

CSC2125: Safety and Certification of Autonomous Vehicles

Lecture 3: Automotive Safety
Instructor: Prof. Marsha Chechik

Lecture Credits

- Research and talks by members of Safety Project (Toronto + McMaster)
 - Mark Lawford
 - Alan Wass Yong
 - Sahar Kokaly
- Alan Wass Yong's talk on hazards
- Toyota unintended acceleration analysis by Phillip Koopman (CMU)
- ISO 26262 process:
 - Functional Safety Draft International Standard for Road Vehicles: Background, Status, and Overview Barbara J. Czerny, Joseph D'Ambrosio, Rami Debouk (GM R&D), Kelly Stashko (GM Powertrain)
 - ISO 26262 introduction, Koen Leekens
 - Functional Safety with Automated Functional Testing – QA Systems GmbH, 2017

Outline

- Why do safety analyses?
- Introduction to ISO 26262
- Spotlight: Hazard analysis
- Spotlight: Safety assurance cases
- Spotlight: Evolution of safety arguments
- Spotlight: Verification and validation
- Toyota unintended acceleration case study

Why Safety?



Modern Car



Adaptive Headlights

Pre-Crash System

Automatic Steering

Backup Camera

Infrared Night Vision

Steering Lock

Traction Control System

Anti-Blocking System

Corner Brake Control

Adaptive Cruise Control

Automatic Collision Notification

Automated Parking System

Automatic Gearbox Control

Airbag

Electronic Stability Program

Tire Pressure Monitoring

Reverse Sensors

Lane Departure Warning

Deflation Detection System

Emergency Brake Assistance

Traffic Sign Recognition

ISO 26262 – Functional Safety of Road Vehicles

Goals of this part

- Exposure to the overall safety process
- Notion of hazard analysis
- Notion of risk assessment and ASIL determination
- Notion of safety case
- V&V methods

Functional Safety

- ISO 26262 (2011): Absence of unreasonable risk due to hazards caused by malfunctioning behavior of E/E (electrical/electronic) system
 - State of the art for automotive
 - Developed with OEM (General Motors in particular)
- IEC 61508: Part of the overall safety related to the equipment under control (EUC) that depends on the correct functioning of the safety-related system

How do E/E systems fail?

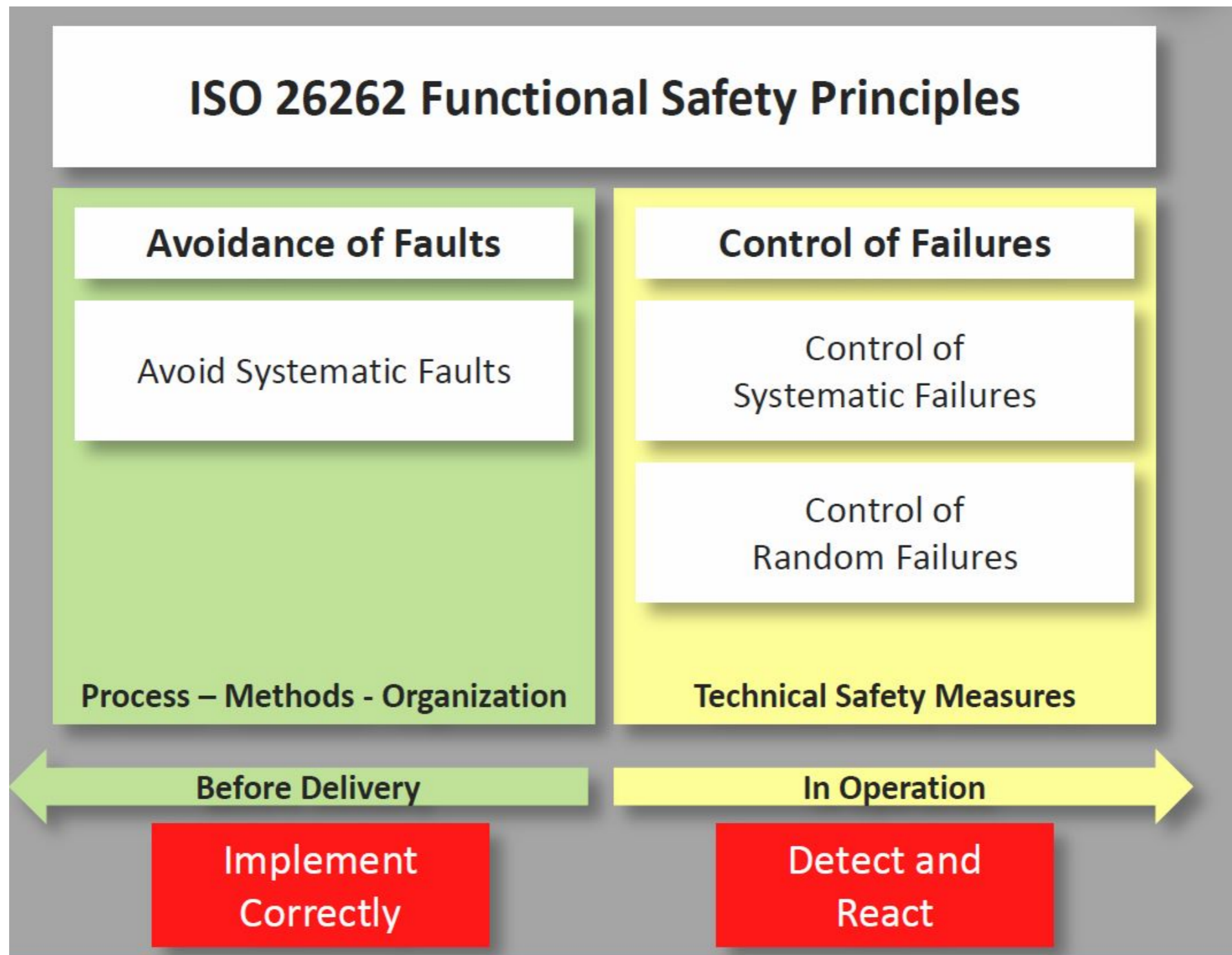
- ▶ **Random Failures:** *“Usually a permanent or transient failure due to a system component loss of functionality – hardware related*



- ▶ **Systematic Failures:** *“Usually due to a design fault, wrong specification, not fit for purpose, error in software program, ...*



ISO 26262 Principles



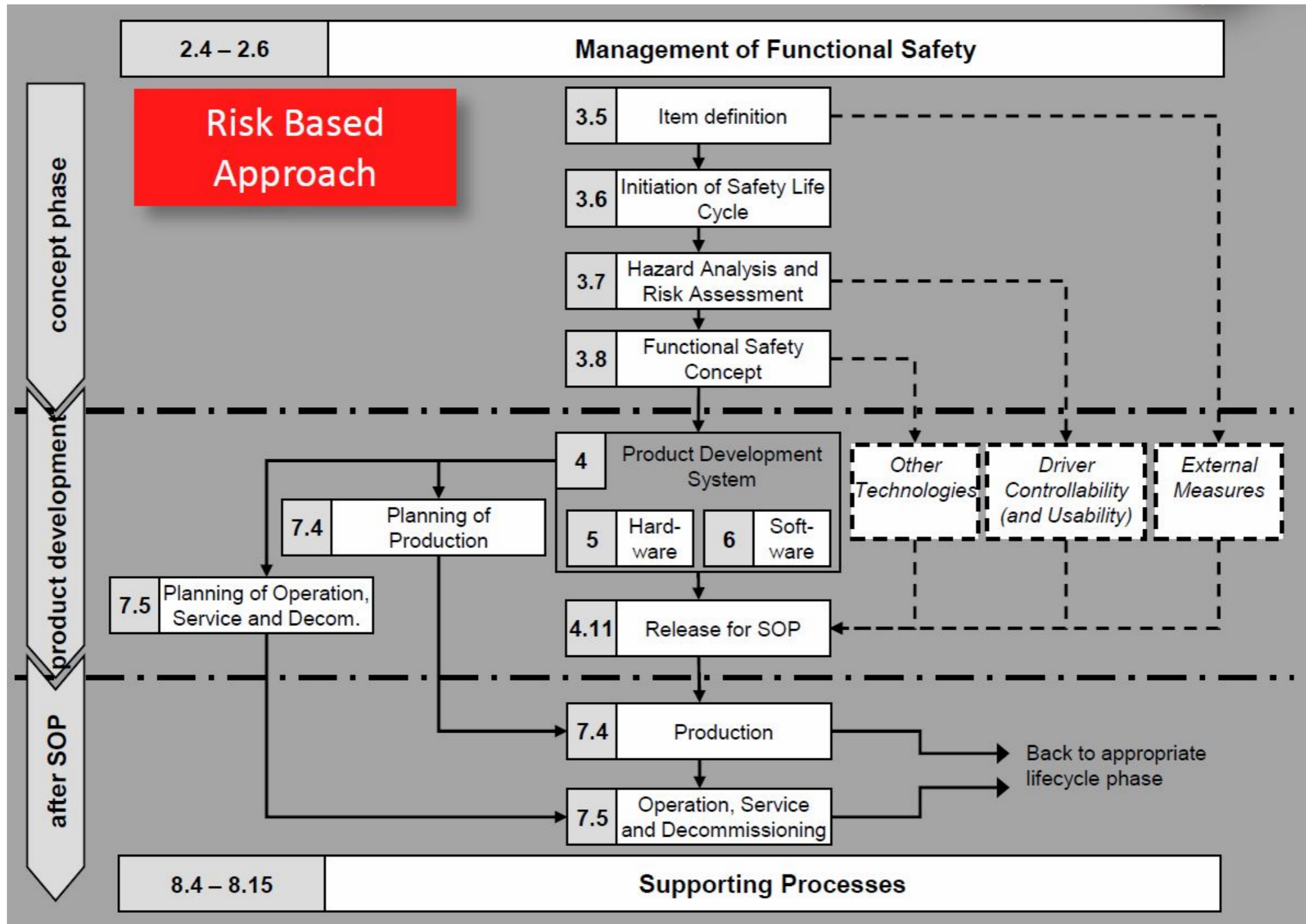
ISO26262 - Functional Safety of Road Vehicles

Standard has 10 parts

- Span across **~450 pages**
- Require the production of **~120 work products**
... that are the result of fulfilling a much larger number of **requirements** and **recommendations**



ISO 26262 follows a Safety Lifecycle



If you did ISO 26262 right

You are Able to Show:



– Completeness:

- Everything accounted for
- Requirements under Control
- Everything tested – pass
- Used the toolsets

– Traceability:

- Structured Process Model
- Documents linked
- Evidence for Everything
- Understandable for external

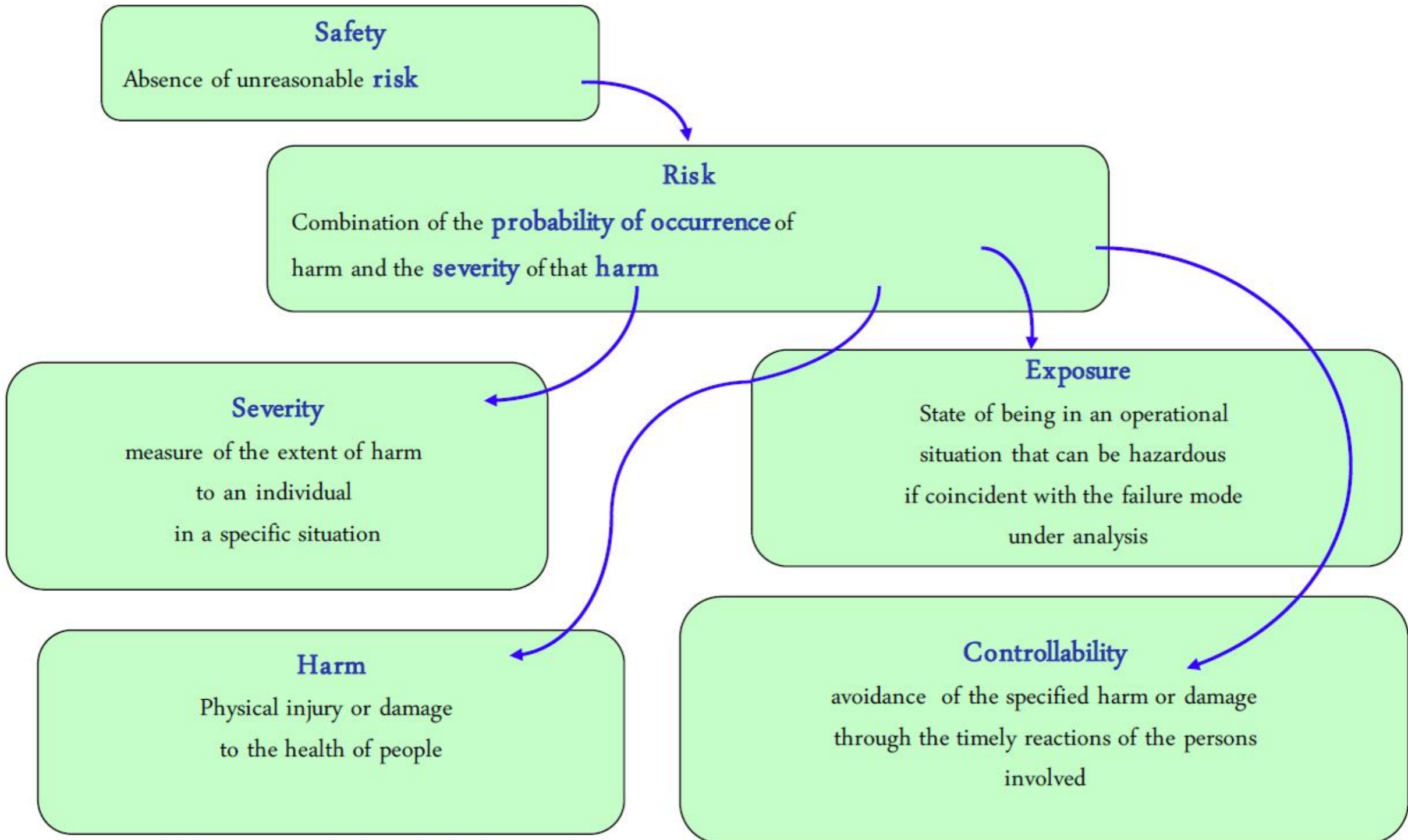
– Consistency

- This is visible for external auditor even when project members have left

– Documentation:

- All activities planned
- Execution documented in SC
- Inspected - Archived
- For a life-time (15year?)

