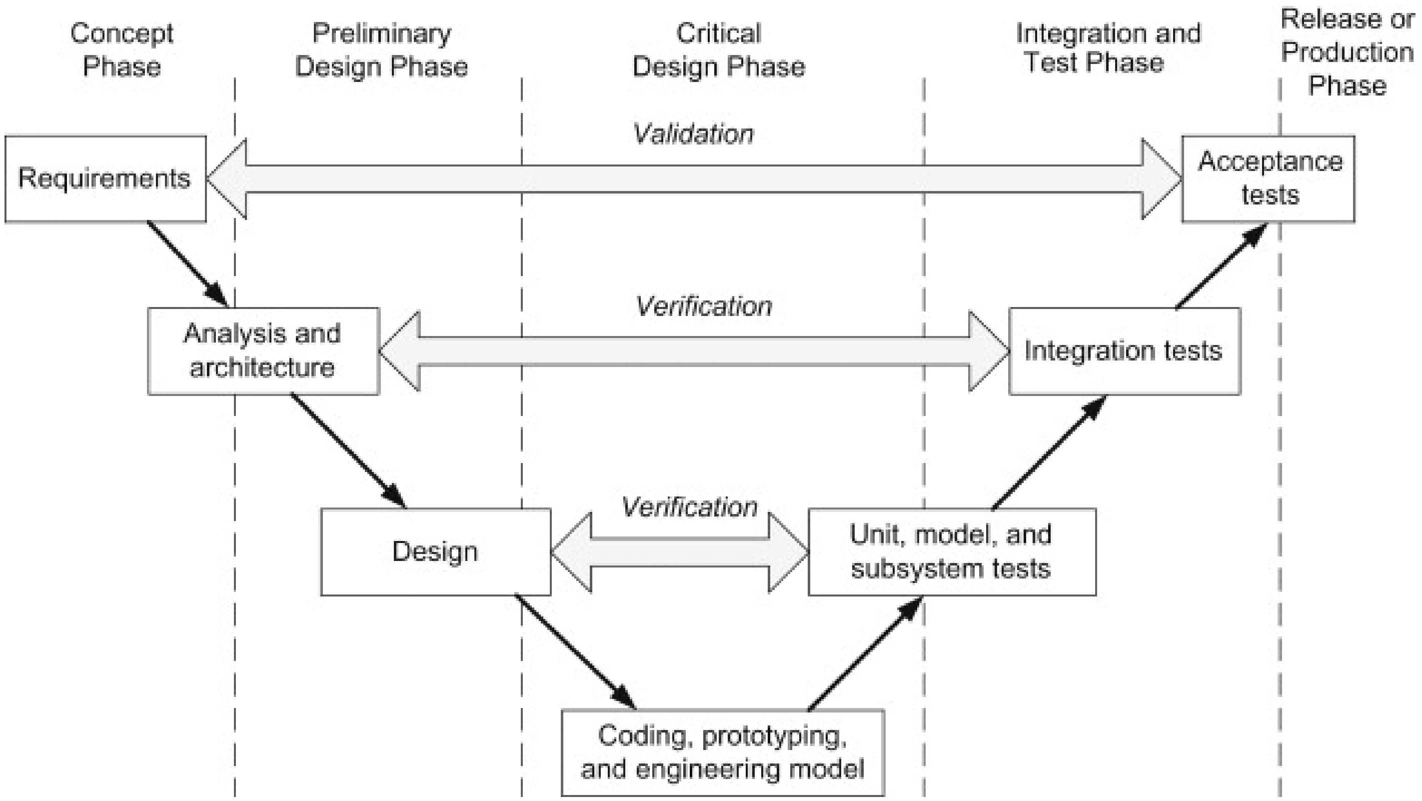
Safe State: operating mode, in case of a failure, of an item without an unreasonable level of risk.

Severity class	Probability class	Controllability class		
		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	A
	E4	QM	A	B
S2	E1	QM	QM	QM
	E2	QM	QM	A
	E3	QM	A	B
	E4	A	B	C
S3	E1	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

Source: ISO 26262 2nd edition

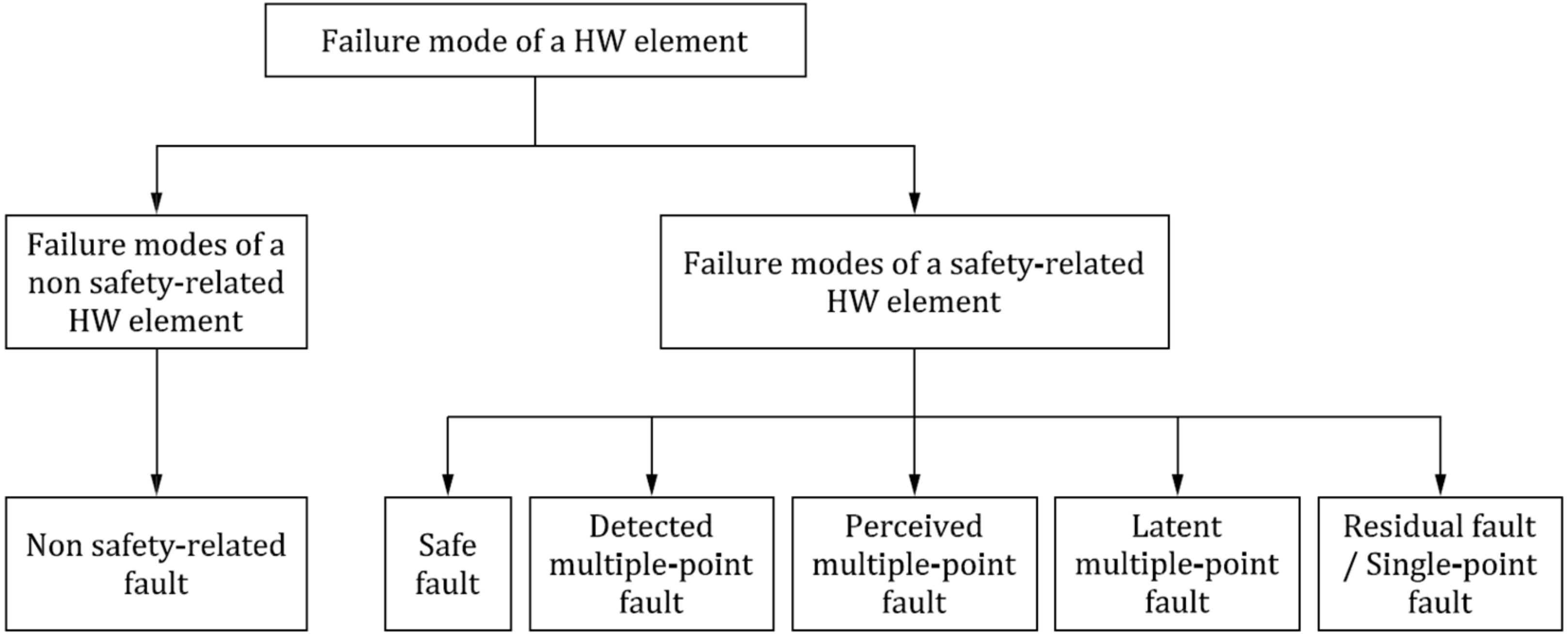
V MODEL

From Requirements to Verification and Validation



HW RANDOM FAILURES

Failures Classification



Source: ISO 26262 2nd edition

ISO 26262 QUANTITATIVE TARGETS

For HW Random Failures

ASIL	SPFM	LFM	PMHF
A	--	--	$< 10^{-6}$
B	$\geq 90\%$	$\geq 60\%$	$< 10^{-7}$
C	$\geq 97\%$	$\geq 80\%$	$< 10^{-7}$
D	$\geq 99\%$	$\geq 90\%$	$< 10^{-8}$ = 10 FIT (1 FIT = 10^{-9})

- **SPFM = Single Point Fault Metric**
  - Robustness of the item to single-point and residual faults either by coverage from safety mechanisms or by design (primarily safe faults).
- **LFM = Latent Fault Metric**
  - Robustness of the item to latent faults either by coverage of faults in safety mechanisms or by the driver recognizing that the fault exists before the violation of the safety goal, or by design (primarily safe faults).
- **PMHF = Probabilistic Metric for random Hardware Failures**
  - Basically the remaining portion of residual and single point failures.

QUANTIFYING RESIDUAL FAILURES

Simplified Formula and Example for a RAM

$$\lambda_{RF} \approx \lambda \times (1 - F_{safe}) \times (1 - K_{RF})$$

so called
"Diagnostic Coverage"

Example:

- RAM failure rate for soft errors = 0.0001 FIT / bit
- 128Mbit RAM failure rate = 128 x 1024 x 1024 x 0.0001 ≅ 13422 FIT
- Assuming 10% unused (F_{safe}= 0.1)
- Assuming SEC-DED ECC (K_{RF} = 0.999)
- Residual failures (soft errors only) = 13422 x 0.9 x 0.001 ≅ 12 FIT

THE FAILURE RATE CHALLENGE

Modern technologies have complex failure mechanisms

- Example from ISO 26262-11 (derived from former IEC/TR 62380):

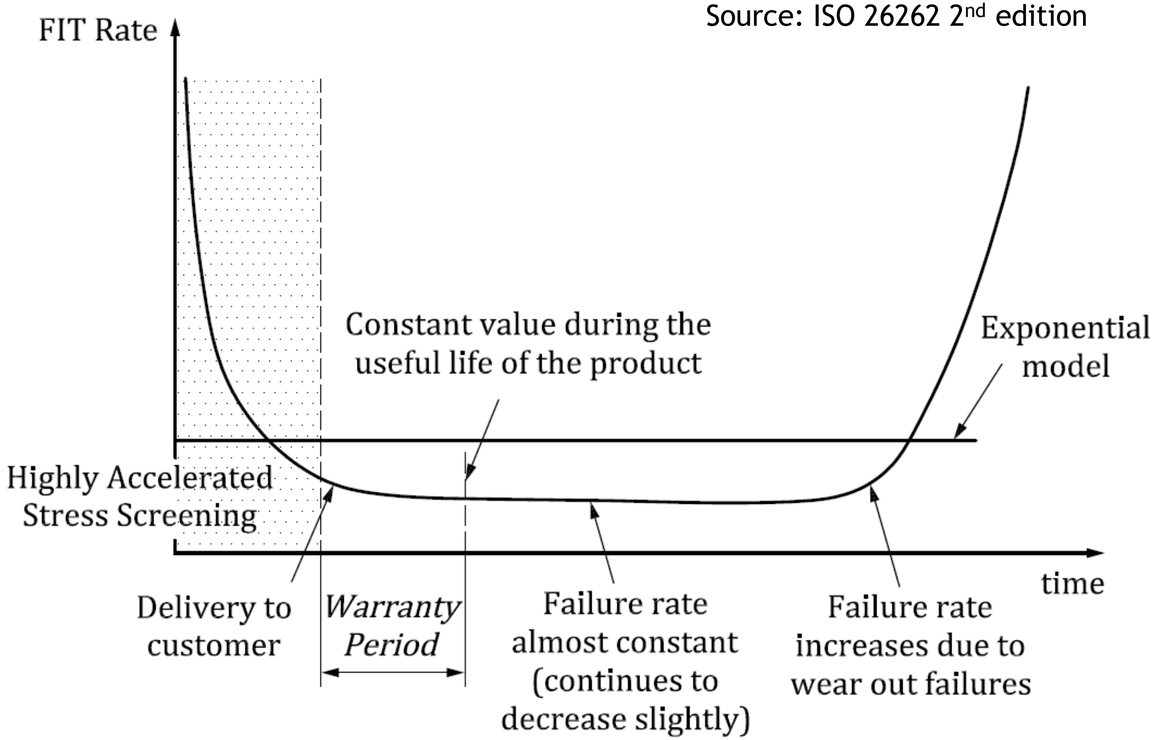
$$\lambda = \left(\underbrace{\left\{ \lambda_1 \times N \times {}^{-0,35\times a} + \lambda_2 \right\} \times \left[\frac{\sum_{i=1}^{y} (\pi_t)_i \times \tau_i}{\tau_{on} + \tau_{off}} \right]}_{\lambda_{die}} + \left\{ \underbrace{2,75\times 10^{-3} \times \pi_\alpha \times \left(\sum_{i=1}^{z} (\pi_n)_i \times (\Delta T_i)^{0,68} \right) \times \lambda_3}_{\lambda_{package}} \right\} + \left\{ \underbrace{\pi_I \times \lambda_{EOS}}_{\lambda_{overstress}} \right\} \right) \times 10^{-9} / h$$

Source: ISO 26262 2nd edition

THE FAILURE RATE CHALLENGE

The end of bath tube reliability curve

Source: ISO 26262 2nd edition



VULNERABILITY FACTORS

Used to Estimate F_{safe}

$$SER^{derated} = \sum_{UCs} F_{UC}(V, f_{clk}) * \sum_{circuits/nodes} SER^{nominal} * TVF * AVF * PVF$$

- **AVF = Architectural Vulnerability Factor**
  - Function of micro-architecture & workload
  - Affects all logic – uArch structures, sequential state, static logic.
- **TVF = Timing Vulnerability Factor**
  - Function of clocking, circuit behavior & workload
  - Affects primarily sequential state.
- **PVF = Program Vulnerability Factor**
  - Function of final user observable program output.

Integer Benchmarks	ACE IPC	ACE Latency (cycles)	# ACE Inst	AVF
bzip2-source	0.55	22	12	19%
cc-200	0.57	18	10	16%
crafty	0.37	15	6	9%
eon-kajiya	0.36	20	7	11%
gap	0.78	17	13	21%
gzip-graphic	0.60	13	8	12%
mcf	0.25	68	17	26%
parser	0.49	24	12	19%
perlbmk-makerand	0.38	17	7	10%
twolf	0.30	27	8	13%
vortex_lendian3	0.42	22	9	15%
vpr-route	0.35	12	4	7%
average	0.45	23	9	15%

Source: Shubhendu S. Mukherjee related works.

QUANTITATIVE ANALYSIS

Combining FMEA/FMEDA with quantitative analysis

Source: ISO 26262 2nd edition

Part	Sub-part	Elementary sub-parts	Safety Related Component ? Not Safety-Related Component ?	Failure modes	Permanent failures: Failure rate (FIT)	Permanent failures: Amount of safe faults (see note 1)	Permanent failures: Safety mechanism(s) preventing the violation of the safety goal	Permanent failures: Failure mode coverage wrt. violation of safety goal	Permanent failures: Residual or Single Point Fault failure rate / FIT	Permanent failures: Safety mechanism(s) preventing latent faults	Permanent failures: Failure mode coverage wrt. Latent failures	Permanent failures: Latent Multiple Point Fault failure rate / FIT	Transient failures: Failure rate (FIT)	Transient failures: Amount of safe faults (see note 1)	Transient failures: Safety mechanism(s) preventing the violation of the safety goal	Transient failures: Failure mode coverage wrt. violation of safety goal	Transient failures: Residual or Single Point Fault failure rate / FIT
Volatile Memory	RAM (16KB)	RAM data bits	SR	permanent fault	1.5000	0%	SM3	96.9%	0.04688	SM3	100%	0.00000					
				transient fault									131.072	0%	SM3	99.69%	0.40894
		Address Decoder	SR	permanent fault	0.0087	0%	none	0%	0.00870								
				transient fault									0.000335	0%	none	0%	0.00034
		Test/redundancy	SR	permanent fault	0.0058	50%	none	0%	0.00290								
				transient fault									0.00033	90%	none	0%	0.00003
				Σ					0.05848			0.00000					0.40931

	Permanent		Transient
Total failure rate	1.51450	Total failure rate	131.07
Total Safety Related	1.51450	Total Safety Related	131.07
Total Not Safety Related	0.00000	Total Not Safety Related	0.00

Single Point Faults Metric 96.1%

Latent Faults Metric 100.0%

Single Point Faults Metric 99.69%

EXAMPLES OF FAILURE MODES

guidelines in ISO 26262-5 and ISO 26262-11

Source: ISO 26262 2nd edition

Element	See tables	Analysed failure modes
Communication		
Data transmission (to be analysed with ISO 26262-6:2018, D.2.4)	D.6 — Communication bus (serial, parallel)	Loss of communication peer Message corruption Message unacceptable delay Message loss Unintended message repetition Incorrect sequencing of messages Message insertion Message masquerading Message incorrect addressing

Source: ISO 26262 2nd edition

Part/subpart	Function	Aspects to be considered for Failure mode[a]
Central Processing Unit (CPU)	Execute given instruction flow according to given Instruction Set Architecture.	CPU_FM1: given instruction flow(s) not executed (total omission) CPU_FM2: un-intended instruction(s) flow executed (commission) CPU_FM3: incorrect instruction flow timing (too early/late) CPU_FM4: incorrect instruction flow result CPU_FM1 can be further refined if necessary into: — CPU_FM1.1: given instruction flow(s) not executed (total omission) due to program counter hang up — CPU_FM1.2: given instruction flow(s) not executed (total omission) due to instruction fetch hang up
CPU Interrupt Handler circuit (CPU_INTH)	Execute interrupt service routine (ISR) according to interrupt request	CPU_INTH_FM1: ISR not executed (omission/too few) CPU_INTH_FM2: un-intended ISR execution (commission/too many) CPU_INTH_FM3: delayed ISR execution (too early/late) CPU_INTH_FM4: incorrect ISR execution (see CPU_FM1/2/4)
CPU Memory Management Unit (CPU_MMU)	The Memory Management Unit (MMU) typically performs two functions: — translates virtual addresses into physical addresses — Controls memory access permissions.	CPU_MMU_FM1: Address translation not executed CPU_MMU_FM2: Address translation when not requested CPU_MMU_FM3: delayed address translation CPU_MMU_FM4: translation with incorrect physical address CPU_MMU_FM5: un-intended blocked access CPU_MMU_FM6: un-intended allowed access CPU_MMU_FM7: delayed access