Terminology



Safety Mechanism

Safety Mechanism

- Activity or technical solution to detect / avoid / control failures or mitigate their harmful effects
- Implemented by an E/E function or element or in other technologies
- The safety mechanism is either
 - able to switch to or maintain the item in a safe state or
 - able to alert the driver such that the driver is expected to control the effect of the failure

Work Products

Work product

- Information or data
- The result of one or more system safety process activities
- Format appropriate to the work product's content
 - Data files, models, source code, etc.
 - May include currently existing documents
 - Several work products may be in one document

Confirmation Measures

Confirmation measures

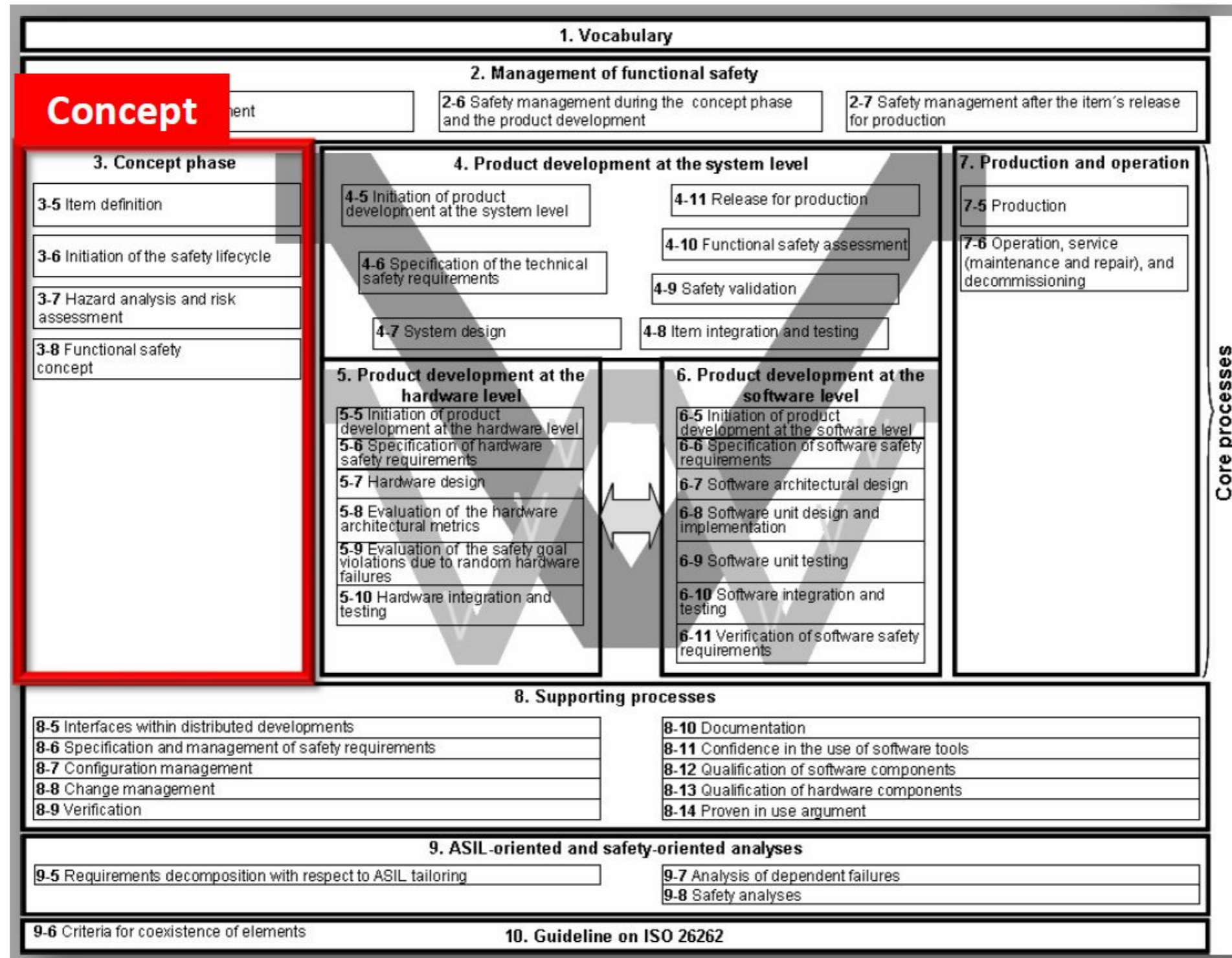
- Ensure the sufficient completion of work products and proper execution of the safety lifecycle.
- Provide for the evaluation of the system safety activities and work products as a whole
- Used to determine the adequacy of achievement of the functional safety goals

Safety Case

Safety case

- Communicates a clear, comprehensive and defensible argument (supported by evidence) that a system is acceptably safe to operate in a particular context.
- Includes references to safety requirements and supporting evidence
- AND a “safety argument” that describes how the safety requirements have been interpreted, allocated, decomposed, etc., and fulfilled as shown by the supporting evidence.

Concept Phase



Concept Phase

- OEM defines item – e.g., prevent use by unauthorized person by mechanical lock
- Initiation of safety lifecycle
- Hazard analysis
 - What can go wrong?
- Risk assessment
 - How risky is that?
 - Use ASILs to answer that

SG No.	HRA Reg	Safety Goal	ASIL	Safe State
SG1	ESCL_001	Unintended locking of ESCL while vehicle is moving shall be avoided	?	Unlocked ESCL

SAFETY GOAL
Avoid a Dangerous Situation

- Functional safety concept

Perform a Hazard Analysis. Determine ASIL

□ Situation Analysis & Hazard Identification

“Identify potential unintended behaviors of the item that could lead to a hazardous event.”

- Vehicle Usage
- Environmental Conditions
- Foreseeable driver use and misuse
- Interaction between vehicle systems

Consequence - Likelihood

$$\text{Risk} = S \times f$$

$$= S \times (E \times C)$$

f - frequency
 S - Severity of violation
 E - Exposure
 C - Controllability

**Moderation Always
with OEM**

	SEVERITY
S0	No injuries
S1	Light / Moderate Injuries
S2	Severe / "Survival probable" injuries
S3	"Survival uncertain" / Fatal injuries

	EXPOSURE
E1	Extremely Low Probability
E2	Low Probability
E3	Medium Probability
E4	High Probability

	CONTROLLABILITY
C1	Simply Controllable
C2	Normally Controllable
C3	Difficult to control / Uncontrollable



		S1	S2	S3
C1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	ASIL A
	E4	QM	ASIL A	ASIL B
C2	E1	QM	QM	QM
	E2	QM	QM	ASIL A
	E3	QM	ASIL A	ASIL B
	E4	ASIL A	ASIL B	ASIL C
C3	E1	QM	QM	ASIL A
	E2	QM	ASIL A	ASIL B
	E3	ASIL A	ASIL B	ASIL C
	E4	ASIL B	ASIL C	ASIL D

Example ASILs

Risk classification	Description of risk	Examples of hazardous event
QM	The combination of probability of accident (<i>Controllability</i> and <i>Exposure</i>) and severity of harm to persons (<i>Severity</i>) given the hazard is considered acceptable. With a QM classification, there are no ISO 26262 requirements on the development	No locking of steering column when leaving the vehicle in a parked position. Not possible to open sunroof
ASIL A	A low combination of probability of accident and severity of harm to persons given the hazard occurring	No airbag deployment in a crash fulfilling airbag deployment criteria
ASIL B	...	Unintended hard acceleration of vehicle during driving
ASIL C	...	Unintended hard braking of vehicle during driving while maintaining vehicle stability
ASIL D	Highest probability of accident and severity of harm to persons given the hazard occurring	Unintended locking of steering column during driving

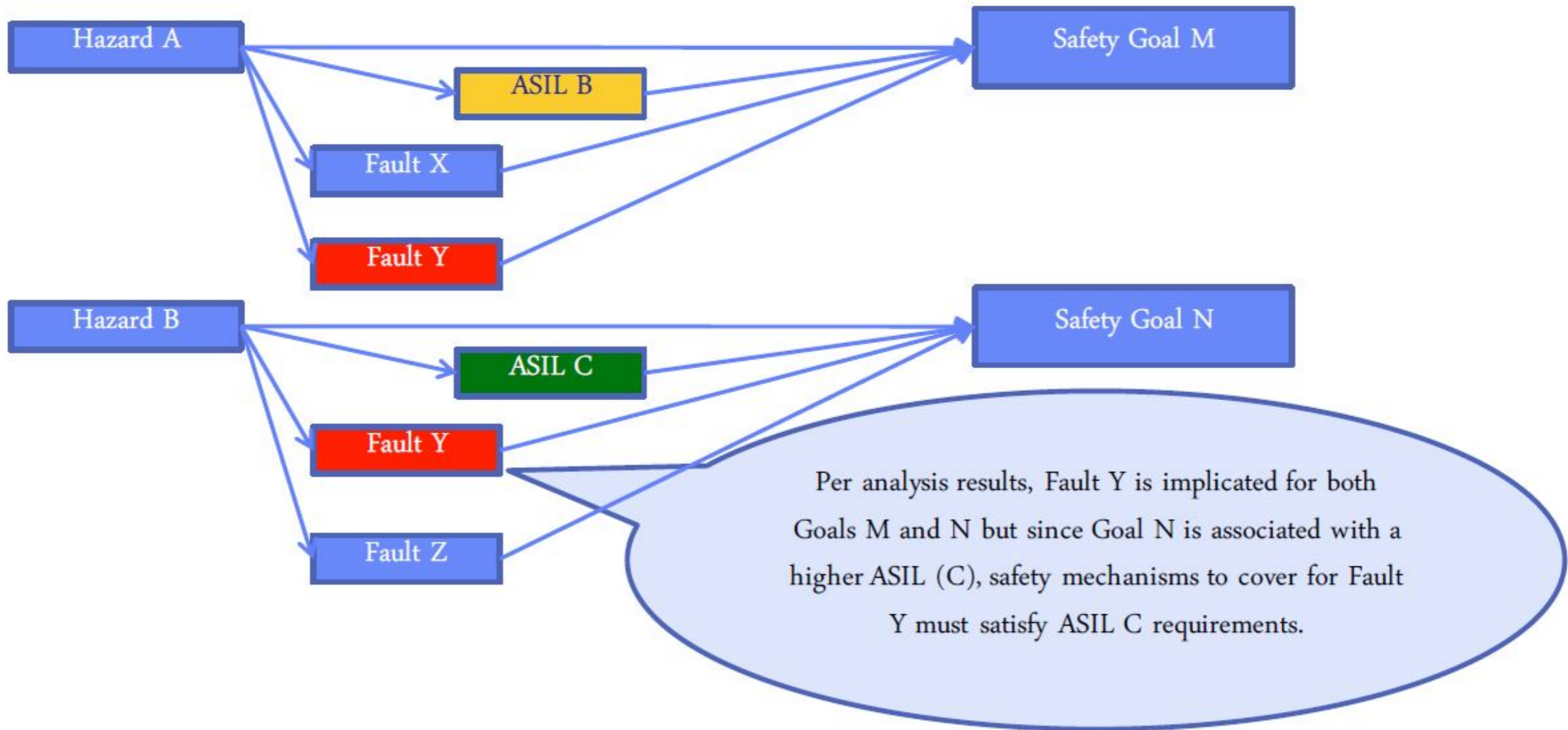
Example Outcome of Hazard Analysis

Function	Hazard	Situation	Hazardous event	ASIL	Safety goal
Steering column lock	Unintended steering column lock	Driving in curve with oncoming traffic	Driver loses control of his vehicle, entering the lane with oncoming traffic	D	Steering column lock shall not be locked during driving
Driver airbags	No deployment of driver airbags	Crash where airbag should deploy	Driver is not protected by airbags in a crash when he should be	A	Driver airbag shall deploy in crash, meeting deployment criteria

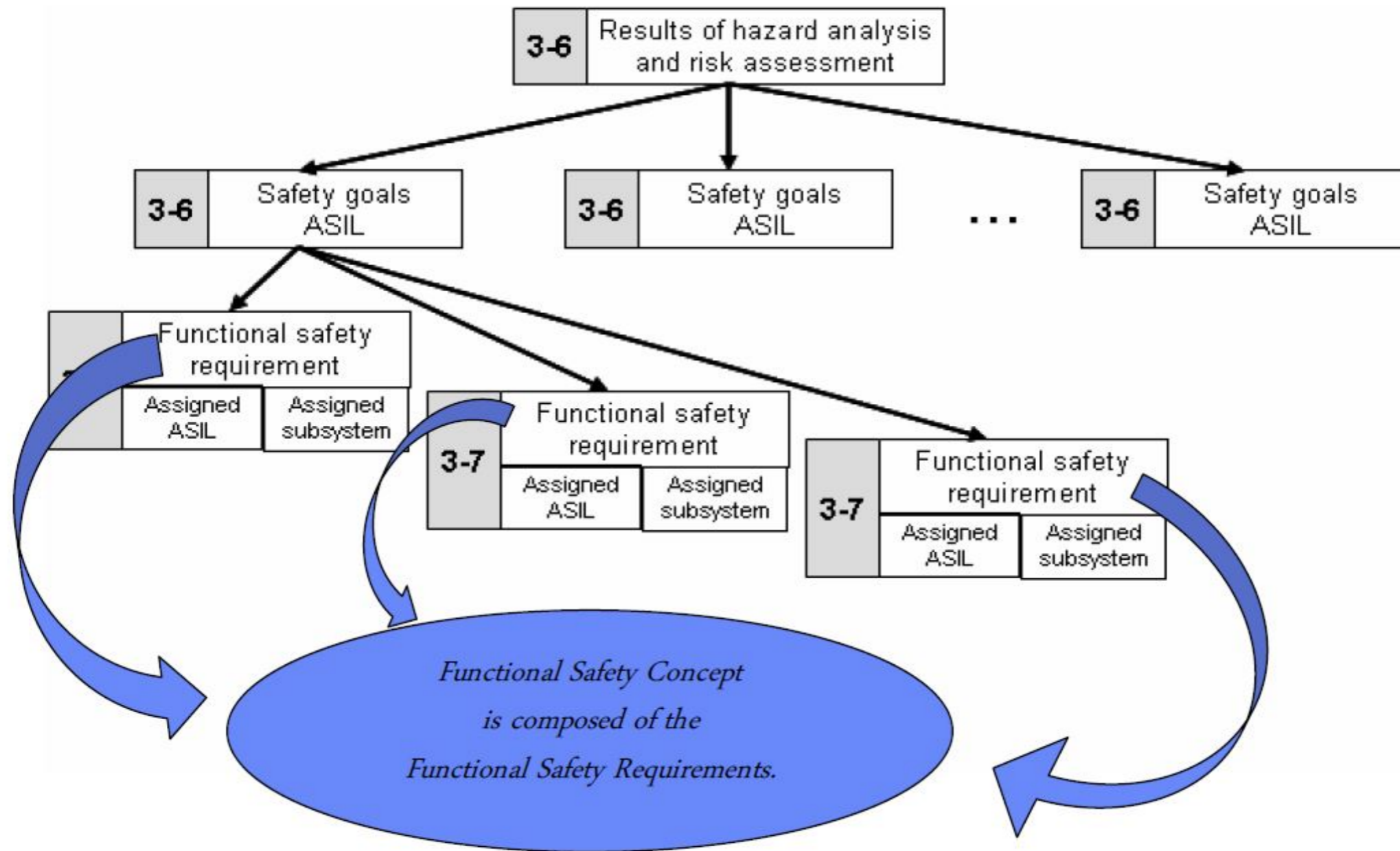
Identify Safety Goals

- Safety Goals are top-level safety requirement as a result of the hazard analysis and risk assessment
- A safety goal is to be determined for each hazardous event evaluated in the hazard analysis
- ASIL determined for the hazardous event is to be assigned to the corresponding safety goal.
- Potential hazard may have more than one safety goal
- If similar safety goals are determined, they can be combined into one safety goal that will be assigned the highest ASIL of the similar goals

Identify Safety Goals - Combination

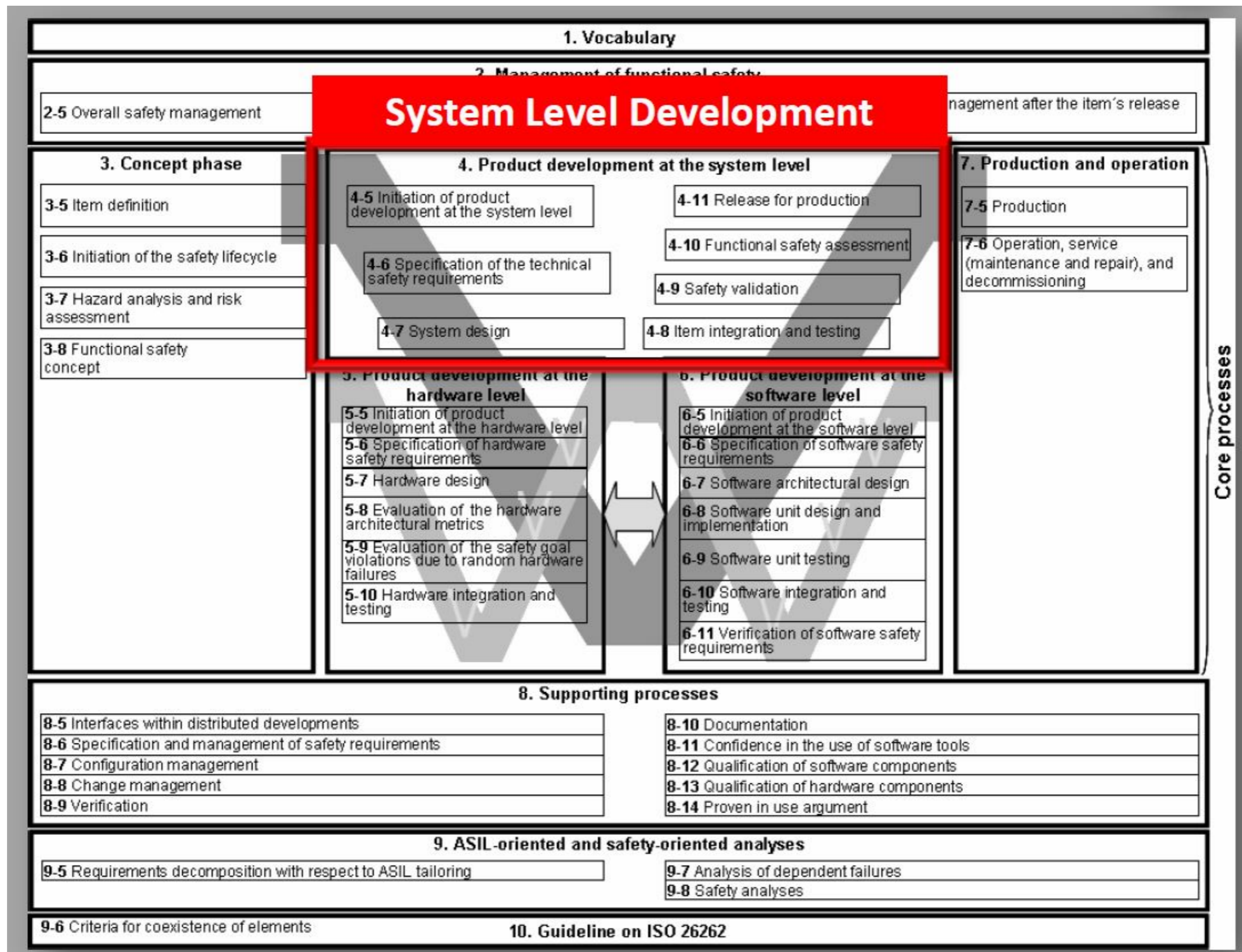


Identify Functional Safety Concept



Source ISO/DIS 26262

System Level Development

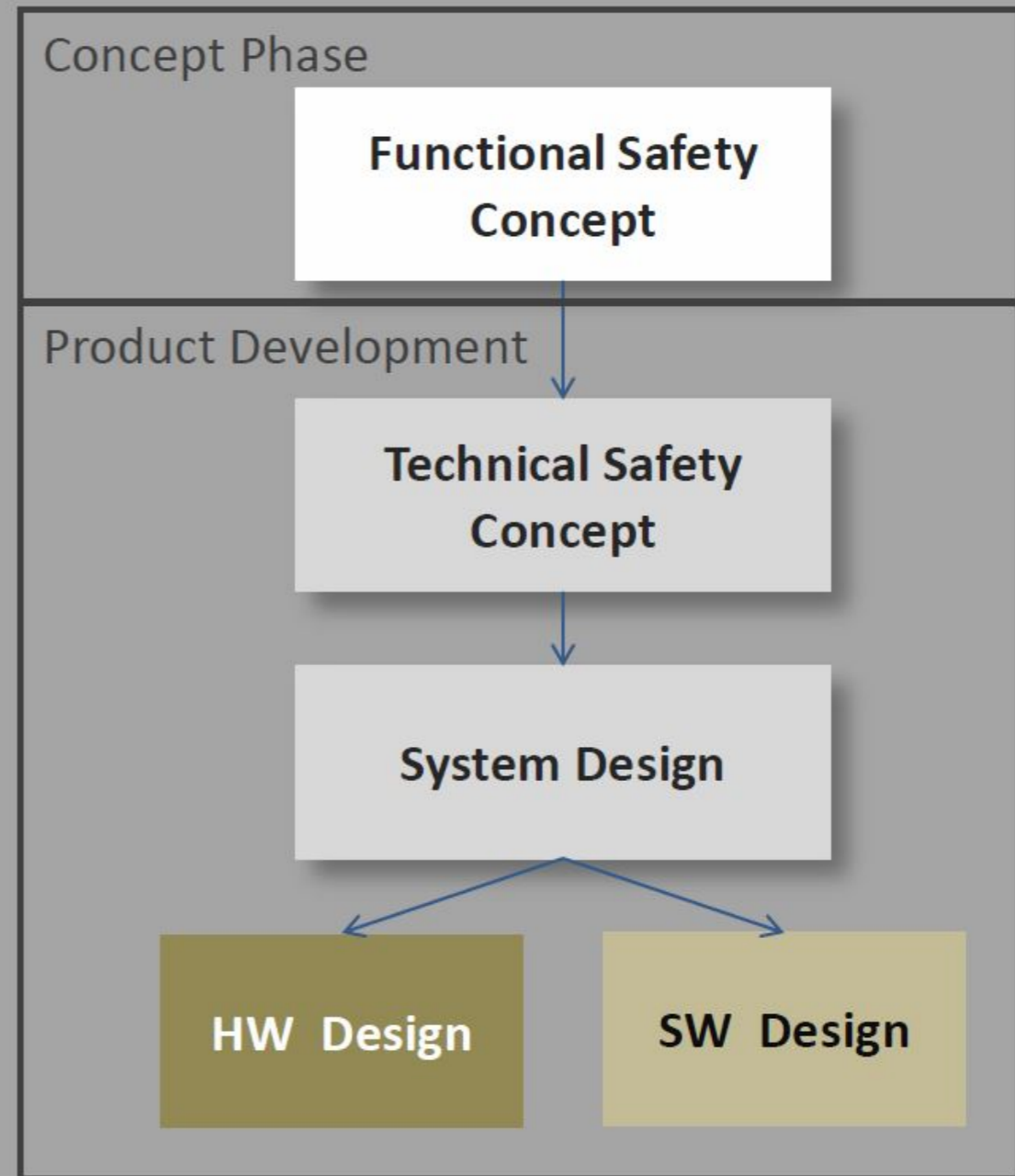


Product Development at System Level

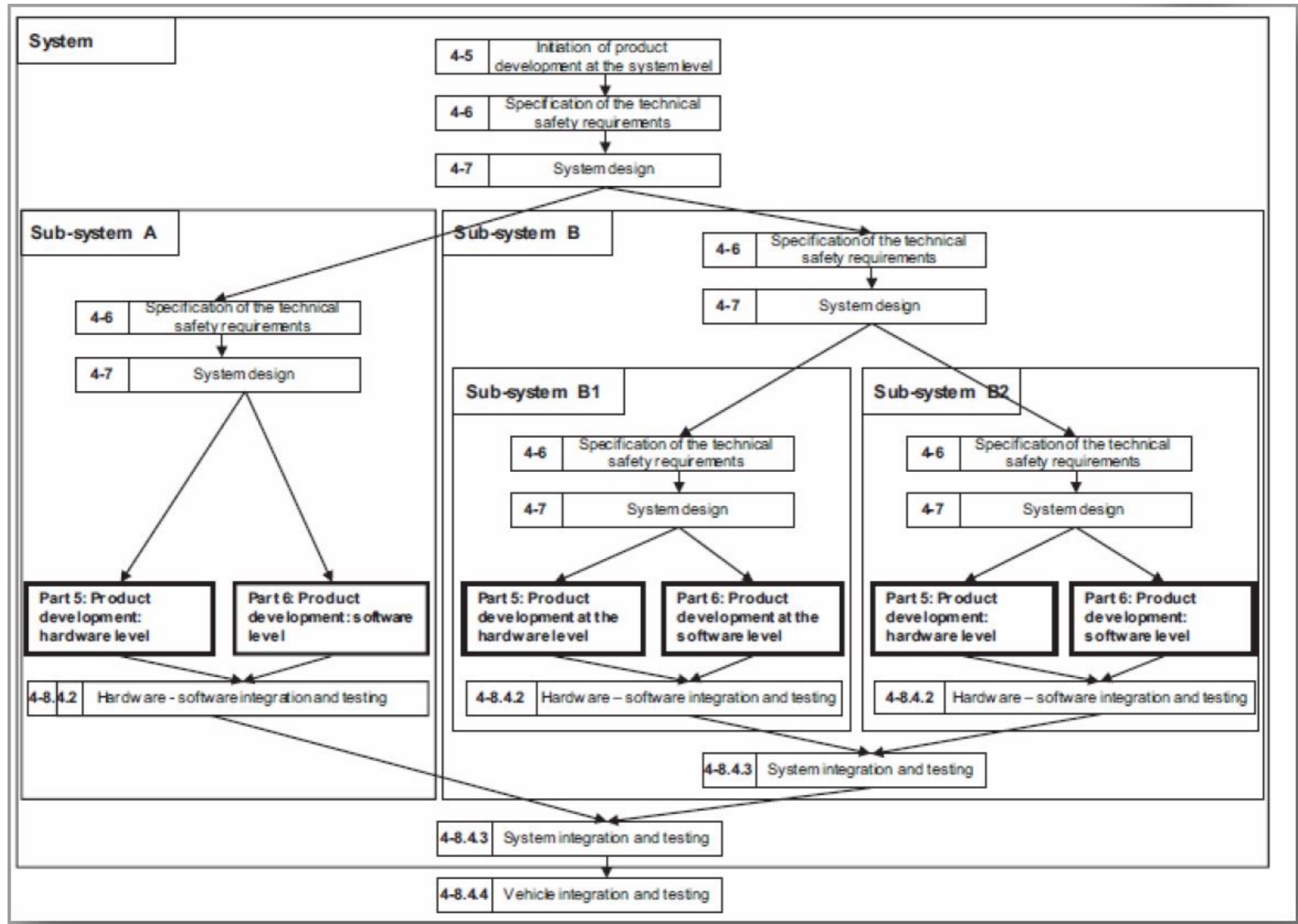
Objectives TSC and System-Design

- Requirements allocation
- Specification of Safety Measures
- Integration
- Validation