EXAMPLES OF SAFETY MECHANISMS

guidelines in ISO 26262-5 and ISO 26262-11

Source: ISO 26262 2nd edition

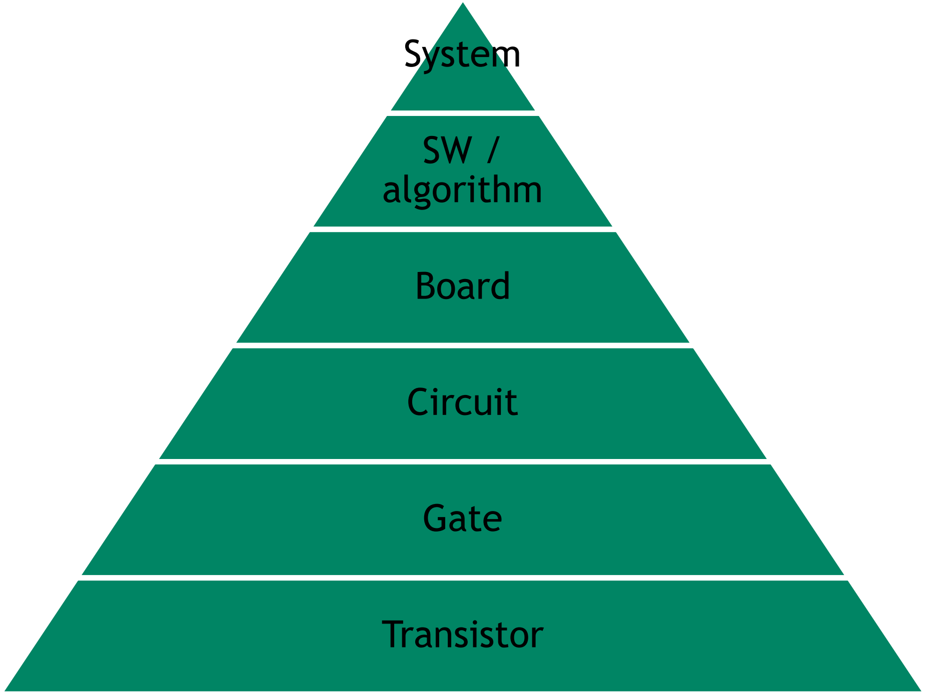
Safety mechanism/ measure	See overview of techniques	Typical diagnostic coverage considered achievable	Notes
One-bit hardware redundancy	D.2.5.1	Low	—
Multi-bit hardware redundancy	D.2.5.2	Medium	—
Read back of sent message	D.2.5.9	Medium	—
Complete hardware redundancy	D.2.5.3	High	Common mode failures can reduce diagnostic coverage
Inspection using test patterns	D.2.5.4	High	—
Transmission redundancy	D.2.5.5	Medium	Depends on type of redundancy. Effective only against transient faults
Information redundancy	D.2.5.6	Medium	Depends on type of redundancy
Frame counter	D.2.5.7	Medium	—
Timeout monitoring	D.2.5.8	Medium	—
Combination of information redundancy, frame counter and timeout monitoring	D.2.5.6, D.2.5.7 and D.2.5.8	High	For systems without hardware redundancy or test patterns, high coverage can be claimed for the combination of these safety mechanisms

Source: ISO 26262 2nd edition

Safety mechanism/ measure	See overview of techniques	Typical diagnostic coverage considered achievable	Notes
Self-test by software: limited number of patterns (one channel)	D.2.3.1	Medium	Depends on the quality of the self-test
Self-test by software cross exchange between two independent units	D.2.3.3	Medium	Depends on the quality of the self-test
Self-test supported by hardware (one-channel)	D.2.3.2	Medium	Depends on the quality of the self-test
Software diversified redundancy (one hardware channel)	D.2.3.4	High	Depends on the quality of the diversification. Common mode failures can reduce diagnostic coverage
Reciprocal comparison by software	D.2.3.5	High	Depends on the quality of the comparison
HW redundancy (e.g. dual core lockstep, asymmetric redundancy, coded processing)	D.2.3.6	High	It depends on the quality of redundancy. Common mode failures can reduce diagnostic coverage
Configuration register test	D.2.3.7	High	Configuration registers only
Stack over/under flow Detection	D.2.3.8	Low	Stack boundary test only
Integrated hardware consistency monitoring	D.2.3.9	High	Coverage for illegal hardware exceptions only

SYSTEM LEVEL VS TRANSISTOR LEVEL

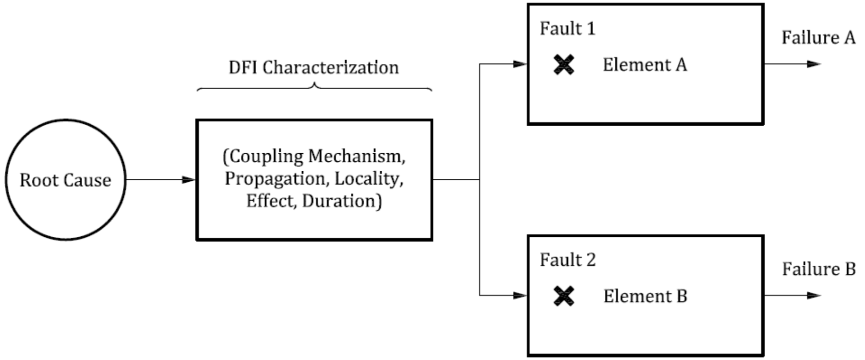
Safety mechanisms: trade-offs and trends



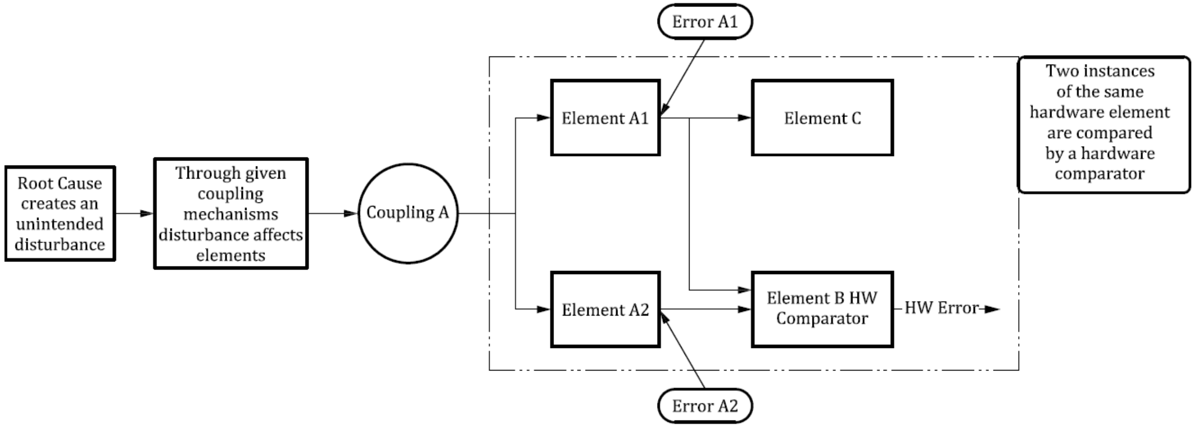
- Industry uses safety mechanisms at different levels
- Complexity of systems and time to market requirements are breaking the pyramid in two areas:
  - Providing an infrastructure at the lowest level (transistor level) to detect (as early as possible) degradation phenomena – e.g. in field self test, network of aging sensors etc.
  - Using those diagnostic information at the SW/algorithm and system level – with the aim of providing detection and control.

DEPENDENT FAILURES

Very difficult to be quantified…. but can be very critical !



Source: ISO 26262 2nd edition



DEPENDENT FAILURES

Avoiding or detecting them

Source: ISO 26262 2nd edition

Table 22 — Dependent failures initiators due to random physical root causes

DFI examples	Short circuits (e.g.: local defects, electro migration, via migration, contact migration, oxide break down) Latch up Cross talk (substrate current, capacitive coupling) Local heating caused e.g. by defective voltage regulators or output drivers
Measures to prevent dependent failures from violating the safety goal	Diversification of impact (e.g. clock delay between master & checker core, diverse master and checker core, different critical paths) Indirect detection (e.g. cyclic self-test of a function that would fail in the case of physical root cause) or indirect monitoring using special sensors (e.g. delay lines used as common-cause failure sensors)
Measures to prevent the occurrence of dependent failures during operation	Dedicated production tests Fault avoidance measures (e.g. physical separation/isolation, corresponding design rules) Physical separation on a single chip

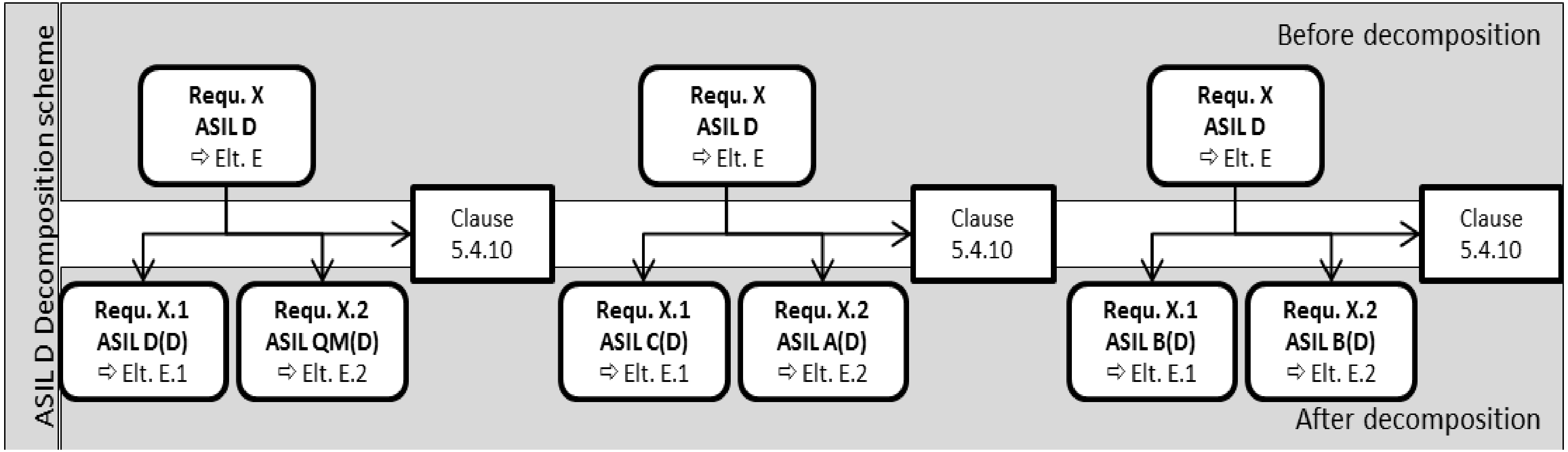
Table 23 — Systematic dependent failure initiators due to environmental conditions

DFI examples	Temperature Vibration Pressure Humidity/Condensation Corrosion EMI Overvoltage applied from external Mechanical stress Wear Aging Water and other fluids intrusion
Measures to prevent dependent failures from violating the safety goal	Diversification of impact (e.g. clock delay between master & checker core, diverse master and checker core, different critical paths) Direct monitoring of environmental conditions (e.g. temperature sensor) or indirect monitoring of environmental conditions (e.g. delay lines used as dependent -failure sensors)
Measures to prevent the occurrence of dependent failures during operation	Fault avoidance measures (e.g. conservative specification/robust design) Physical separation (e.g. distance of the die from a local heat source external to the die) Adaptive measures to reduce susceptibility (e.g. voltage/operating frequency decrease) Limit the access frequency or limit allowed operation cycles for subparts (e.g. specify the number of write cycles for an EEPROM) Robust design of semiconductor packaging

ASIL DECOMPOSITION

To Reduce Complexity

Source: ISO 26262 2nd edition



ASIL decomposition:

- apportioning of redundant safety requirements to elements, with sufficient independence, conducing to the same safety goal, with the objective of reducing the ASIL of the redundant safety requirements that are allocated to the corresponding elements.