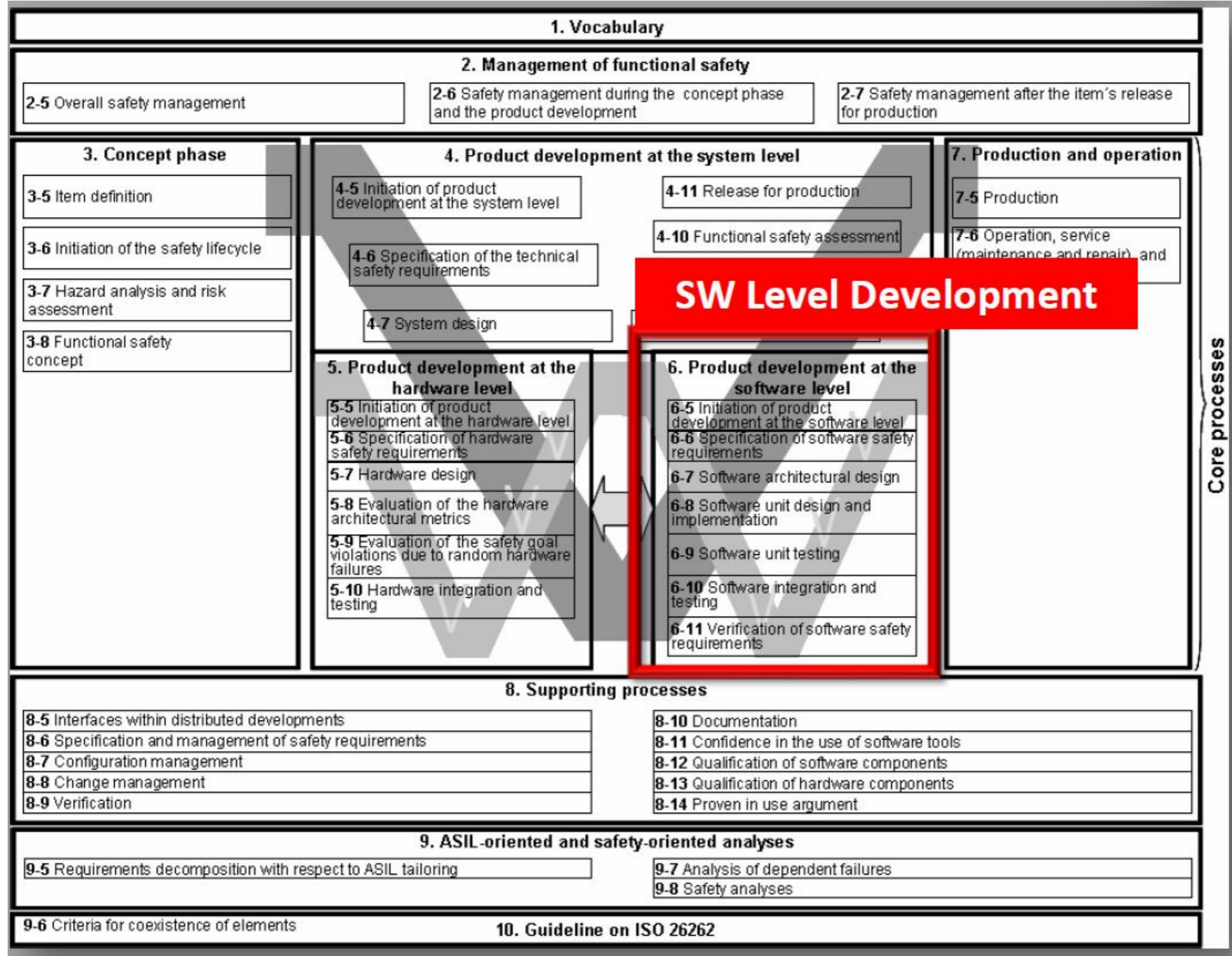
INTEGRITY



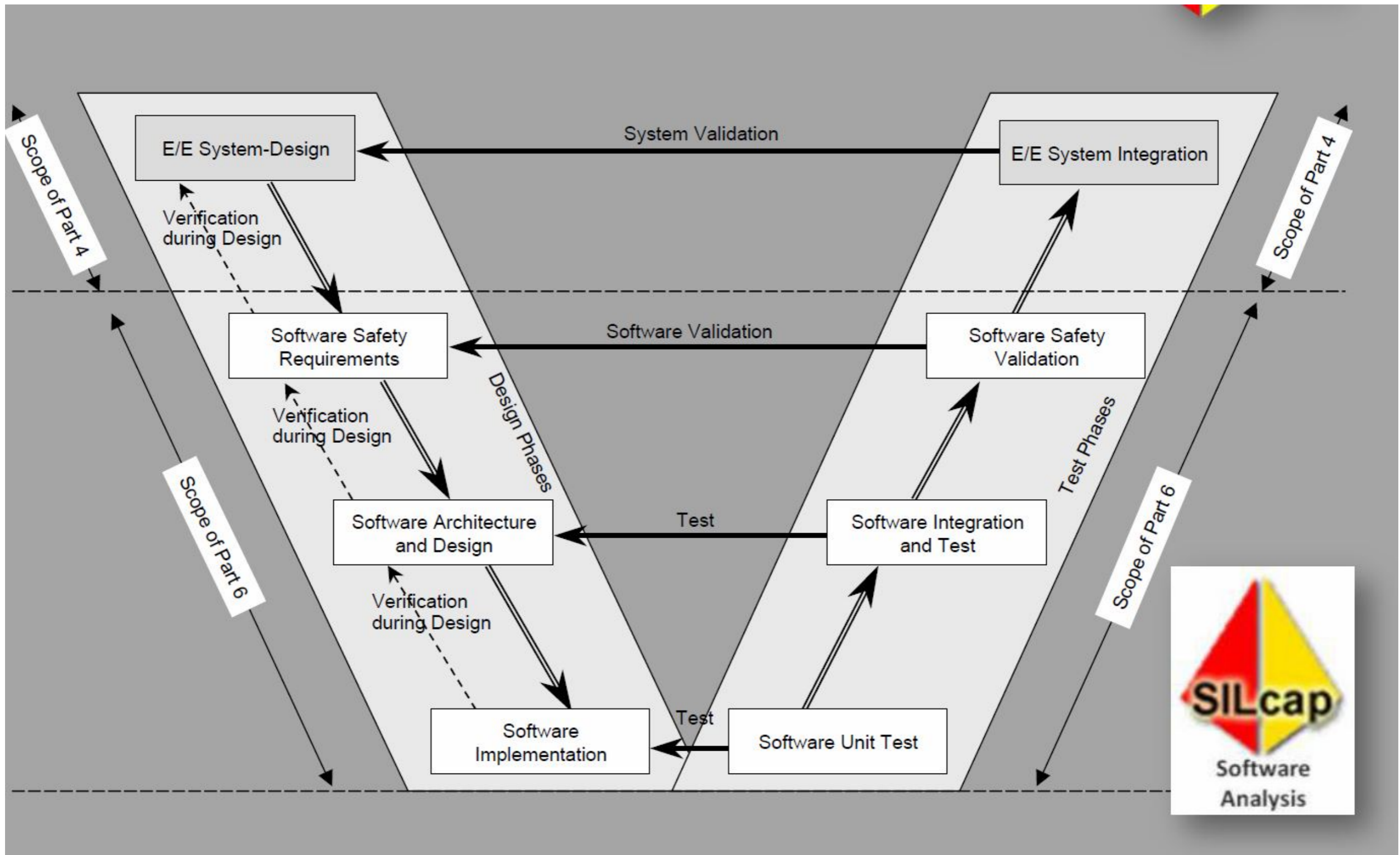
Product Development System Level



ISO 26262 Structure



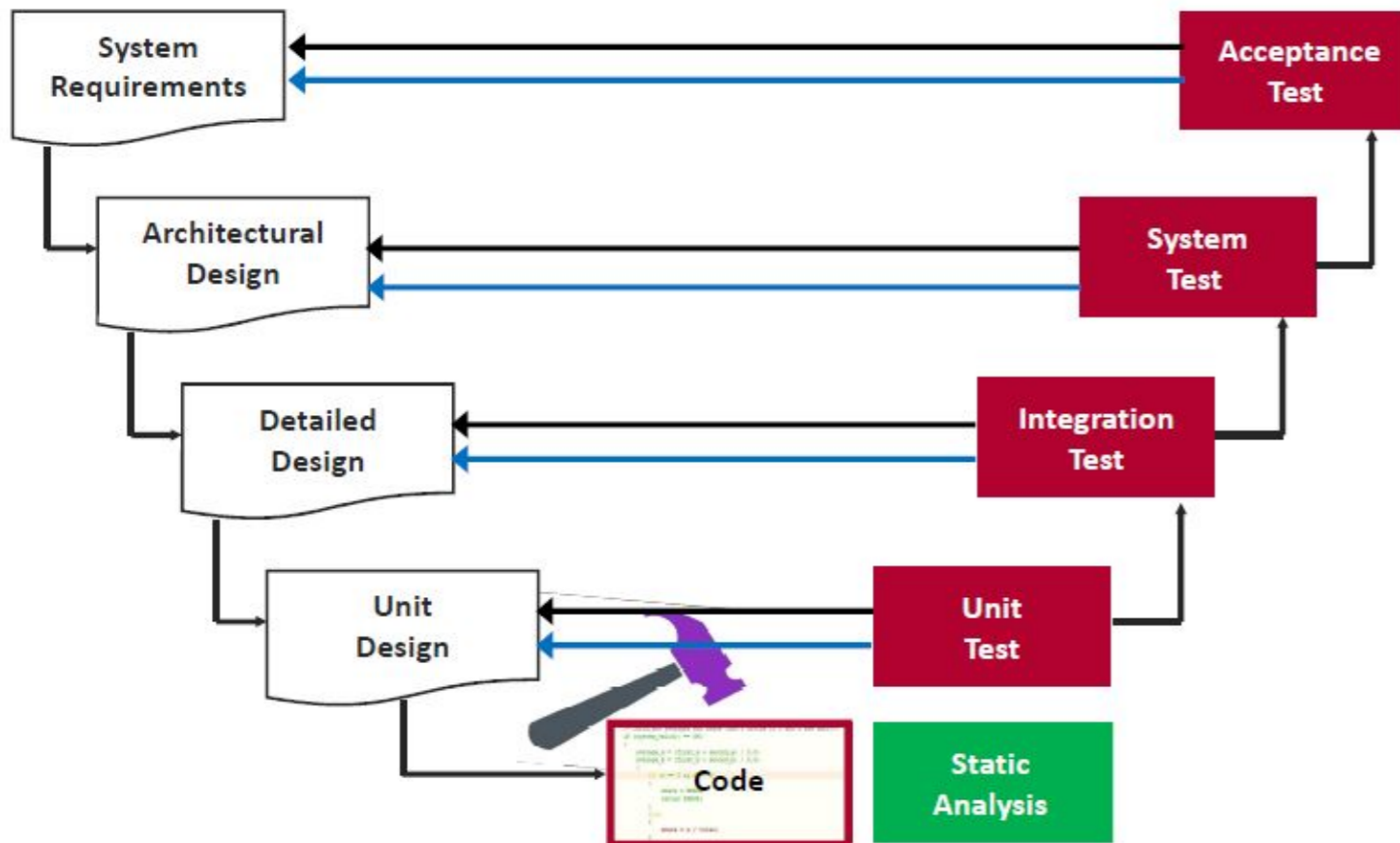
Product Development Software Level



Safety Analyses

- Requirements decomposition w.r.t. ASIL tailoring
- Criteria for Coexistence of elements
- Dependent Failure Analysis
- Safety Analyses

Verification and Validation Order



Verification Order

1 Does code meet the quality standard?

✓ Static Analysis

2 Does code do what it should?

✓ Functional Requirements Testing

✓ Non-functional Requirements Testing







3 Does code not do what it should not?

✓ Robustness Testing

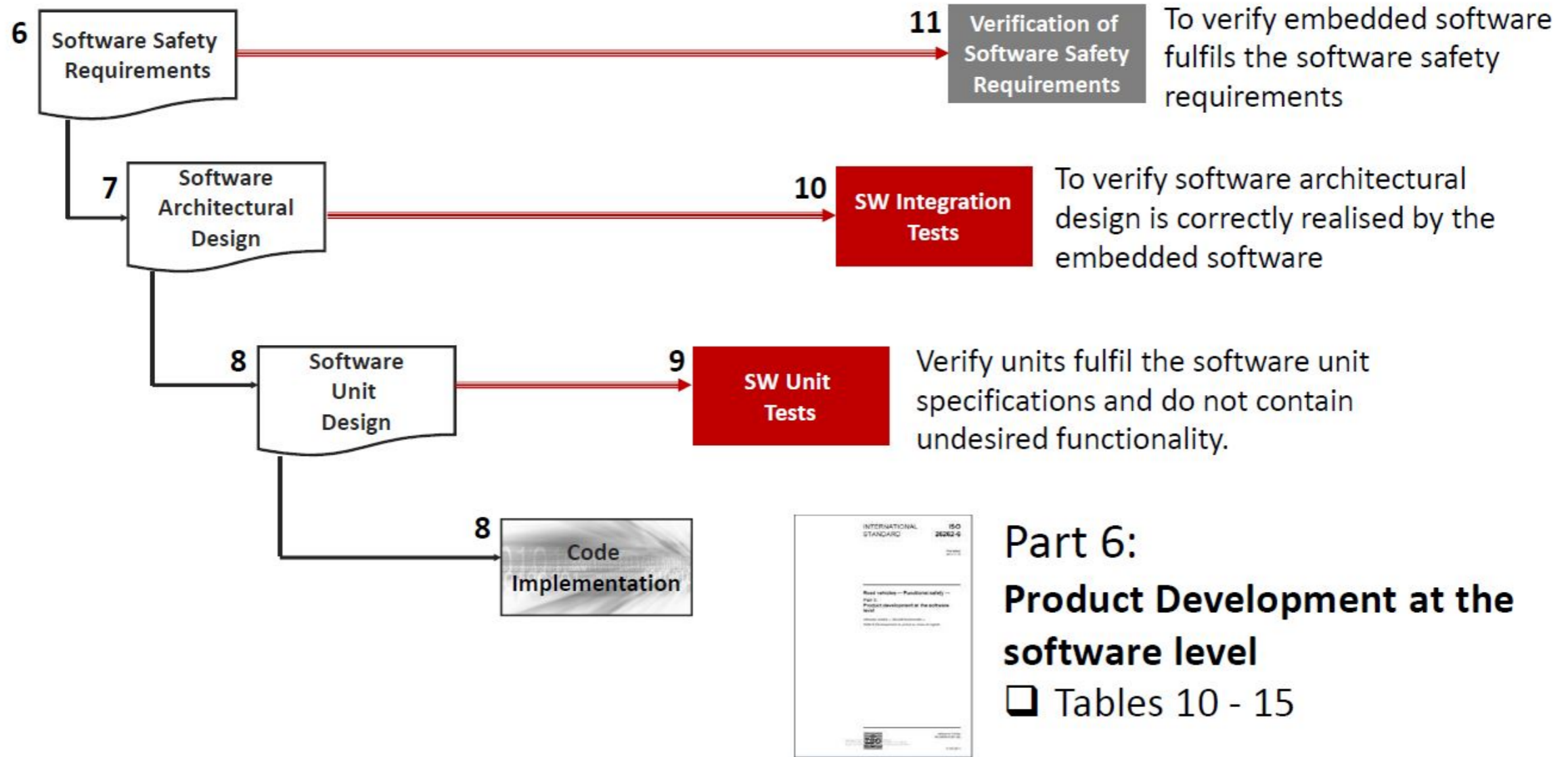
4 Is code tested enough?

✓ Structural Coverage Testing

Testing Methods by Safety Standard


Sector						
Testing Methods	ISO 26262	DO-178B/C	IEC 62304	IEC 61508	EN 50128	IEC 60880
Static Analysis	✓	✓	✓	✓	✓	✓
Requirements based tests	✓	✓	✓	✓	✓	✓
Data / Control Flow interfaces	✓	✓		✓	✓	✓
State Transition		✓		✓		✓
Resource Usage test	✓	✓	✓	✓	✓	✓
Timing tests	✓	✓	✓	✓	✓	✓
Equivalence Partitioning	✓	✓		✓	✓	✓
Boundary Value Analysis	✓	✓		✓	✓	✓
Error Guessing	✓			✓	✓	
Error Seeding / Fault Injection	✓	✓		✓	✓	✓
Structural Coverage testing	✓	✓	✓	✓	✓	✓

ISO 26262



ISO 26262 Dynamic Testing Methods

ISO 26262 Tables 10 & 13 – Methods for software unit & integration testing



Methods	ASIL A	ASIL B	ASIL C	ASIL D
1a. Requirement-based test	++	++	++	++
1b. Interface test	++	++	++	++
1c. Fault injection test*	+	+	+	++
1d. Resource usage test	+	+	+	++
1e. Back-to-back comparison test between model and code (if applicable)	+	+	++	++

* This includes injection of arbitrary faults

(e.g. by corrupting values of variables, by introducing code mutations, or by corrupting values of CPU registers).

9.4.3 / 10.4.3

“The software unit / integration test methods listed in Table 10/13 shall be applied to demonstrate that both the software components and the embedded software achieve:

...

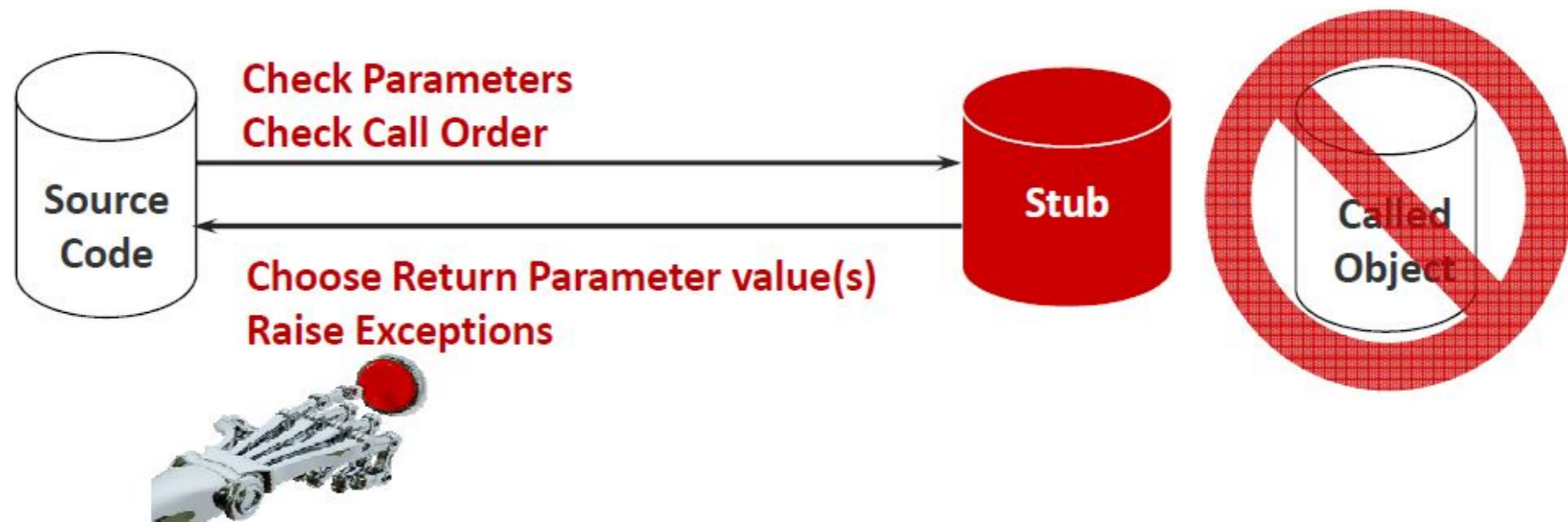
d) robustness;

EXAMPLE Absence of inaccessible software; effective error detection and handling.”

Fault Injection - Simulation

Interface Simulation

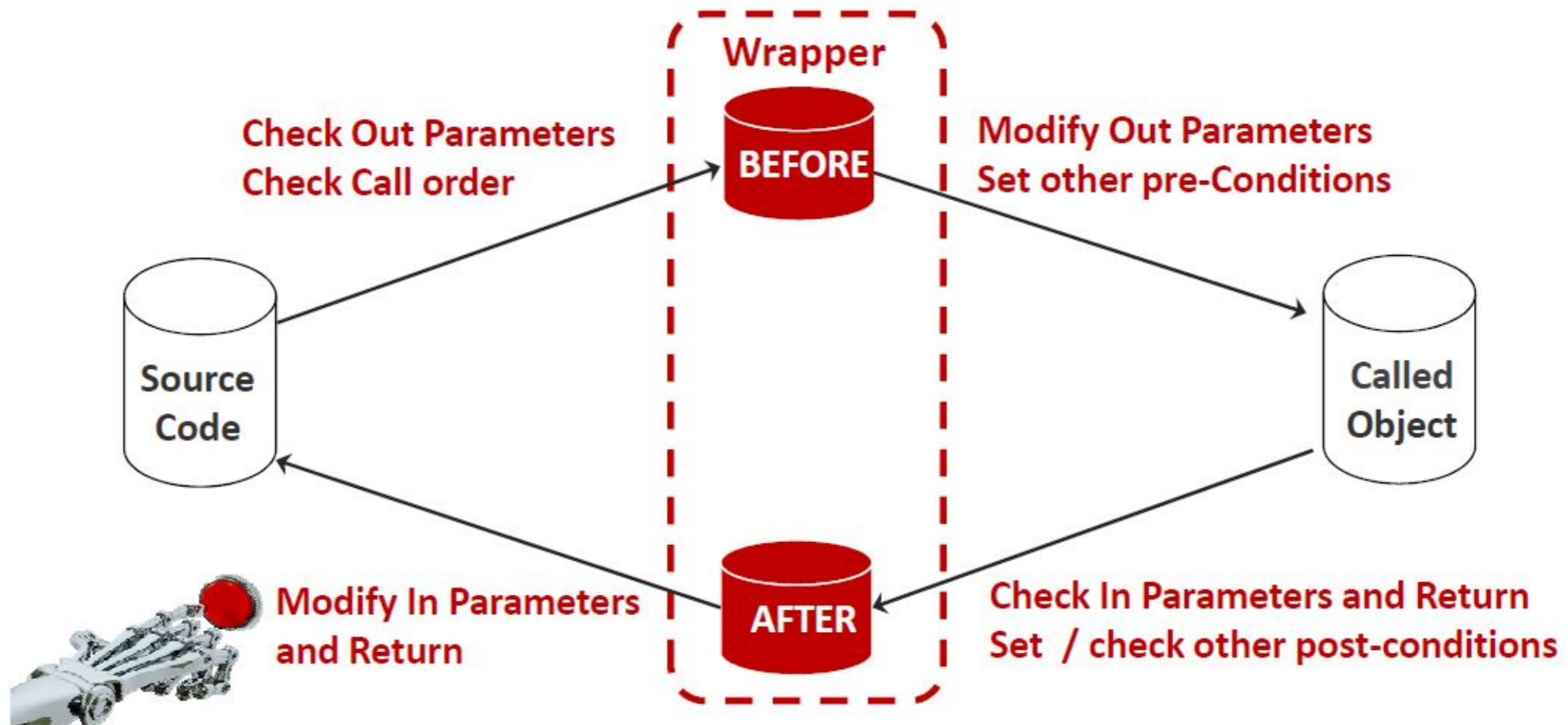
- ✓ A function/method inside test script with programmable instances
- ✓ Stub, (mock, fake, etc) replaces called object interface at link time with dummy code



Fault Injection - Interception


Interface Interception

- ✓ A function/method inside test script with programmable instances
- ✓ Wrapper intercepts specific calls to and uses real called objects at run time



ISO 26262 – Dynamic Analysis – see Testing Lecture

ISO 26262 Tables 12 & 15 – Structural Coverage Metrics at the software unit & architecture levels



Methods	ASIL A	ASIL B	ASIL C	ASIL D
1a. Statement coverage	++	++	+	+
1b. Branch coverage	+	++	++	++
1c. MC/DC Modified Condition/Decision Coverage)	+	+	+	++
1a. Function coverage	+	+	++	++
1b. Call Coverage	+	+	++	++

9.4.5

“To evaluate the completeness of test cases and to demonstrate that there is no unintended functionality, the coverage of requirements at the software unit level shall be determined and the structural coverage shall be measured in accordance with the metrics listed in Table 12. If the achieved structural coverage is considered insufficient, either additional test cases shall be specified or a rationale shall be provided.”