S/W SAFETY

Mainly focusing on avoiding systematic failures

Source: ISO 26262 2nd edition

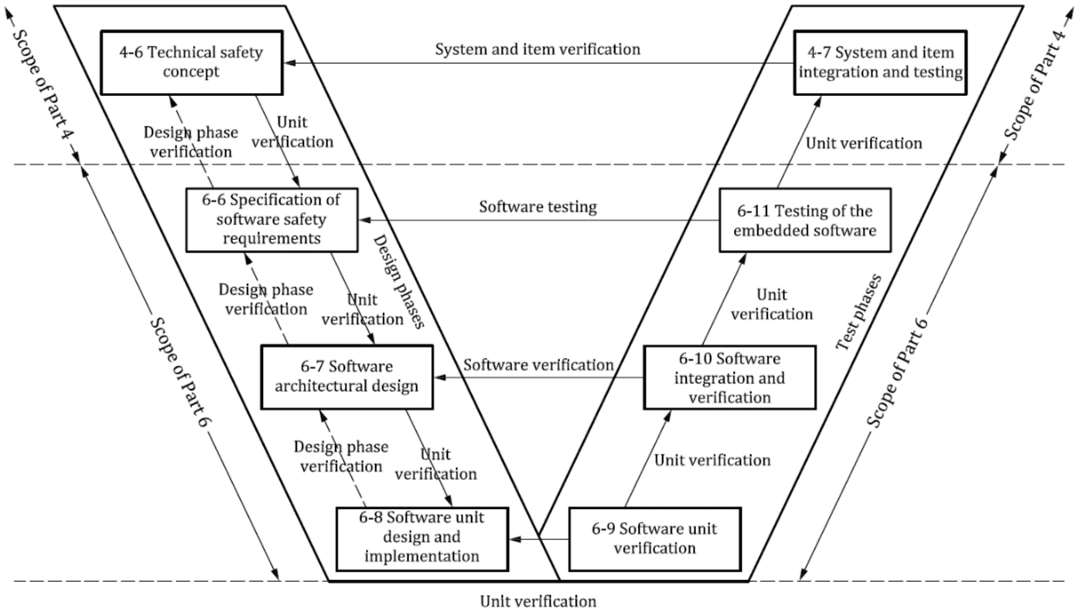


Table 6 — Design principles for software unit design and implementation

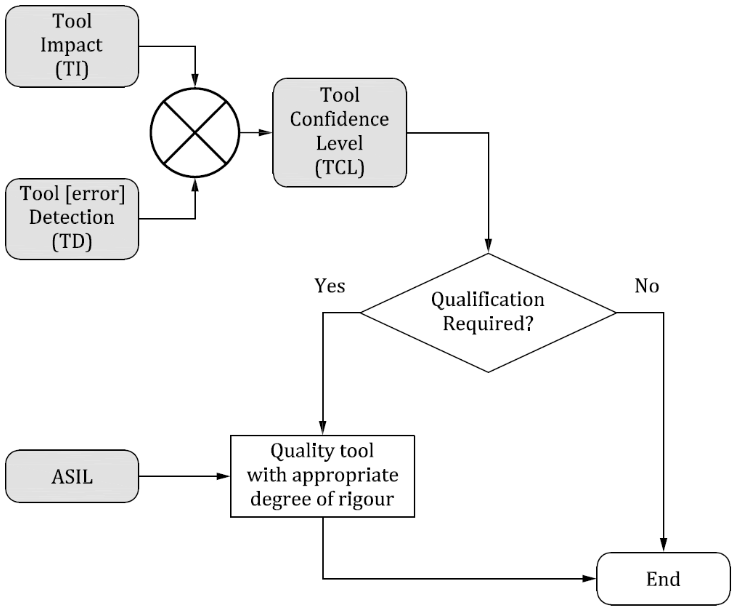
	Principle	ASIL A	ASIL B	ASIL C	ASIL D
1a	One entry and one exit point in subprograms and functions[a]	++	++	++	++
1b	No dynamic objects or variables, or else online test during their creation[a]	+	++	++	++
1c	Initialization of variables	++	++	++	++
1d	No multiple use of variable names[a]	++	++	++	++
1e	Avoid global variables or else justify their usage[a]	+	+	++	++
1f	Restricted use of pointers[a]	+	++	++	++
1g	No implicit type conversions[a]	+	++	++	++
1h	No hidden data flow or control flow	+	++	++	++
1i	No unconditional jumps[a]	++	++	++	++
1j	No recursions	+	+	++	++

[a] Principles 1a, 1b, 1d, 1e, 1f, 1g and 1i may not be applicable for graphical modelling notations used in model-based development.

NOTE For the C language, MISRA C (see Reference [3]) covers many of the principles listed in Table 6.

TOOL SAFETY

Determining confidence in use of tools



		Tool error detection		
		TD1	**TD2**	**TD3**
Tool impact	**TI1**	TCL1	TCL1	TCL1
	TI2	TCL1	TCL2	TCL3

Table 4 — Qualification of software tools classified TCL3

Methods		ASIL			
		A	B	C	D
1a	Increased confidence from use in accordance with 11.4.7	++	++	+	+
1b	Evaluation of the tool development process in accordance with 11.4.8	++	++	+	+
1c	Validation of the software tool in accordance with 11.4.9	+	+	++	++
1d	Development in accordance with a safety standard[a]	+	+	++	++

[a] No safety standard is fully applicable to the development of software tools. Instead, a relevant subset of requirements of the safety standard can be selected.

EXAMPLE Development of the software tool in accordance with ISO 26262, IEC 61508, EN 50128 or RTCA DO-178C.

Table 5 — Qualification of software tools classified TCL2

Methods		ASIL			
		A	B	C	D
1a	Increased confidence from use in accordance with 11.4.7	++	++	++	+
1b	Evaluation of the tool development process in accordance with 11.4.8	++	++	++	+
1c	Validation of the software tool in accordance with 11.4.9	+	+	+	++
1d	Development in accordance with a safety standard[a]	+	+	+	+

[a] No safety standard is fully applicable to the development of software tools. Instead, a relevant subset of requirements of the safety standard can be selected.

EXAMPLE Development of the software tool in accordance with ISO 26262, IEC 61508, EN 50128 or RTCA DO-178C.

SAFETY OF THE INTENDED FUNCTIONALITY

ISO 21448 (a.k.a. SOTIF)

- Autonomous vehicles that rely on sensing can miss their goal and cause safety violations even in absence of H/W or S/W failures, due to:
  - Sensor limitations
  - Algorithm limitations
  - Actuator limitations

SAFETY OF THE INTENDED FUNCTIONALITY

ISO 21448 (a.k.a. SOTIF)

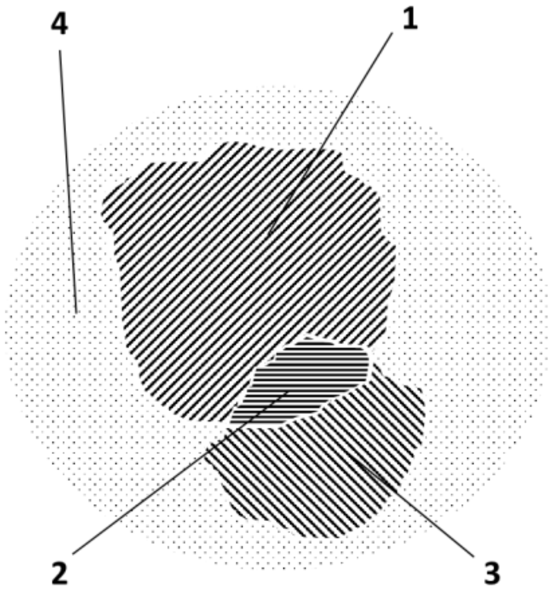
Source	Cause of hazardous event	Within scope of
System	E/E System failures	ISO 26262 series
	Performance limitations or insufficient situational awareness, with or without reasonably foreseeable misuse	ISO/PAS 21448
	Reasonably foreseeable misuse, incorrect HMI (e.g. user confusion, user overload)	ISO/PAS 21448 ISO 26262 series European statement of principal on the design of human-machine-interface
	Hazards caused by the system technology	Specific standards
External factor	successful attack exploiting vehicle security vulnerabilities	ISO 21434[a] or SAE J3061
	Impact from active Infrastructure and/or vehicle to vehicle communication, external devices and cloud services.	ISO 20077 series; ISO 26262 series
	Impact from car surroundings (other users, "passive" infrastructure, environmental conditions: weather, Electro-Magnetic Interference...)	ISO/PAS 21448 ISO 26262 series