10.4.5

“To evaluate the completeness of tests and to obtain confidence that there is no unintended functionality, the coverage of requirements at the software architectural level by test cases shall be determined. If necessary, additional test cases shall be specified or a rationale shall be provided.”

10.4.6

“To evaluate the completeness of test cases and to obtain confidence that there is no unintended functionality, the structural coverage shall be measured in accordance with the metrics listed in Table 15. If the achieved structural coverage is considered insufficient, either additional test cases shall be specified or a rationale shall be provided.”

ISO 26262 – Static Analysis Verification Methods

ISO 26262 Table 9 – Methods for the verification of software unit design and implementation

Methods	ASIL A	ASIL B	ASIL C	ASIL D
1a. Walk-through	++	+	0	0
1b. Inspection	+	++	++	++
1c. Semi-formal verification	+	+	++	++
1d. Formal verification	0	0	+	+
1e. Control flow analysis	+	+	++	++
1f. Data flow analysis	+	+	++	++
1g. Static code analysis	+	++	++	++
1h. Semantic code analysis	+	+	+	+

8.4.5

“The software unit design and implementation shall be verified in accordance with Clause 9, and by applying the verification methods listed in Table 9 to demonstrate:

...

- d) the compliance of the source code with the coding guidelines (see 5.5.3); and
- e) the compatibility of the software unit implementations with the target hardware

Adherence to MISRA Standards

MISRA – Motor Industry Software Reliability Association

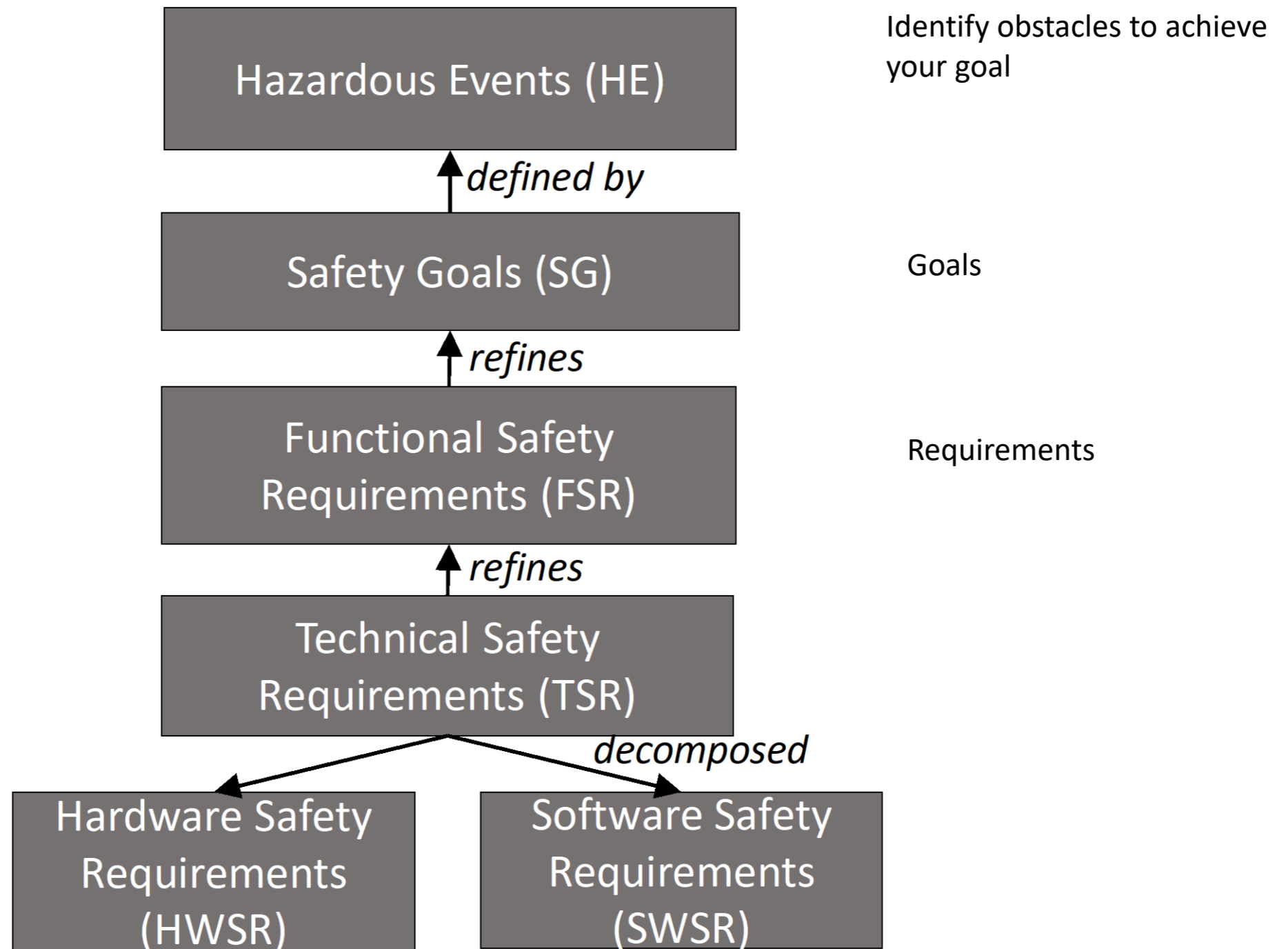
MISRA C

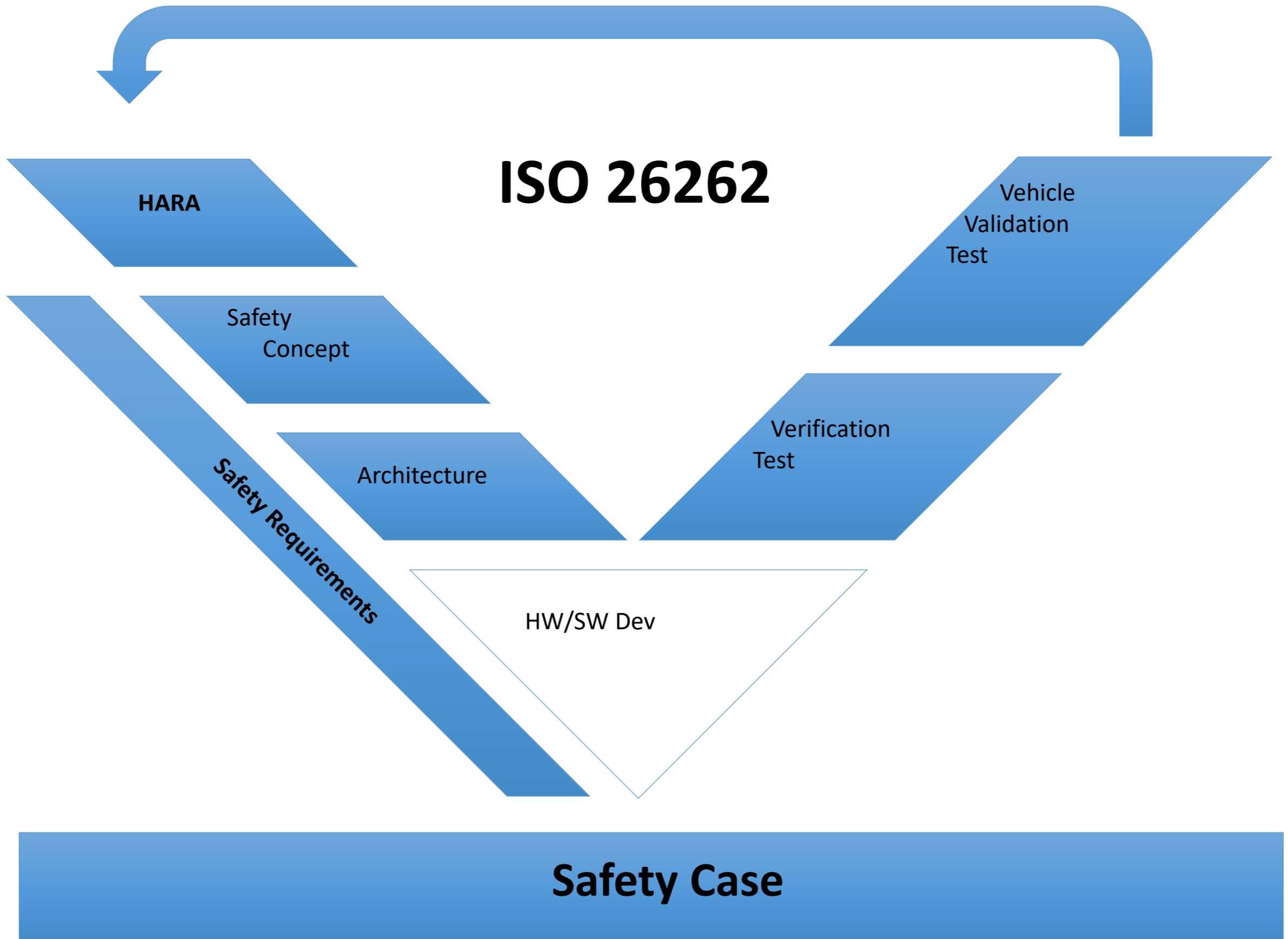
- ✓ 1998 - Guidelines for the use of the C language in vehicle based software
 - ✓ MISRA C:1998 (MISRA C1)
- ✓ 2004 - MISRA C:2004 Guidelines for the use of the C language in critical systems
 - ✓ MISRA C:2004 (MISRA C2)
- ✓ 2013 - MISRA C:2012 Guidelines for the use of the C language in critical systems
 - ✓ MISRA C:2012 (MISRA C3)
 - ✓ 159 rules of which 138 are statically enforceable

MISRA C++

- ✓ 2008 - Guidelines for the use of the C++ language in critical systems
 - ✓ 228 rules of which 219 are statically enforceable

ISO 26262 Recommendation - Summary



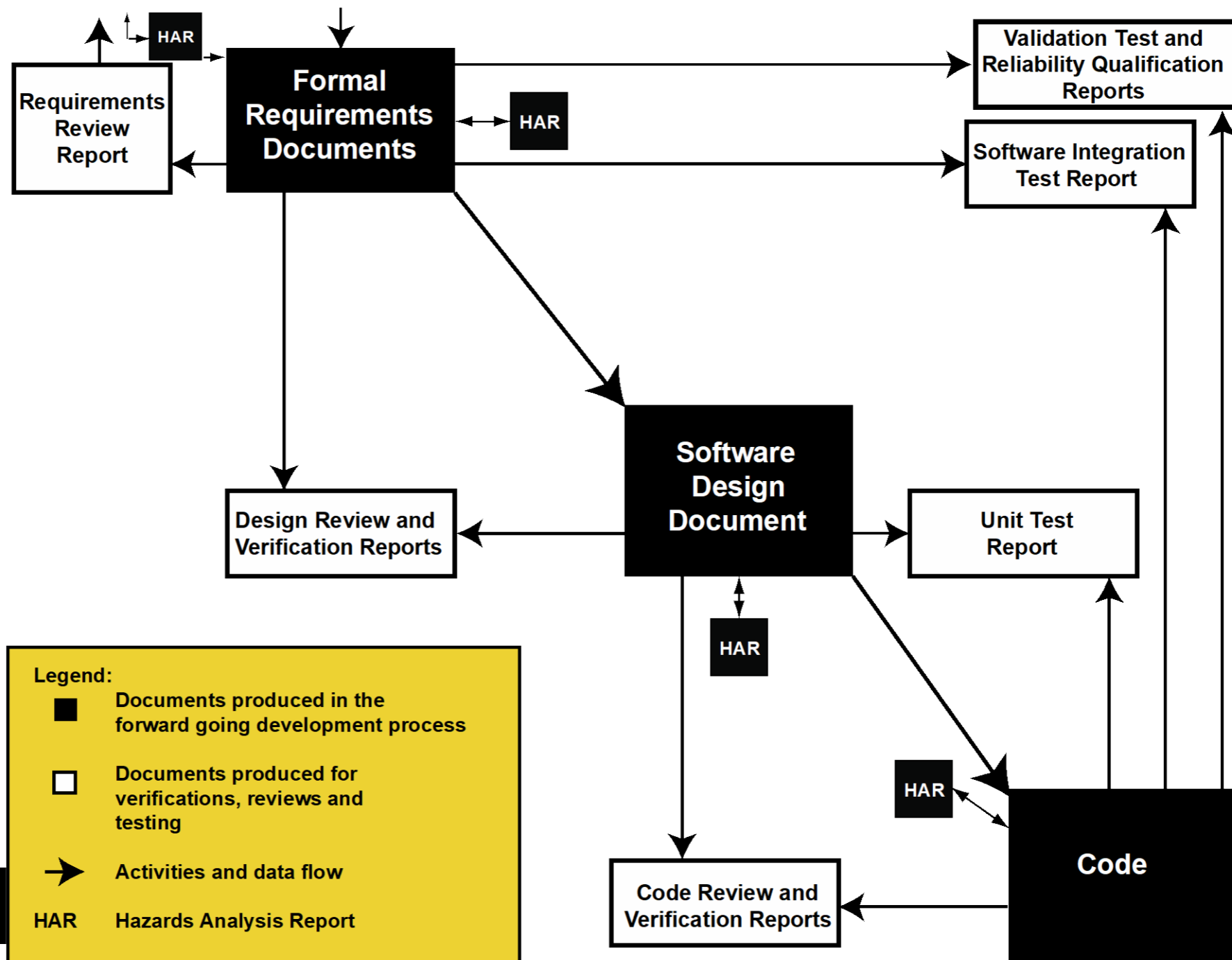


Hazard Analyses

Credit: Alan Wassiyong, McMaster University

How do we construct correct, safe and reliable software?

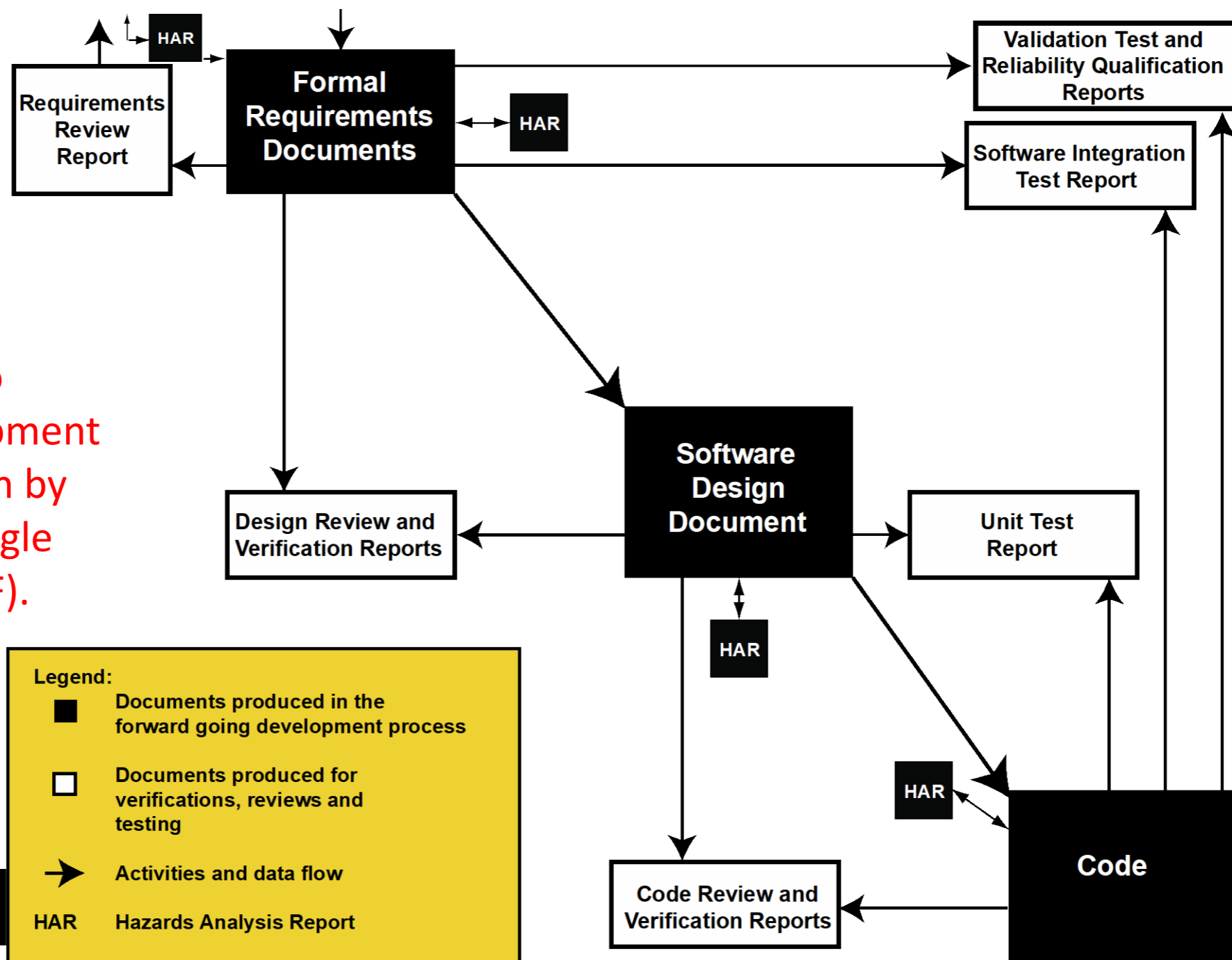
- Rigorous software engineering



Hazard analysis is iterative over the life of the project!