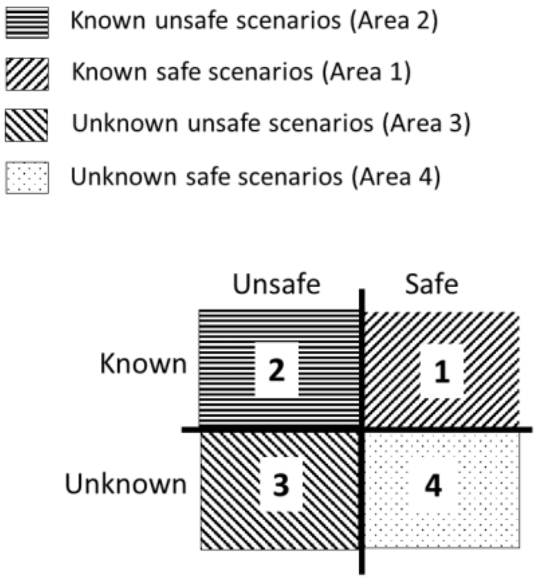
[a] Under preparation. Stage at the time of publication: ISO/SAE CD 21434.

SOTIF GOAL

Known, Unknown, Safe and Unsafe

Source: ISO/PAS 21448

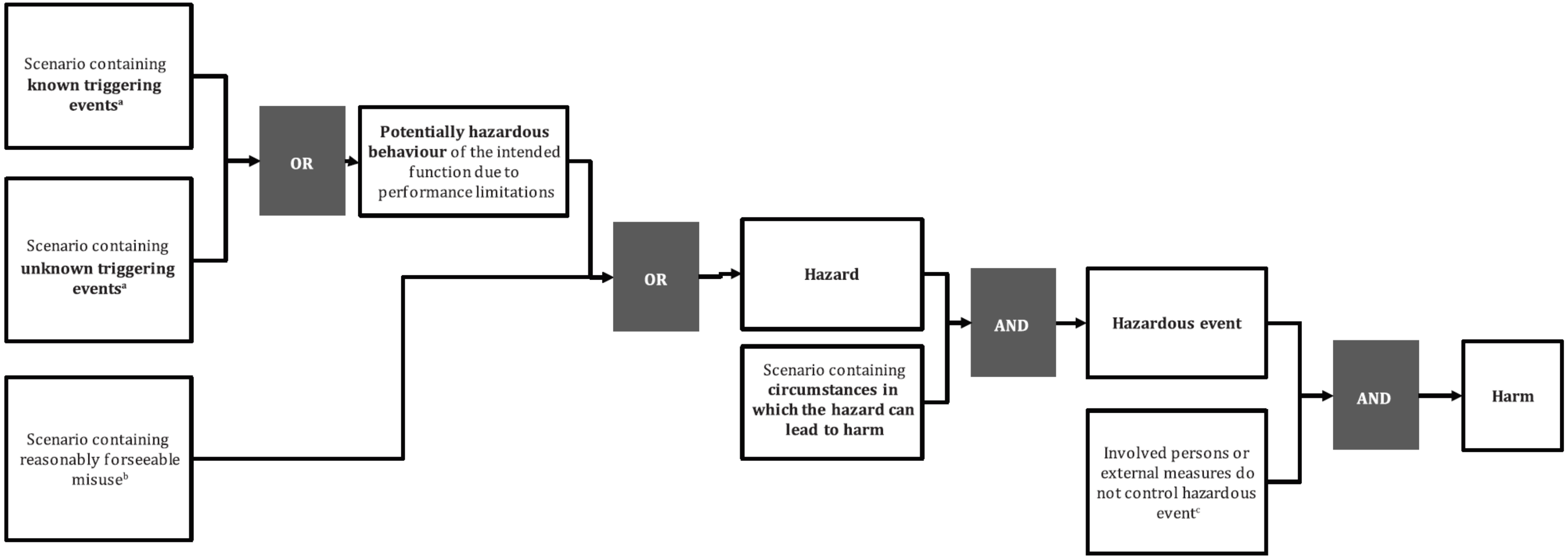




- At the beginning of the development Areas 2 and Area 3 might be too large, resulting in unacceptable residual risk.
- The ultimate goal of the SOTIF activities to evaluate the SOTIF in Area 2 and Area 3 and to provide an argument that these areas are sufficiently small and therefore that the resulting residual risk is acceptable.

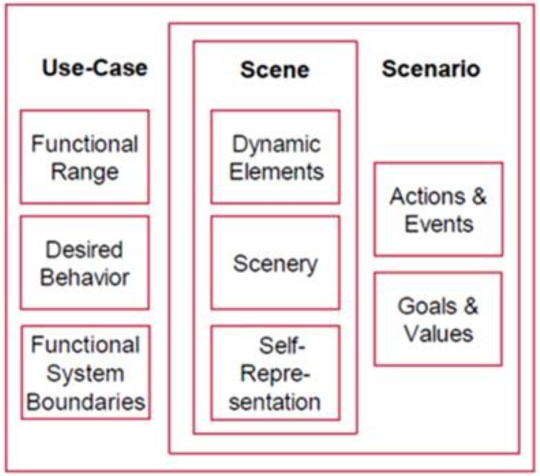
ASSESSING THE SOTIF RISK OF HARM

From scenarios to harm

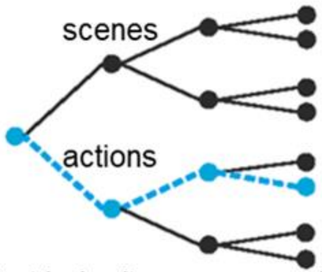


BREAKING DOWN THE COMPLEXITY

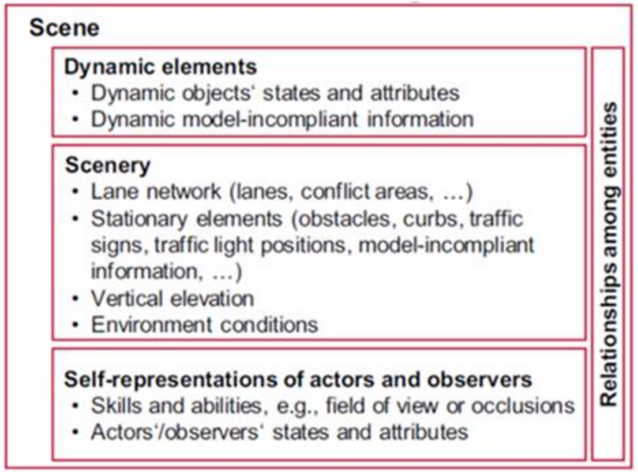
Scene, Scenario, Situation



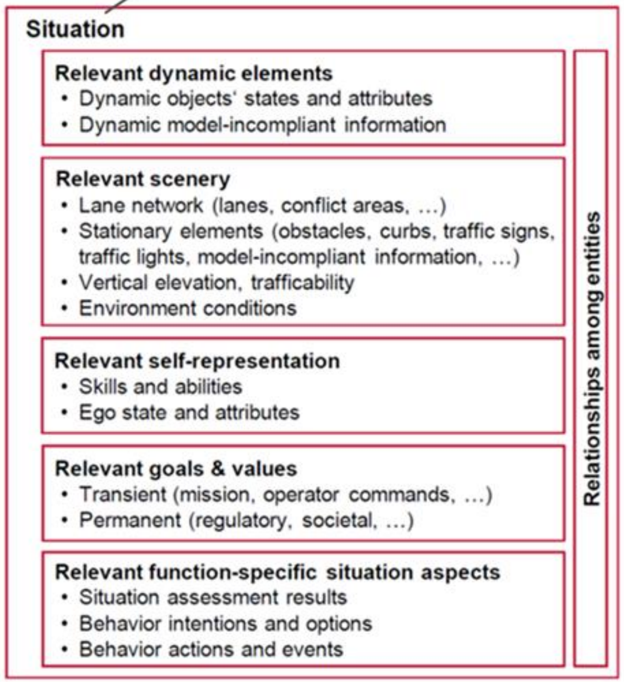
Only a scene in a simulated world can be all-encompassing (i.e. an objective scene). In the real world the scene is incomplete, incorrect, uncertain, and from one or several observers' points of view (i.e. a subjective scene)



Scenario (dashed) as a temporal sequence of actions/events (edges) and scenes (nodes)



selection of an appropriate behaviour pattern at a particular point of time



Source: ISO/PAS 21448

PUSHING VALIDATION TO ITS LIMIT

End-to-end Testing and Validation

Table 4: Deriving test cases for verification and validation

	Methods
1a	Analysis of requirements
1b	Analysis of external and internal interfaces
1c	Generation and analysis of equivalence classes
1d	Analysis of boundary values
1e	Error guessing based on knowledge or experience
1f	Analysis of functional dependencies
1g	Analysis of common limit conditions, sequences, and sources of dependent failures
1h	Analysis of environmental conditions and operational use cases[a]
1i	Analysis of field experience[b]
1j	Analysis of system architecture (including redundancies)
1k	Analysis of sensors design and their known potential limitations
1l	Analysis of algorithms and their decision paths[c]
1 m	Analysis of system ageing

[a] including known sources of potential unintended behaviour of the element or system

[b] this considers various driving conditions, driving styles, driving environment and end customer claims

[c] including also the analysis of machine learning algorithms, if they are used (see Annex G)

Table 5: Sensor verification

Table 6: Decision Algorithm verification

Table 7: Actuation verification

Table 8: Integrated system verification

1a	Verification of robustness to … injection testing)
1b	Requirement-based Test wher… range, precision, resolution, ti…
1b	In the loop testing (e.g. SIL / H…
1c	System test under different en… damp conditions)
1d	Verification of system ageing a…
1e	Randomized input tests [a]
1f	Vehicle level testing on selecte…
1g	Controllability tests

[a] Randomised input tests can inclu… sensors adding flipped images, alte… adding ghost targets by multiple-p…

Table 9: Evaluation of residual risk

	Methods
1a	Validation of robustness to Signal-to-Noise Ratio degradation (e.g. by noise injection testing)
1b	Verification of the architectural properties including independence, if applicable
1c	In the loop testing on randomized test cases (derived from a technical analysis and by error guessing)
1d	Randomized input tests [a]
1e	Vehicle level testing on selected test cases (derived from a technical analysis and by error guessing)
1f	Long term vehicle test
1g	Captured fleet tests
1h	Test derived from field experience
1i	Tests of corner cases

[a] Randomised input tests can include erroneous patterns e.g. in the case of image sensors adding flipped images, altered image patches or in case of radar sensors adding ghost targets by multiple-path.