How do we construct correct, safe and reliable software?

- Rigorous software engineering



We have a defence-in-depth approach to the software development process itself – driven by identification of a single point of failure (SPOF).

What is a 'hazard'?

- It's a property or condition in the system together with a condition in the environment that has the potential to cause {harm or damage} = loss [Nancy Leveson]

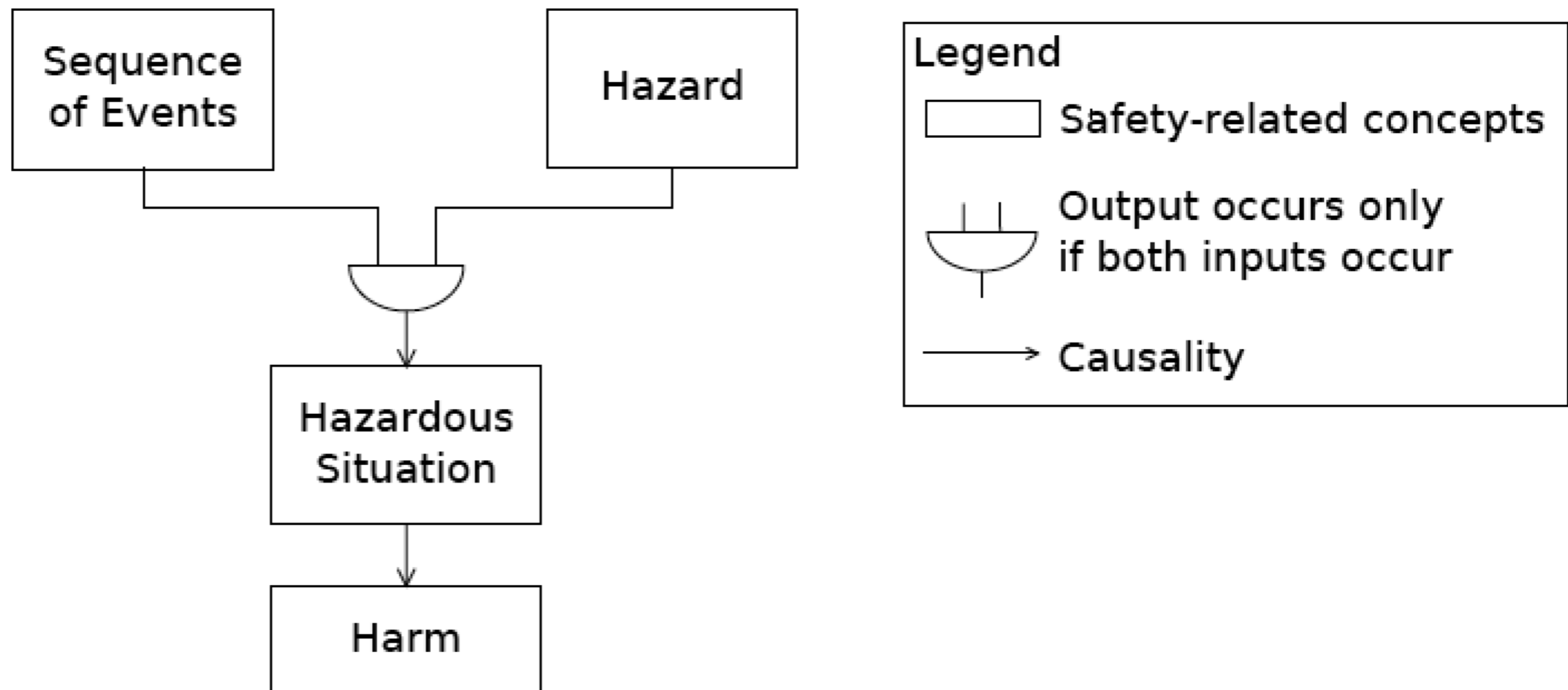
What is 'hazard analysis'?

- Document hazards
- Document hazard controls – how to mitigate each hazard
- Hazard analysis introduces a different way of thinking about our systems and processes
 - There is lots of anecdotal evidence to suggest that we find and mitigate more hazards when we follow some systematic process designed to perform the hazard analysis
- Hazard analysis is mandatory in safety critical domains
 - Nuclear, medical, chemical process industry, ...
- And part of following standards in automotive domain

HA Flavours

- Lots to choose from
 - Hazard & Operability Study (HAZOPS)
 - Fault Tree Analysis (FTA)
 - Failure Modes and Effects Analysis (FMEA)
 - Failure Modes and Effects Criticality Analysis (FMECA)
 - Functional Resonance Accident Model (FRAM)
 - System-Theoretic Process Analysis (STPA)

Concept – ISO 14971

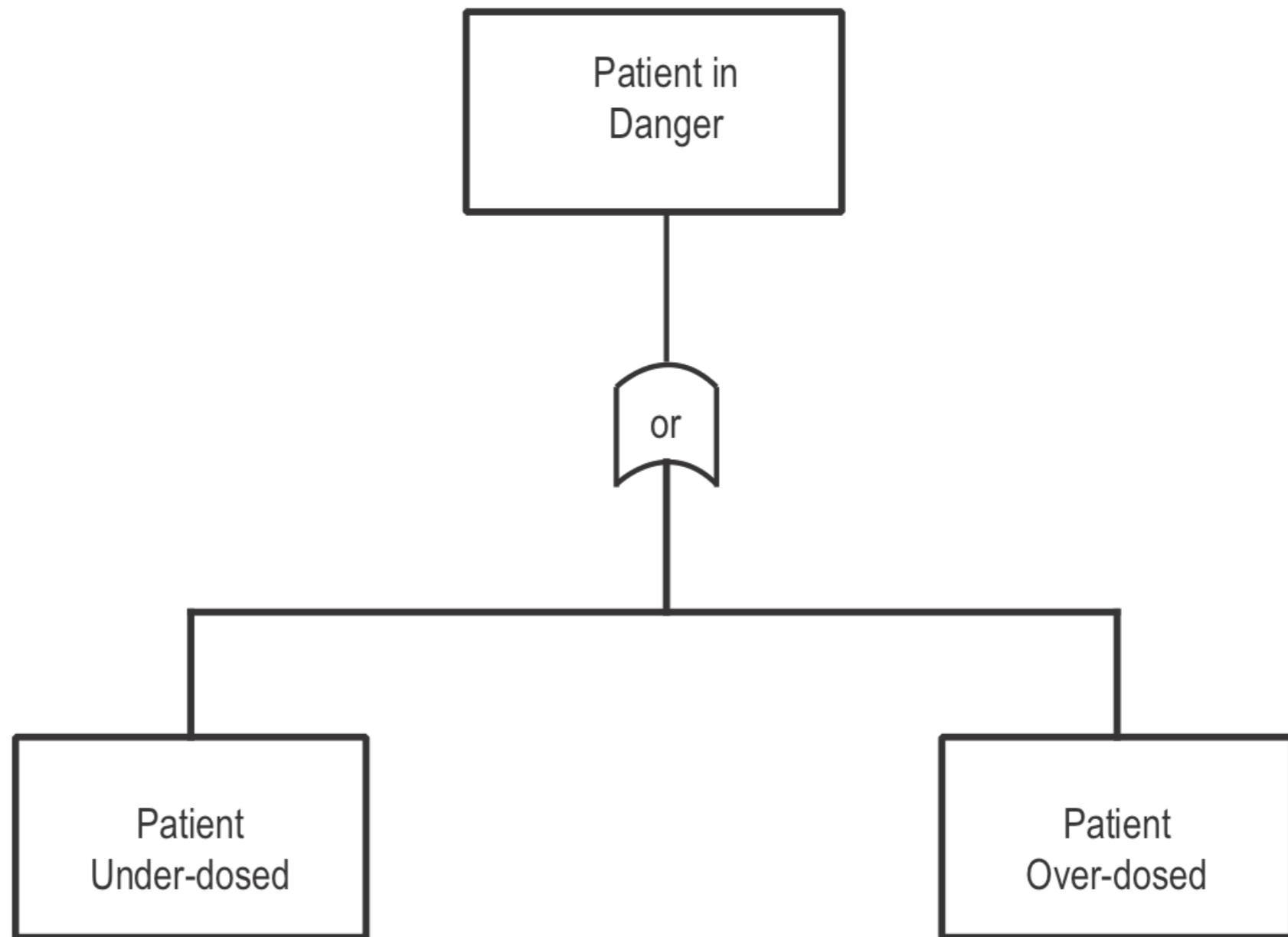


FTA

- Top down
- Process
 - Define the TOP event to be analyzed
 - Identify the lower level events which may lead to the TOP event and complete the gates
 - (optional) Find minimal cut sets (qualitative)
 - (optional) Calculate the failure rate of TOP event (quantitative)
- Cut set = events that together cause the top event (sometimes called a fault path)
- Good for identifying single points of failure

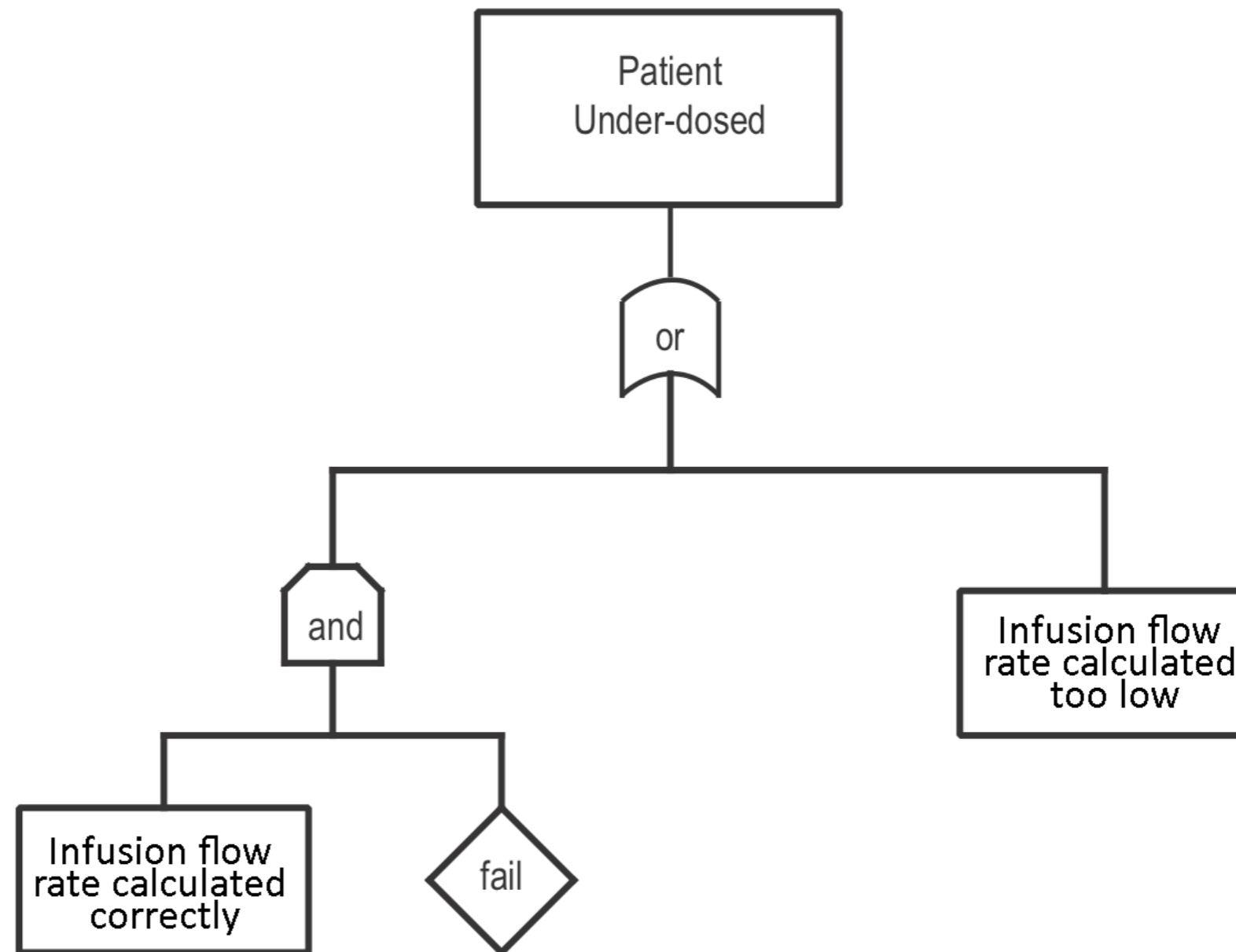
Example FTA: Insulin Pump

- Insulin Pump Extract – top level FTA



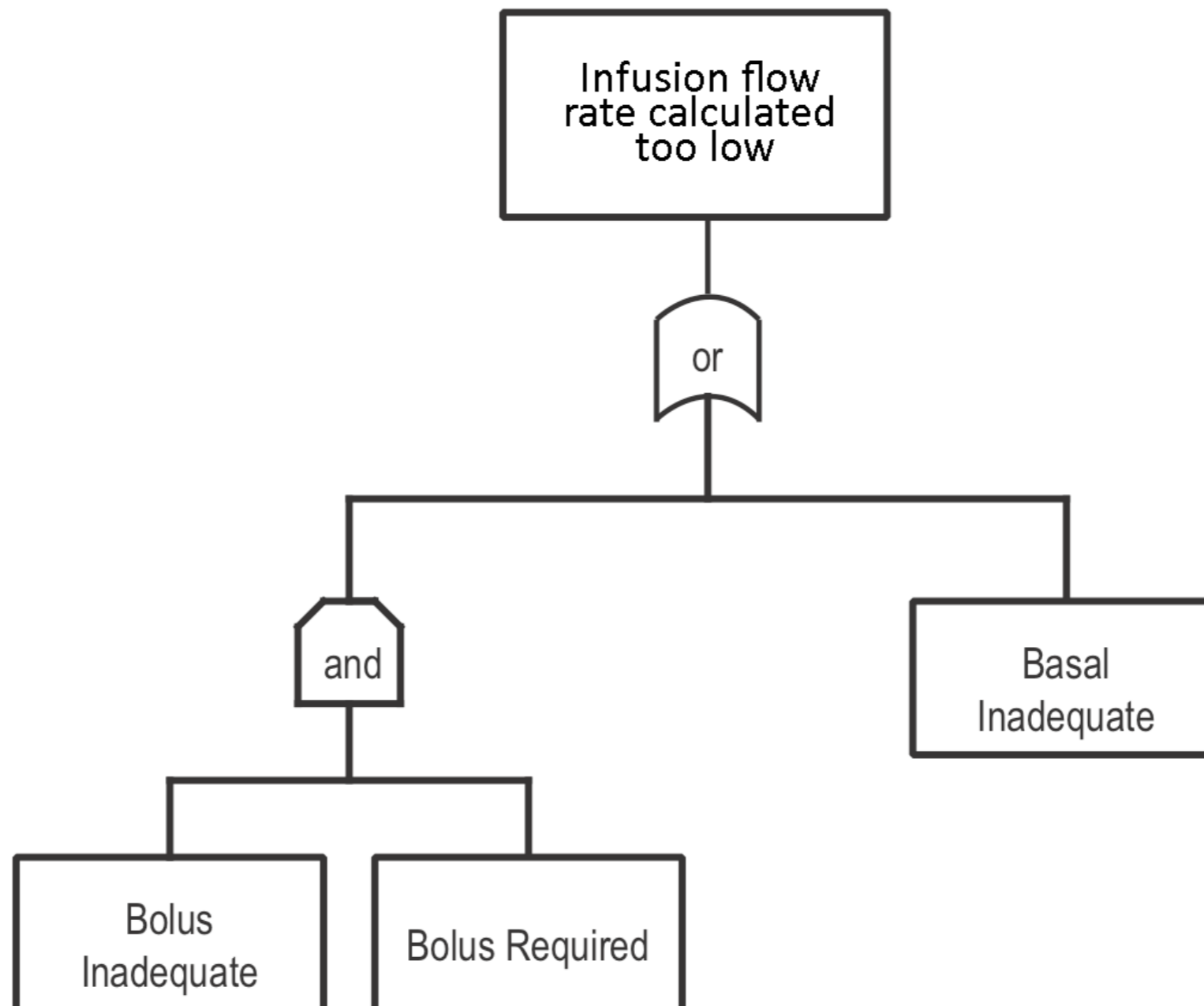
Example FTA: Insulin Pump

- Insulin Pump Extract – FTA - expand under-dosed



Example FTA: Insulin Pump

- Insulin Pump Extract – FTA - expand too low



FMEA

- Bottom up approach – need to know all details
- Was not designed to consider combination failure initiating events
- Performed on both processes and products
- Many people use RPN to prioritize – so mitigate only those hazards with $RPN > x$
 - $RPN = \text{Risk Priority Number}$
 $= \text{Severity} * \text{Probability of Occurrence} * \text{Detection Rating}$