Source: ISO/PAS 21448

DATA COLLECTION

SOTIF Guidelines

Source: ISO/PAS 21448

Time of day		
Type		**Percentage**
Day		50 %
Night		35 %
Dusk		15 %
Vehicle Speed		
Speed [mi/h]	**Speed [km/h]**	**Percentage**
0–25	0–40	60 %
26–50	41–80	40 %
>50	>80	0 %
Weather condition		
Type		**Percentage**
Dry/Clear sky		65 %
Rain		7 %
Fog		5 %
Snow		5 %
Overcast		10 %
Heavy rain		5 %

- Continuous data collection, in different markets, weather and illumination conditions.
- Specific data collection, in conditions which are normally rare and less represented in normal driving but that might impact perception:
  - Vision perception — data at dusk or dawn;
  - Lidar system — adverse weather;
  - Radar system — rain and splash conditions on salt spread roads;
  - All systems — entering, exiting or within a tunnel.
- Specific data collection, in uncommon scenarios that might increase the likelihood of a safety violation, e.g. driving on roads with sparse traffic and no lead cars can increase the probability of failure of in-path target selection and detection of ghost targets.
- Specific data collection, based on system limitations.

SOTIF MEASURES

Example

Source: ISO/PAS 21448

	Causal factor of hazard with example	Example of derived SOTIF measure
E/E System Factor	E/E System performance limitation	• Reduce the performance of the system and inform the driver and handover the authority to the driver. • Gently stop the function • Degrade and keep the function
Driver Factor	Reasonably foreseeable misuse	• Prevent inadvertent operation by the driver. • Monitor and warn the driver when an incorrect operation is detected.

DNN SAFETY

FUSA

Correctness of DNN model implementation in SW

Correct software implementation of the deep learning framework

Ability to avoid or detect faults introduced by tools

Systematic issues in the training process

Vulnerability analysis of GPU

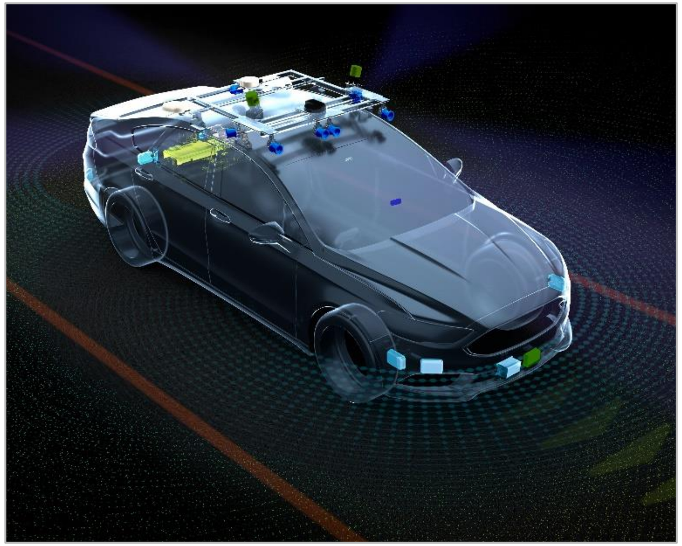
SOTIF

Quality and completeness of the training

Quality and completeness of the verification and validation

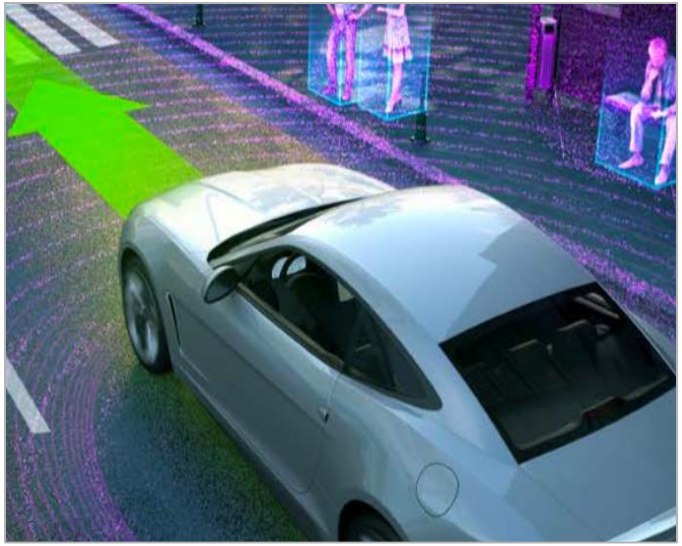
AV SAFETY VALIDATION

The Challenges



Highly Complex System
Large Computers, DNNs, Sensors

Real-Life Scenario Coverage
Account for Rare & Unpredictable Cases



Continuous Reaction Loop
Vehicle & World are Dependent

THE AV VALIDATION GAP

COMPONENT LEVEL SIL

Low Fidelity | Scalable

ON ROAD TESTING

High Fidelity| Doesn’t Scale

No Coverage for
Extreme & Dangerous Scenarios

AV REQUIRES A COMPREHENSIVE VALIDATION APPROACH

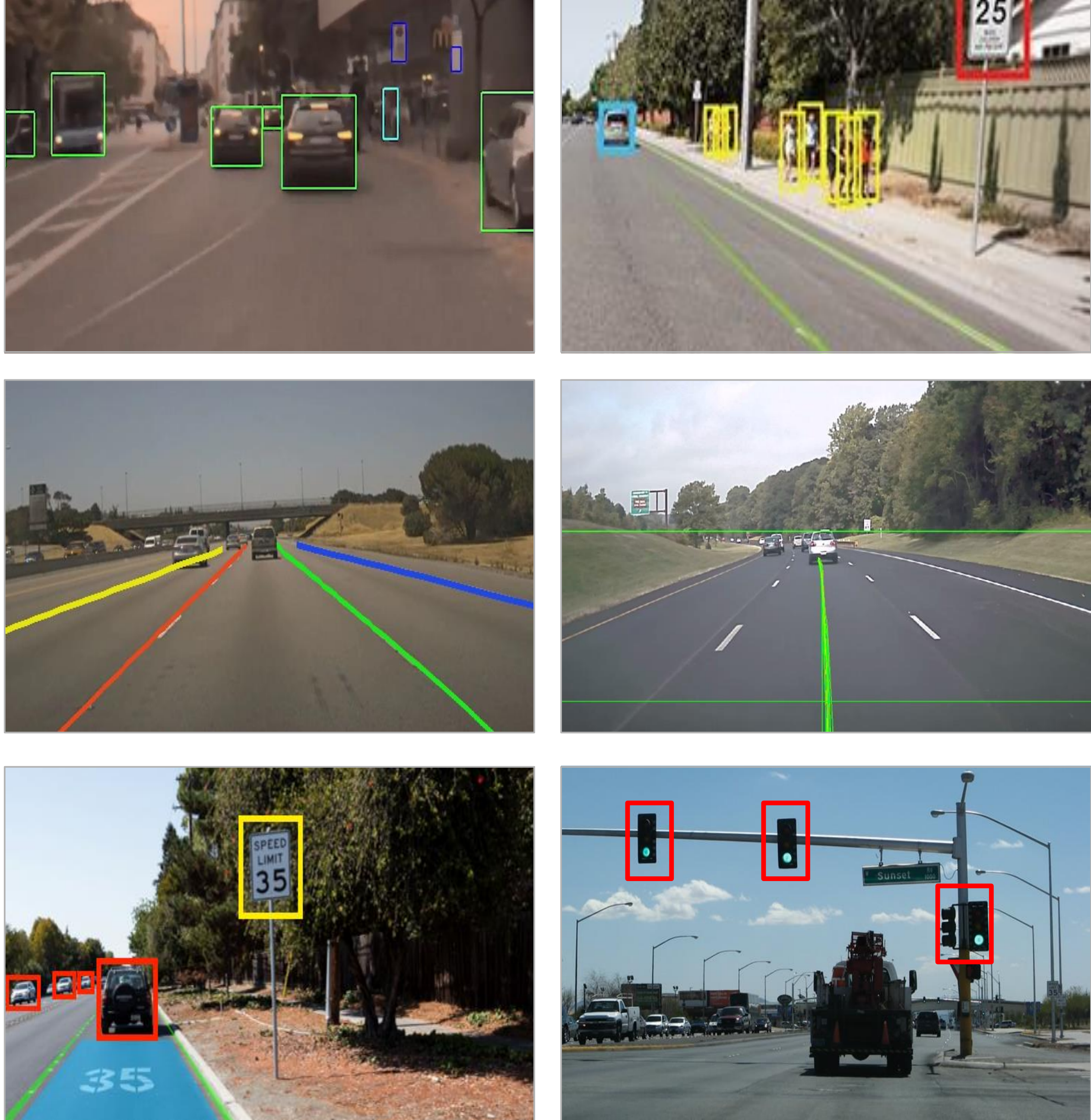
End-to-End System Level Test

Large Scale | Millions of Miles

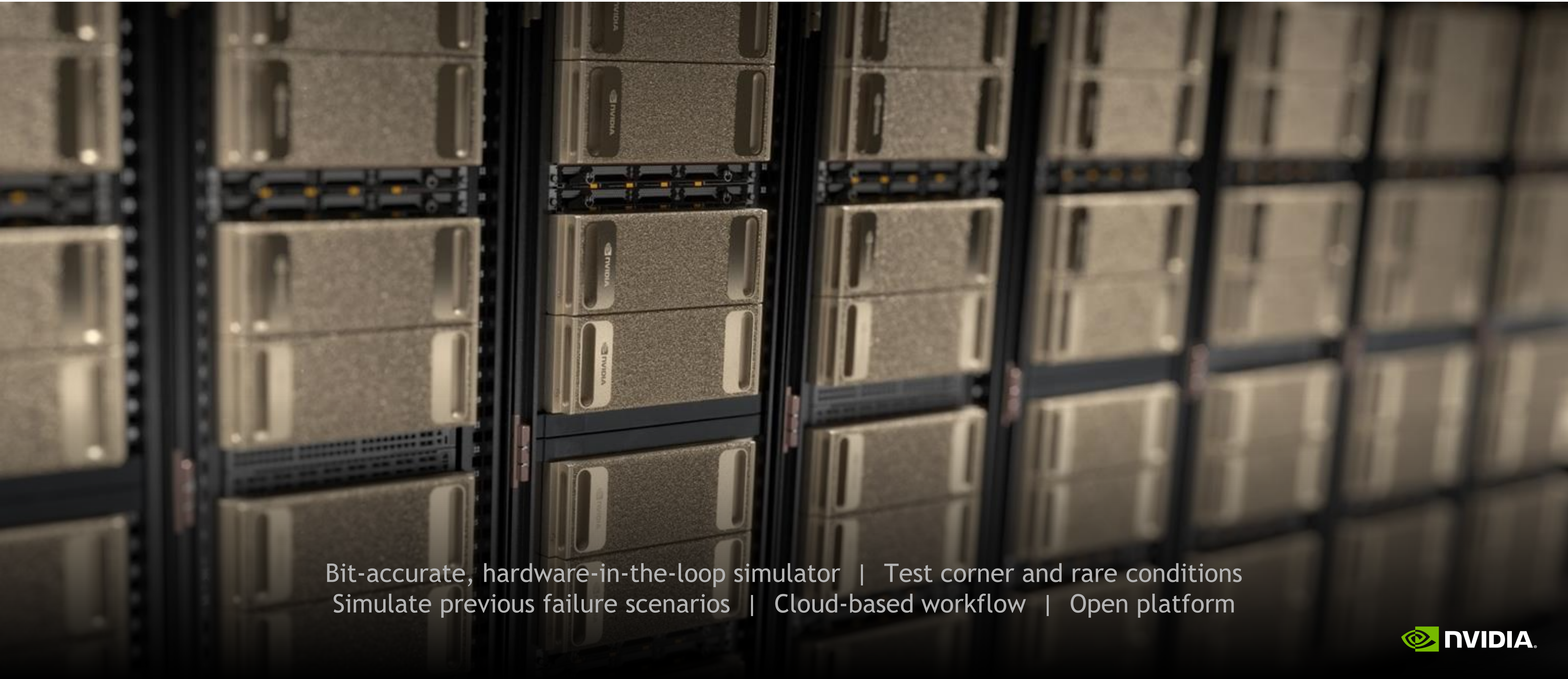
Diverse Vehicle and World Conditions

Data Driven | Scenario based

Repeatable and Reproducible



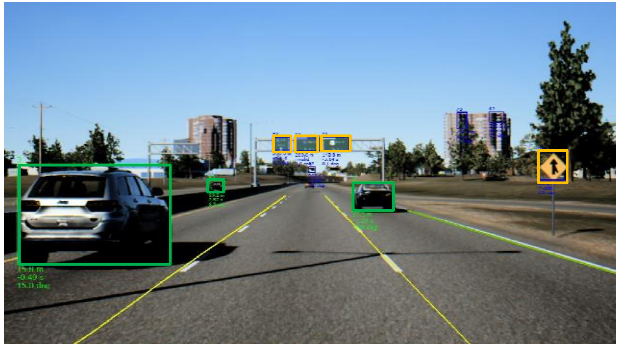
VIRTUAL TEST FLEET IN THE CLOUD



Bit-accurate, hardware-in-the-loop simulator | Test corner and rare conditions
Simulate previous failure scenarios | Cloud-based workflow | Open platform

HARDWARE IN THE LOOP SIMULATION

Bit Accurate & Timing Accurate



PERCEPTION

Camera | Radar | Lidar | IMU

CONTROL

Steering | Throttle | Brake

BEYOND VALIDATION

The Need for Formal Models and Methods

- Industry recognized that validation, despite essential to provide safety of automated vehicles, *per se* is not enough.

- It is necessary to combine validation with an overarching theory (and related mechanisms) for mapping world perception into constraints on control that, if obeyed, prevents “all” collisions.

- Those mechanisms should, as much as possible, function independently of the full complexity of software required to obey all traffic rules and rules courteously.

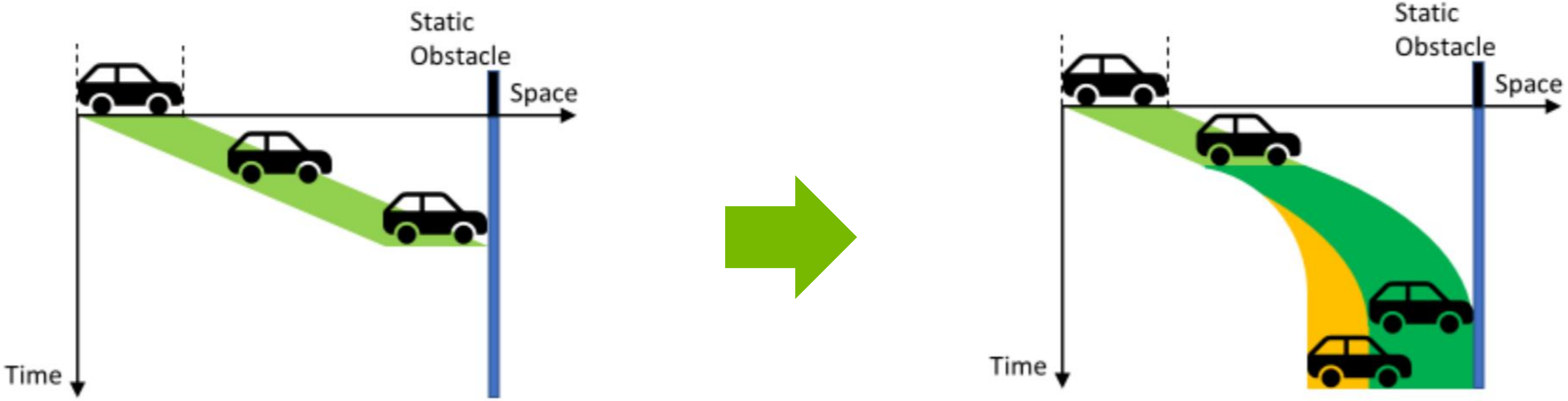
- NVIDIA outlined a safety driving policy known as “Safety Force Field”, or SFF.

- SFF consists of “forces” acting on every actor (including my car) so that collisions between any two actors are avoided.

SFF IN A NUTSHELL

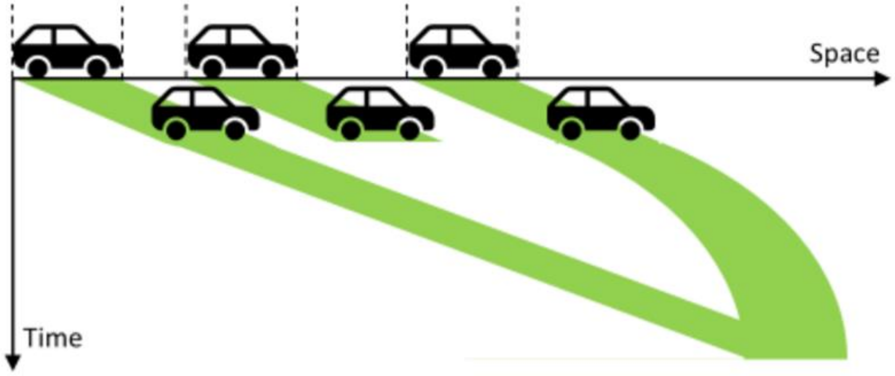
Details: www.nvidia.com/en-us/self-driving-cars/safety-force-field/

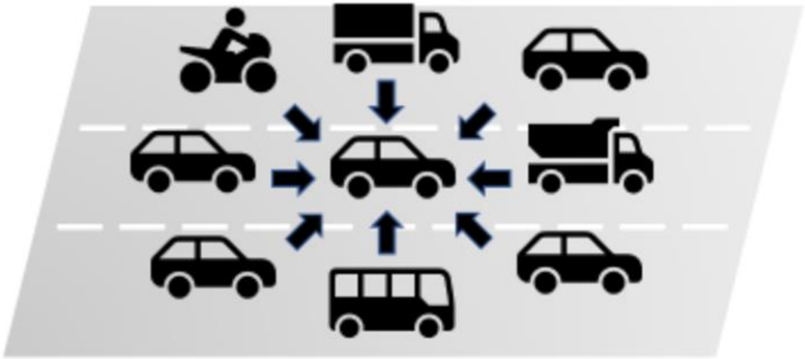
- SFF is built on a simple single core safety principle rather than a complex set of case-by-case rules, which can get unwieldy to implement and validate.
  - Example: the safety procedure is a requirement to decelerate at least as much as a certain amount (dark green). There is also a maximum braking schedule (orange).



SAFETY DEPENDS ON OTHER ACTORS

It is Not Possible to Guarantee Absence of Collisions Regardless of What Other Actors Do



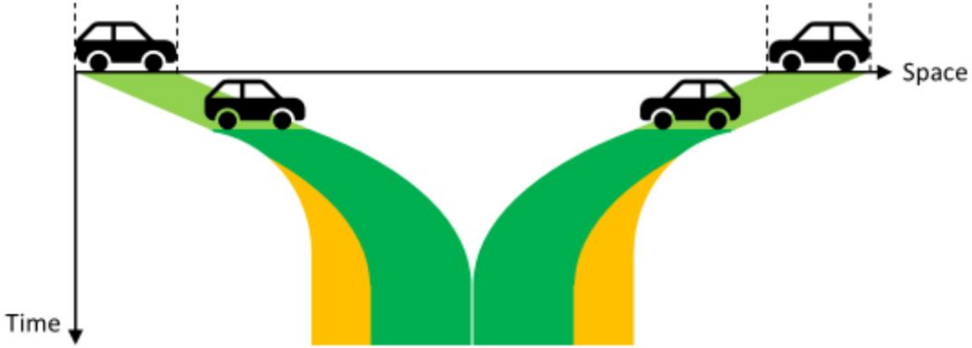


The vehicle in the middle has nowhere to go if its lead vehicle decides to brake and the following vehicle continues to accelerate.

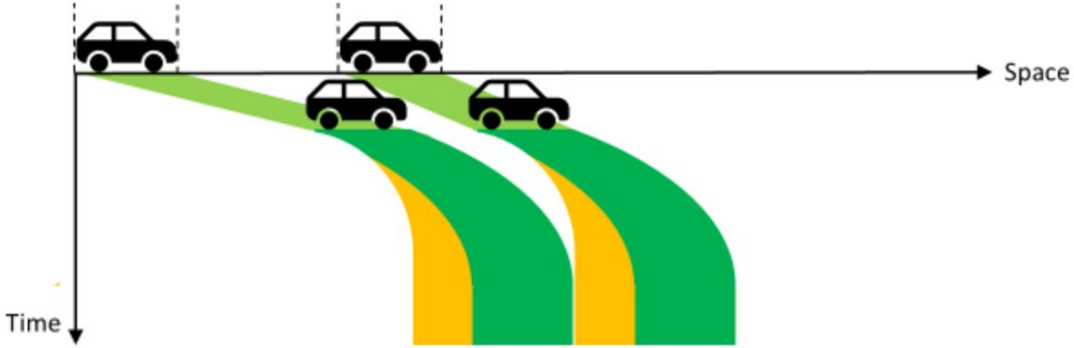
The situation is the same in two dimensions since other vehicles may be blocking the sides. We could ask that we be stopped before a collision
occurs but would then be unable to drive at speed on a congested highway….

COLLABORATING FOR SAFETY

Both Actors have to Apply their Safety Procedures



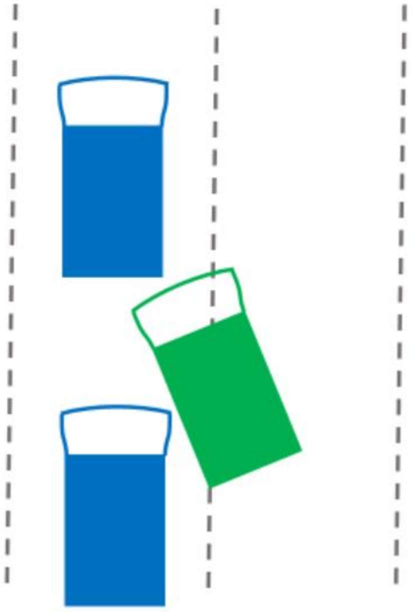
In the case of two oncoming cars, the minimal constraint is that both actors have to apply their safety procedures just before they are about to overlap.



The case of one car following another also becomes critical exactly when the claimed sets intersect. At that moment, the following car has to apply its safety procedure, while the front car has no constraint other than staying ahead of maximum deceleration.

LATERAL AND LONGITUDINAL

The longitudinal and lateral dimensions shall be handled jointly



An approach that looks at longitudinal and lateral safety margins separately cannot allow the case of pushing diagonally into a lane at low speed. The reason is that at high congestion, we cannot expect to longitudinally clear the vehicle we want to take way from before we are partially in its lateral path.

SFF naturally allows making way into a congested lane at slow speed as can be required in congested highway situations. This is not possible with a formulation that separates lateral and longitudinal distances and requires at least one of them to be acceptable. Note that in this situation, the ego vehicle (green) is neither laterally nor longitudinally clear from the car behind it to the left.

THE MATHEMATICAL MODEL BEHIND

Details: www.nvidia.com/en-us/self-driving-cars/safety-force-field/

***Definition 1**: The state of actor A is a vector $x_A(t) \in \mathbb{R}^m$ as a function of time that encodes the properties of actor A at time t. When viewed as a function of time, we refer to it as the state trajectory of actor A.*

***Definition 2**: The set Ω is the collection of the state spaces of all actors we consider, including static obstacles.*

***Definition 3**: A control model $f(x_A, t, c)$ for actor A is a function f of the state x_A of the actor, time t, and control parameters c into $\mathbb{R}^m$.*

***Definition 12**: A safe control policy $\frac{dx_A}{dt}$ for actor A with respect to a set $\Theta \subseteq \Omega$ of actors is one for which $F_{AB} \frac{dx_A}{dt} \geq \min_{s_A \in S_A} F_{AB} s_A$ for each other actor $B \in \Theta$.*

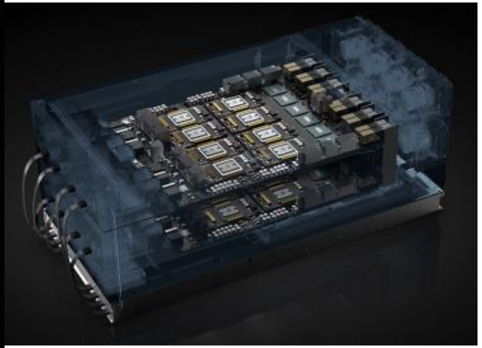
POWERING THE AI REVOLUTION

Fusing HPC and AI computing into one unified architecture

Datacenter

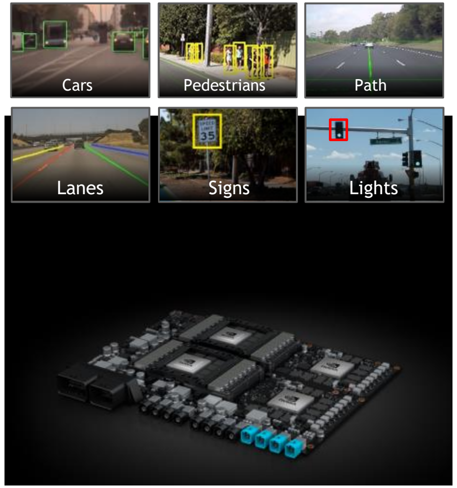
NEW DGX2
2 PFLOPS | 512GB HBM2 | 10 kW

Cloud



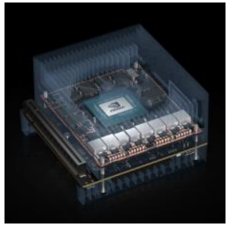
NEW HGX2
2 PFLOPS

Vehicle



NEW DRIVE™ Pegasus
320 TOPS | 2x Xavier + 2x Next Gen GPU

Machines

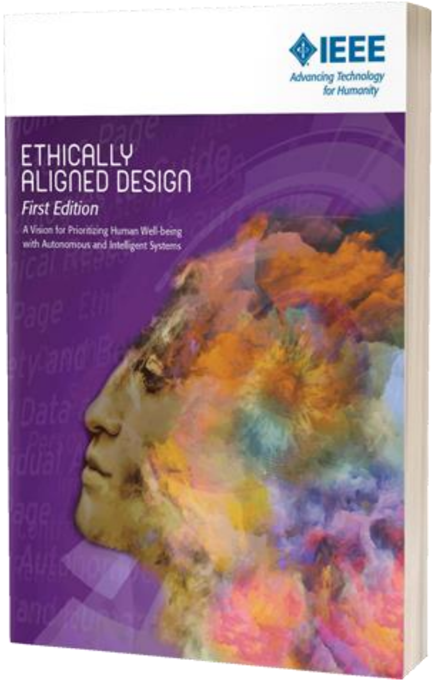


ISAAC Robotic platform
JETSON Xavier DevKit

IEEE INITIATIVES

IEEE COMPUTER SOCIETY
RELIABLE, SAFE, SECURE, AND TIME-DETERMINISTIC INTELLIGENT SYSTEMS
Special Technical Community

www.computer.org/communities/special-technical-communities/rsstdis



NVIDIA®