Example Process FMEA

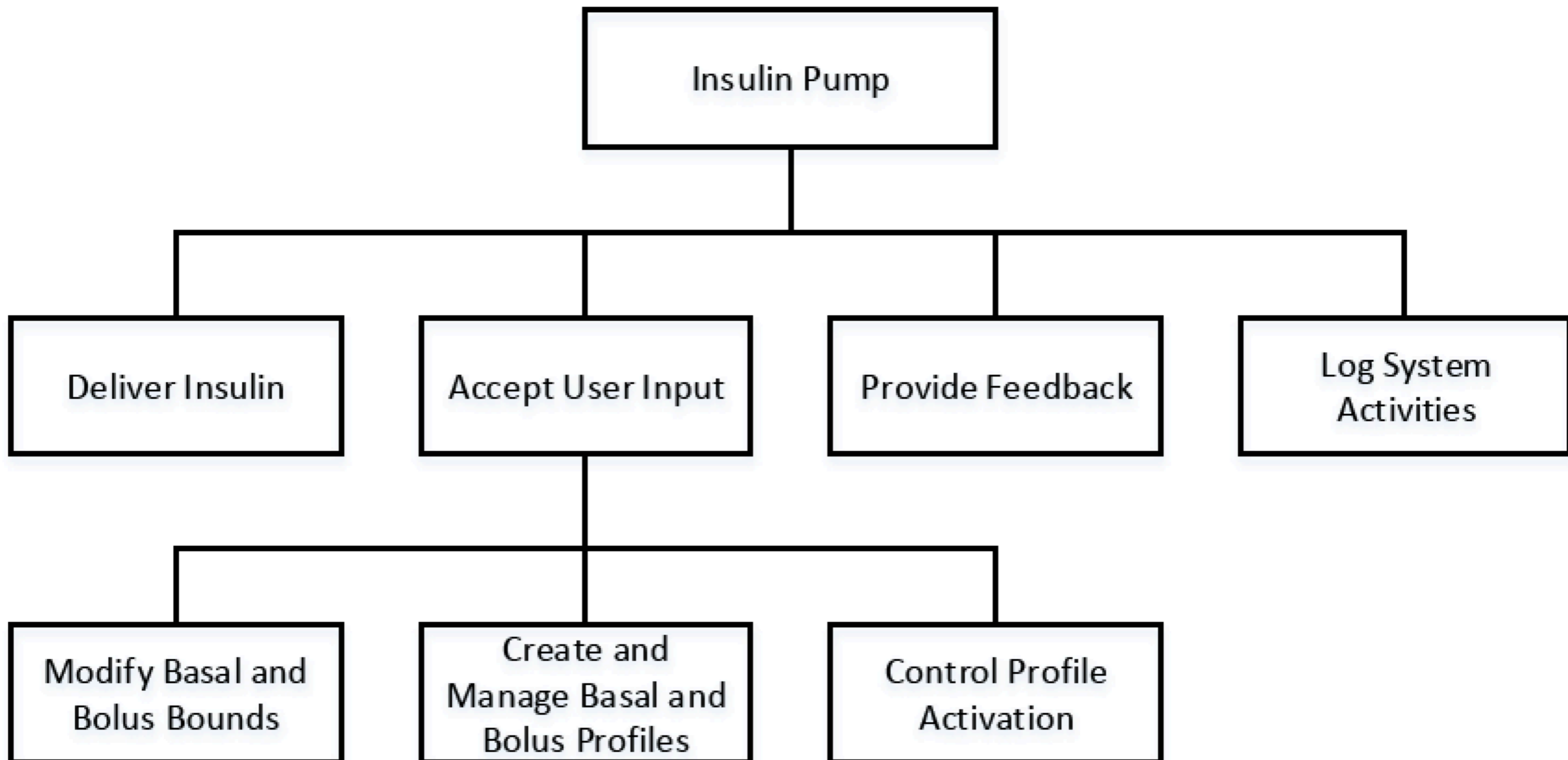
TABLE II. FMEA for conventionally fractionated and hypofractionated DMLC tracking-based delivery. For each failure mode, numeric values are assigned to the probability of occurrence (O), severity of effect (S), and detectability of failure (D), and a RPN is calculated.

Process step	Failure mode	Possible causes of failure	Effect of failure	Conventional fractionation				Hypofractionation			
				O	S	D	RPN_C	O	S	D	RPN_H
1. Position Monitoring (PM) system estimates real-time target position	i. Target moves outside spatial tolerance set by PM system, but beam-hold is not asserted	i. (a) Beam-hold not sent by monitoring system to tracking system	Dose error >15%	3.4	5.7	4.0	78	3.3	7.1	3.6	84
2. Tracking system receives real-time target position and recalculates MLC leaf positions as a function of dose fraction and target position	ii. Communication with PM system is lost, i.e., target position is no longer current but beam-hold not asserted	ii. (a) PM system failure	Dose error >30%	3.7	7.1	2.6	68	3.7	8.4	2.6	81
	iii. Error in coordinate system conversion	ii. (b) Data transfer cable(s) physically disconnected	Dose error >60%	1.7	8.6	4.3	63	1.7	9.3	3.9	61
	iv. Optimal leaf-fitting not achieved	iii. (a) System installation error iii. (b) Hardware/software changes	Dose error >10%	5.4	4.0	6.7	146	5.3	5.0	6.7	177
3. Tracking system checks if fluence map is completely within field	v. Beam-hold not asserted when fluence map is partially or completely under one or more jaws	iv. (a) Complex motion+highly modulated field	Dose error >20%	2.9	4.9	3.7	52	2.6	6.1	3.9	61
4. MLC controller actuates leaf motion. If leaves are within tolerance, linac delivers dose, else beam is held off.	vi. System latency outside expected range	v. (a) Software crash or failure v. (b) Jaws outside tolerance	Dose error >5%	4.0	3.0	6.0	72	4.0	4.1	6.3	104
	vii. Too many beam-holds. Efficiency drops below desired threshold	vi. (a) Hardware and/or software changes vi. (b) Changes in network connection speed vii. (a) MLC leaves cannot keep up with target motion	Efficiency <70%	7.0	2.3	1.4	23	6.0	2.9	1.6	27

Taken from (Sawant, Dieterich, Svatos, & Keall, 2010)

Example FMEA: Insulin Pump

Functional decomposition of the insulin pump



Example FMEA: Insulin Pump

Failure Mode and Effects Analysis							
System: Generic Insulin Infusion Pump Subsystem: N/A Phase/Mode: System Requirements							
Design Function	Failure Modes	Effects of Failure	Causes of Failure	Detection	Recommended Action	SR	Ref.
Deliver insulin	Does not deliver enough or fails to deliver any insulin	Hyperglycemia in patient	<ul style="list-style-type: none"> a. Insulin delivery mechanism failure; b. Infusion delivery control failure; c. Incorrect basal or bolus bounds settings; d. Incorrect basal or bolus profile settings; e. Incorrect profile activation; f. Controller failure; g. Not enough insulin available; 		<ul style="list-style-type: none"> a. Include flow meter in system design to measure insulin output flow and check against controlled output, report system error in case of malfunction. b. Each component should verify the validity of the signal it is receiving. c. Refer to H2. d. Refer to H3. e. Refer to H4. f. Require system to detect controller failure (e.g. use watchdog to detect system lockup). g. Require device to check insulin levels before and during insulin delivery and notify user when insulin levels are low. 	<ul style="list-style-type: none"> a:SR-1 b:SR-2 f:SR-3 g:SR-4 	H1-1
	Delivers too much insulin	Hypoglycemia in patient	<ul style="list-style-type: none"> a. Insulin delivery mechanism failure; b. Infusion delivery control failure; c. Incorrect basal or bolus bounds settings; d. Incorrect basal or bolus profile settings; e. Incorrect profile activation; f. Controller failure. 		<ul style="list-style-type: none"> a. Same as H1-1a. b. Same as H1-1b. c. Refer to H2. d. Refer to H3. e. Refer to H4. f. Same as H1-1f. 	<ul style="list-style-type: none"> a:SR-1 b:SR-2 f:SR-3 	H1-2
	Takes too long to deliver insulin	Hyperglycemia in patient	<ul style="list-style-type: none"> a. Insulin Delivery Mechanism Failure; b. Infusion delivery control failure; c. Incorrect basal or bolus bounds settings; d. Incorrect basal or bolus profile settings; e. Incorrect profile activation; f. Controller failure. 		<ul style="list-style-type: none"> a. Same as H1-1a. b. Same as H1-1b. c. Refer to H2. d. Refer to H3. e. Refer to H4. f. Same as H1-1f. 	<ul style="list-style-type: none"> a:SR-1 b:SR-2 f:SR-3 	H1-3

FMEA → Safety Requirements

1.1 Safety Requirements

Using the results of the FMEA, we can come up with the following safety requirements for our system in order to mitigate the identified hazards. Accuracy, tolerance, timing, and other details for each requirement need to be determined.

SR-1:

The device shall be able to detect insulin delivery mechanism failure by measuring the output insulin flow and verifying the correctness of the desired output flow with the measured output flow. An alarm/error shall be produced if the measured output flow is not within TBD insulin units/time units.

Rationale: A problem with the insulin delivery mechanism, such as a blockage, could result in an incorrect amount of insulin being delivered to a patient.

Associated Hazards: H1-1a, H1-2a, H1-3a.

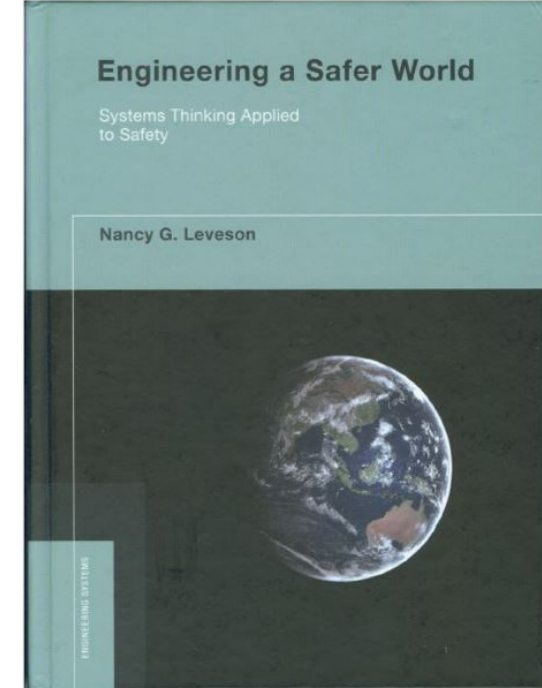
SR-2:

Each component shall verify the validity of the signal it is receiving.

Rationale: A problem with an input signal or communication between components could result in incorrect operation and undesired insulin delivery.

Associated Hazards: H1-1b, H1-2b, H1-3b, H2-3b, H2-4b, H3-1b, H3-2b, H4-1b, H4-2b, H5-1b, H5-2b, H5-3b.

Motivation for STPA



- Many failures are traced back to interaction failures – components work well, but put them together in a specific environment and we get unanticipated failures
- STPA may help us find those hazardous interactions in the environment, for example
- Question: Why do so many cars in Canada not have their rear lights on at night?

STPA

From Nancy Leveson

- Four categories of control actions to consider
 - A control action required for safety is not provided or is not followed
 - An unsafe control action is provided that leads to a hazard
 - A potentially safe control action is provided too early, too late, or out of sequence
 - A safe control action is stopped too soon (for a continuous or non-discrete control action)

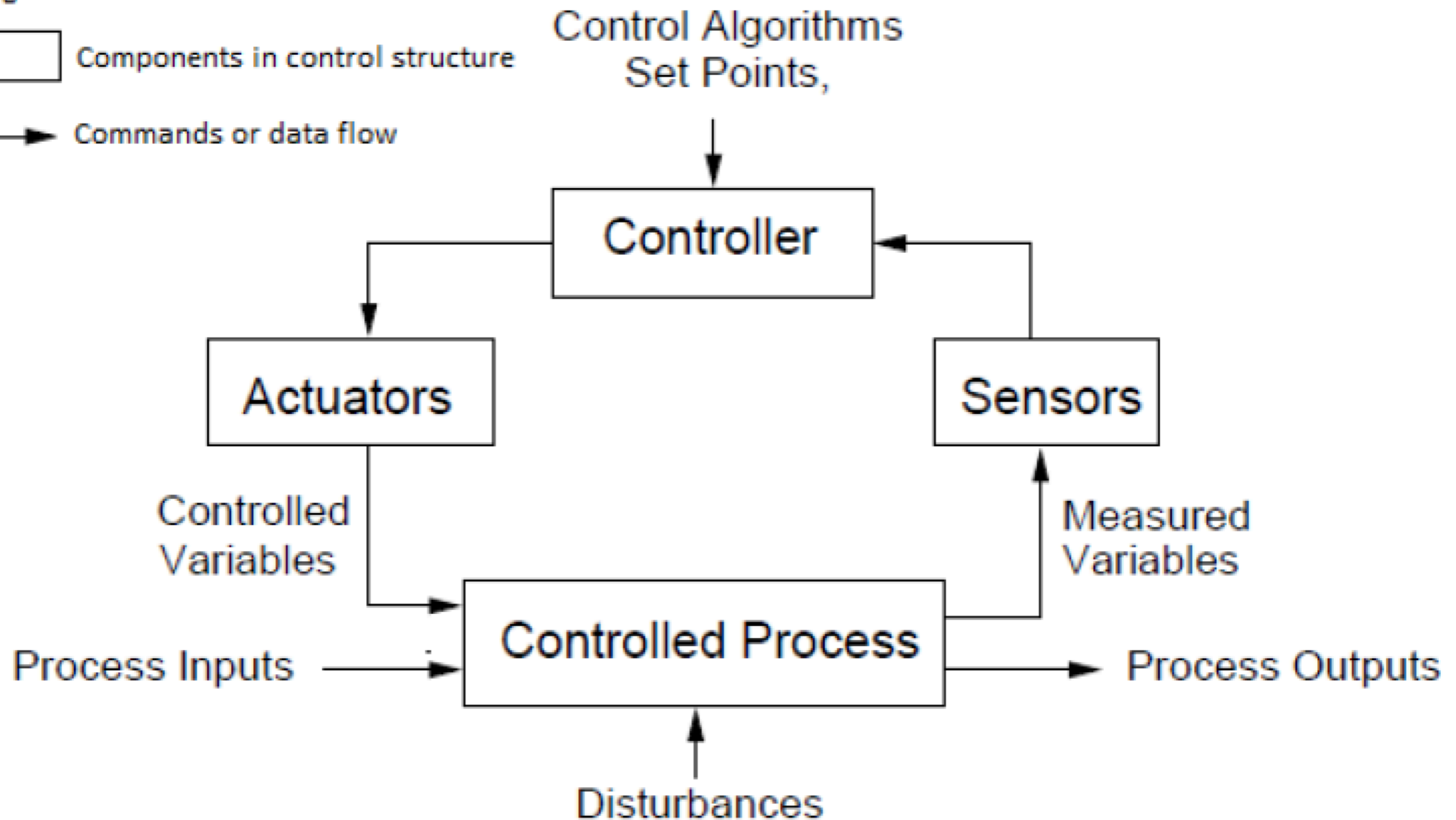
!! Some idea of “completeness” that helps us consider “all” possibilities

STPA

Legend:

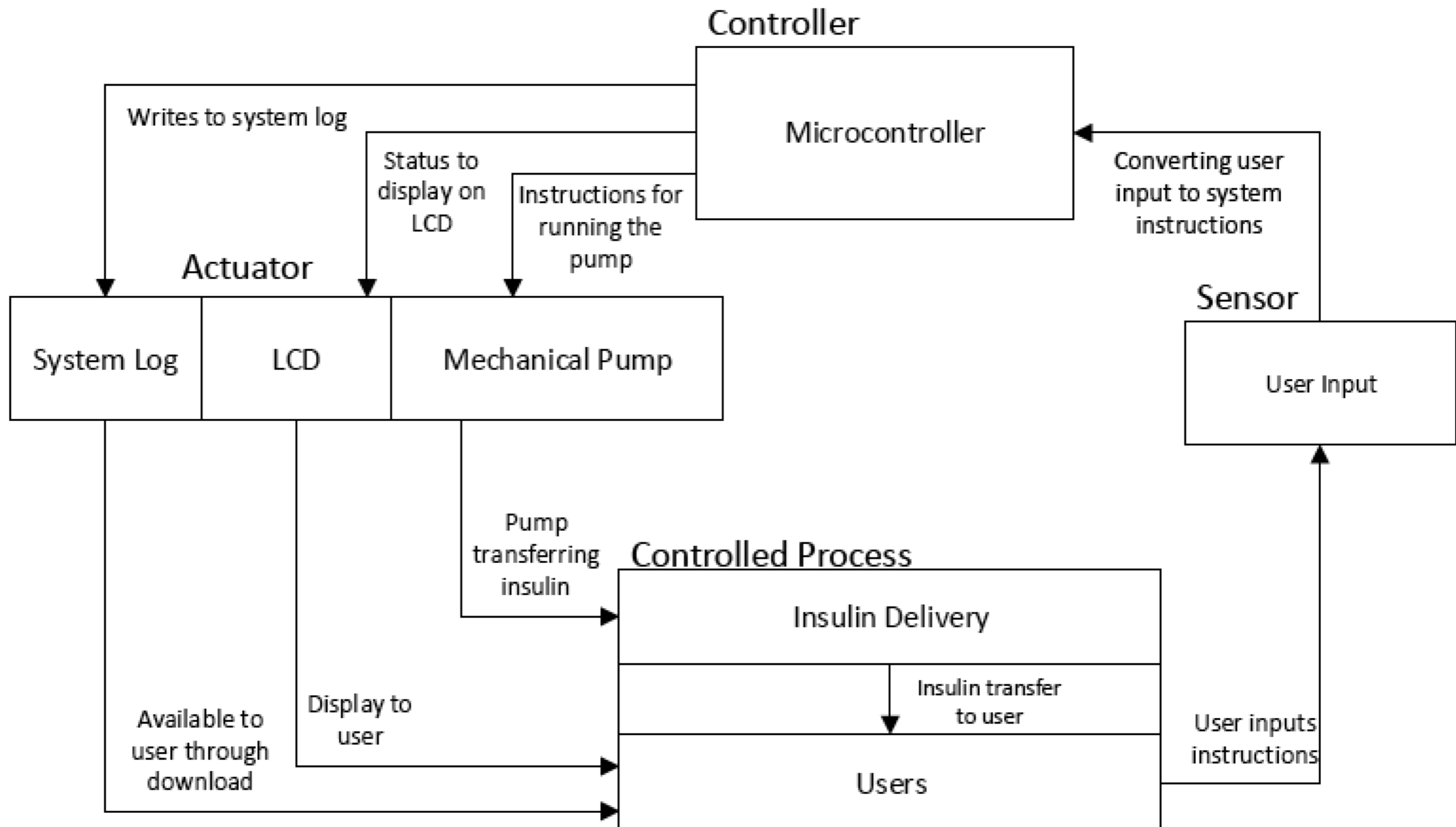
□ Components in control structure

→ Commands or data flow



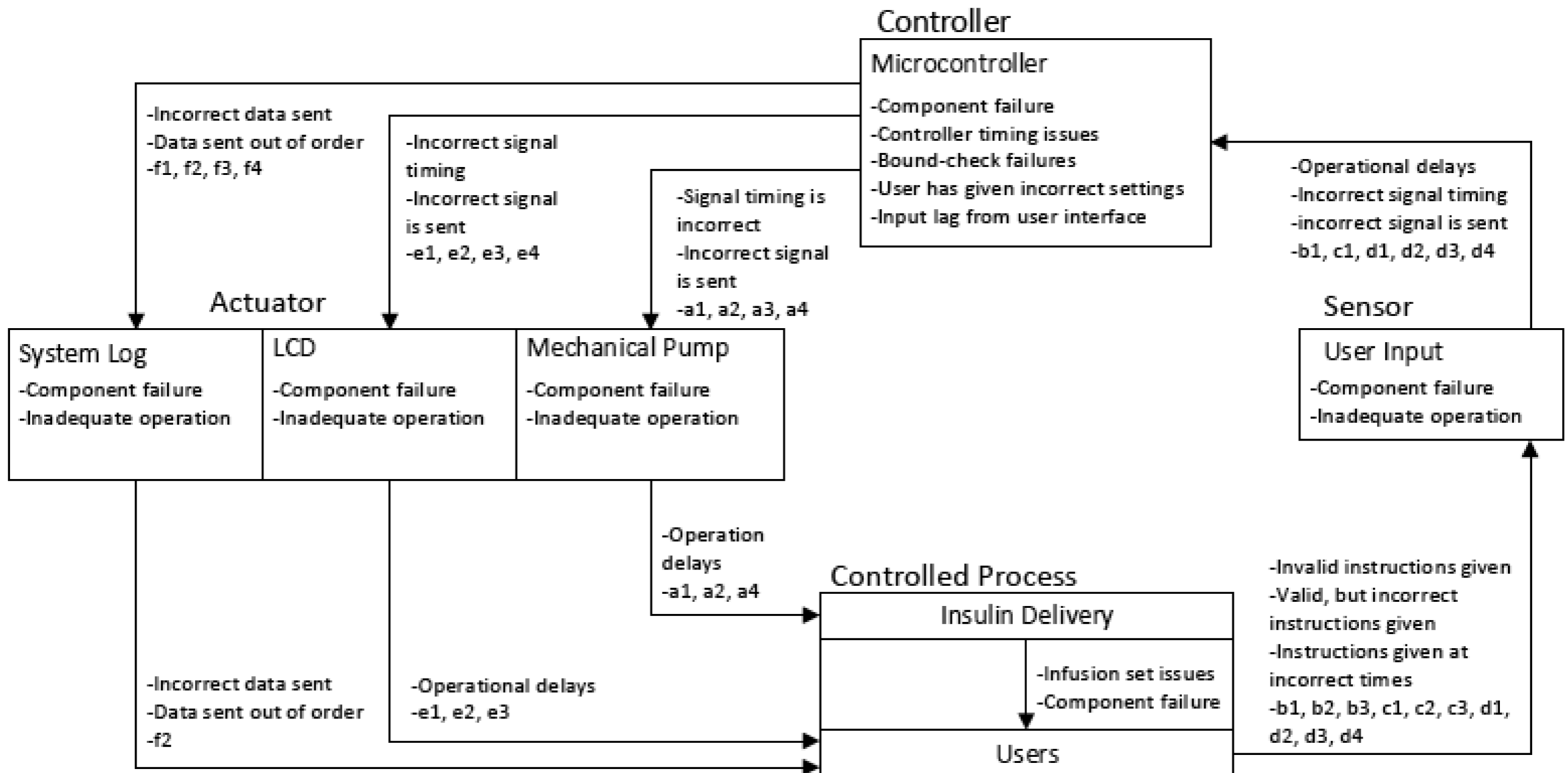
Example STPA: Insulin Pump

Viewed as a control system



Example STPA: Insulin Pump

Hazards included



Example STPA: Insulin Pump

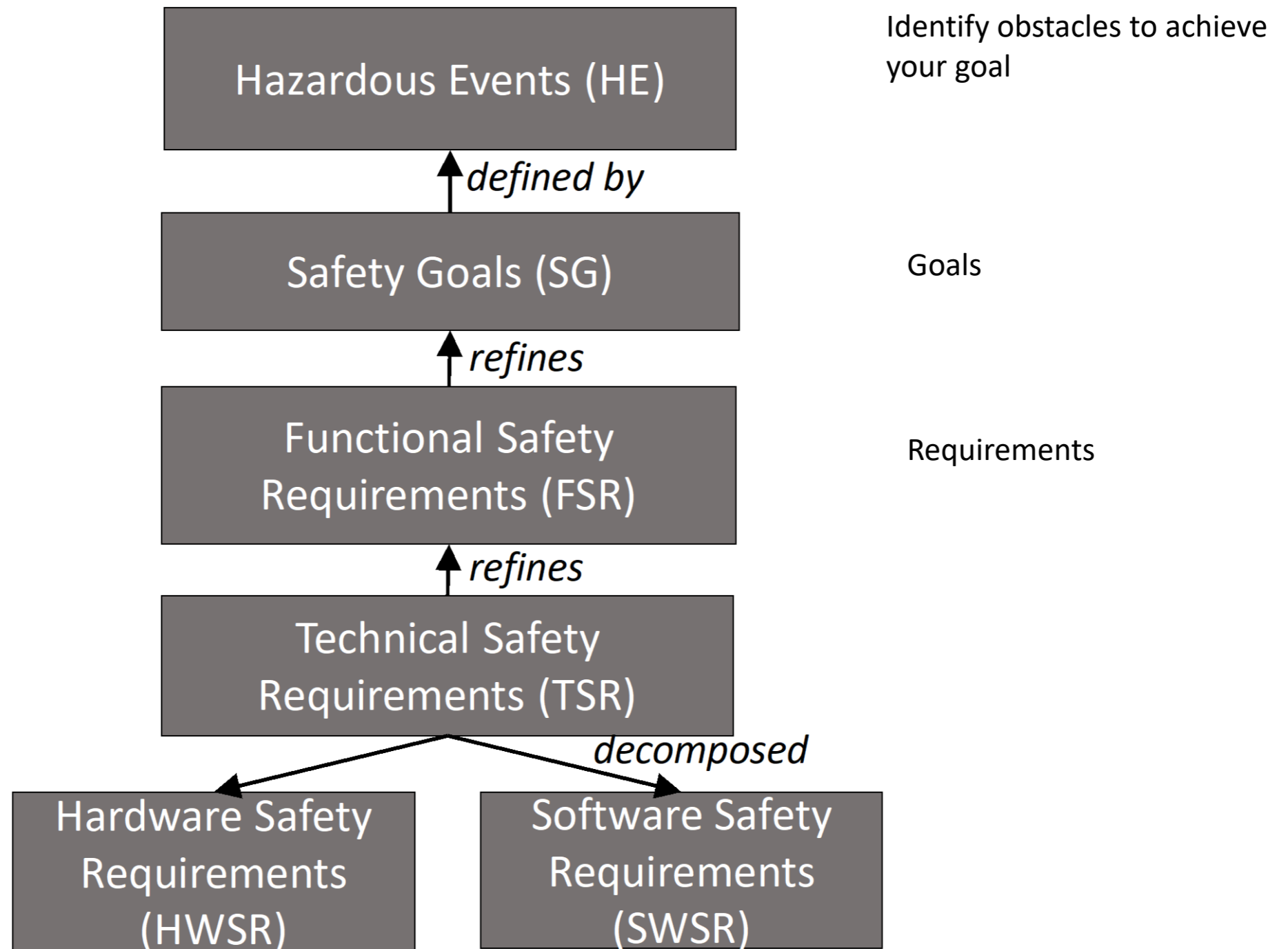
Control Action	Category 1: A control action required for safety is not provided or is not followed.	Category 2: An unsafe control action is provided that leads to a hazard.	Category 3: A potentially safe control action is provided too early, too late, or out of sequence).	Category 4: A safe control action is stopped too soon.
Deliver Insulin	System does not output insulin when requested, or does not stop providing insulin when it is required to.(a1)	System outputs too much insulin, or not enough insulin.(a2)	System stops insulin delivery too early, too late, or not at all. System starts insulin delivery too late, or not at all.(a3)	Insulin delivery is stopped before reaching the required delivery amount.(a4)
Create Basal and Bolus Profiles	Input bounds are incorrect or unsafe.(b1)	Incorrect Basal and Bolus information is entered.(b2)	Input bounds are entered at incorrect prompts (e.g. entering min bound when prompted for max).(b3)	N/A
Modify Basal and Bolus Bounds	Input bounds are incorrect or unsafe.(c1)	Incorrect Basal and Bolus information is entered.(c2)	Input bounds are entered at incorrect prompts (e.g. entering min bound when prompted for max).(c3)	N/A
Control Profile Activation	User does not set a profile for activation in an appropriate time frame.(d1)	User activates an incorrect profile.(d2)	User activates profile too early, or too late.(d3)	User ends a profile before it completes execution.(d4)
Provide Feedback	User feedback fails to inform the user of the systems status. Most hazardous in an alarm situation.(e1)	Feedback provides user with incorrect information.(e2)	User feedback, such as an alarm, happens too late. Status updates occur at incorrect times.(e3)	While user responds to an alarm, or status update, the system stops reporting an issue before it is fully resolved.(e4)
Logging information	System fails to write information to the log.(f1)	The information that is being logged or shown is incorrect.(f2)	Logging information recorded is too early, or too late. Logging information is done out of sequence, which could write incorrect information to the log.(f3)	Logging operation ends before finishing.(f4)

Safety Cases

**A clear,
comprehensive and defensible argument
that a system is acceptably safe to operate
in a particular context.**

(Tim Kelly / Rob Weaver University of York)

Recall: ISO 26262 Recommendation



Assurance Process

1. Completely and correctly identify goals (for safety / security / privacy)
2. Collect sufficient evidence that you have adequately dealt with each of them

Assurance Case



- A.k.a. safety case, security case, privacy case, etc.
- An artifact that shows how each of the important **claims** about the system can be argued for, ultimately from **evidence** obtained about the system
- **Evidence** can come in many forms:
 - test results
 - analyses
 - model checking results
 - expert opinion
 - etc.
- The **argument** is often informal
 - “sufficient”
 - “adequate”...with some degree of **confidence**

Safety (Assurance) Case Types: Textual

Within the **context** of the tolerability targets for hazards (from reference Z) and the list of hazards identified from the functional hazard analysis (from reference Y), we follow the **strategy** of arguing over all three of the identified hazards (H1, H2, and H3) to establish sub-claim 1, yielding three additional **claims**: H1 has been eliminated; H2 has been sufficiently mitigated; and H3 has been sufficiently mitigated.

The **evidence** that H1 has been eliminated is formal verification.

The **evidence** that catastrophic hazard H2 has been sufficiently mitigated is a fault tree analysis showing that its probability of occurrence is less than 1×10^{-6} per annum. The **justification** for using this evidence is that the acceptable probability in our environment for a catastrophic hazard is 1×10^{-6} per annum.

The **evidence** that the major hazard H3 has been sufficiently mitigated is a fault tree analysis showing that its probability of occurrence is less than 1×10^{-3} per annum. The **justification** for using this evidence is that the acceptable probability in our environment for a major hazard is 1×10^{-3} per annum.

We establish sub-claim (2) within the **context** of the list of hazards identified from the functional hazard analysis in reference Y, and the integrity level (IL) process guidelines defined in reference X. The process **evidence** shows that the primary protection system was developed to the required IL 4. The process **evidence** also shows that the secondary protection system was developed to the required IL 2.

Claim 1: Control system is acceptably safe.

Context 1: Definition of acceptably safe.

Claim 1.1: All identified hazards have been eliminated or sufficiently mitigated.

Context 1.1-a: Tolerability targets for hazards (reference Z).

Context 1.1-b: Hazards identified from functional hazard analysis (reference Y).

Strategy 1.1: Argument over all identified hazards (H1, H2, H3)

Claim 1.1.1: H1 has been eliminated.

Evidence 1.1.1: Formal verification

Claim 1.1.2: Probability of H2 occurring $< 1 \times 10^{-6}$ per annum.

Justification 1.1.2: 1×10^{-6} per annum limit for catastrophic hazards.

Evidence 1.1.2.: Fault Tree analysis.

Claim 1.1.3: Probability of H3 occurring $< 1 \times 10^{-3}$ per annum.

Justification 1.1.3: 1×10^{-3} per annum limit for major hazards.

Evidence 1.1.3: Fault tree analysis.

Claim 1.2: The software has been developed to the integrity level appropriate to the hazards involved.

Context 1.2-a: (same as Context 1.1-b)

Context 1.2-b: Integrity level (IL) process guidelines defined by reference X.

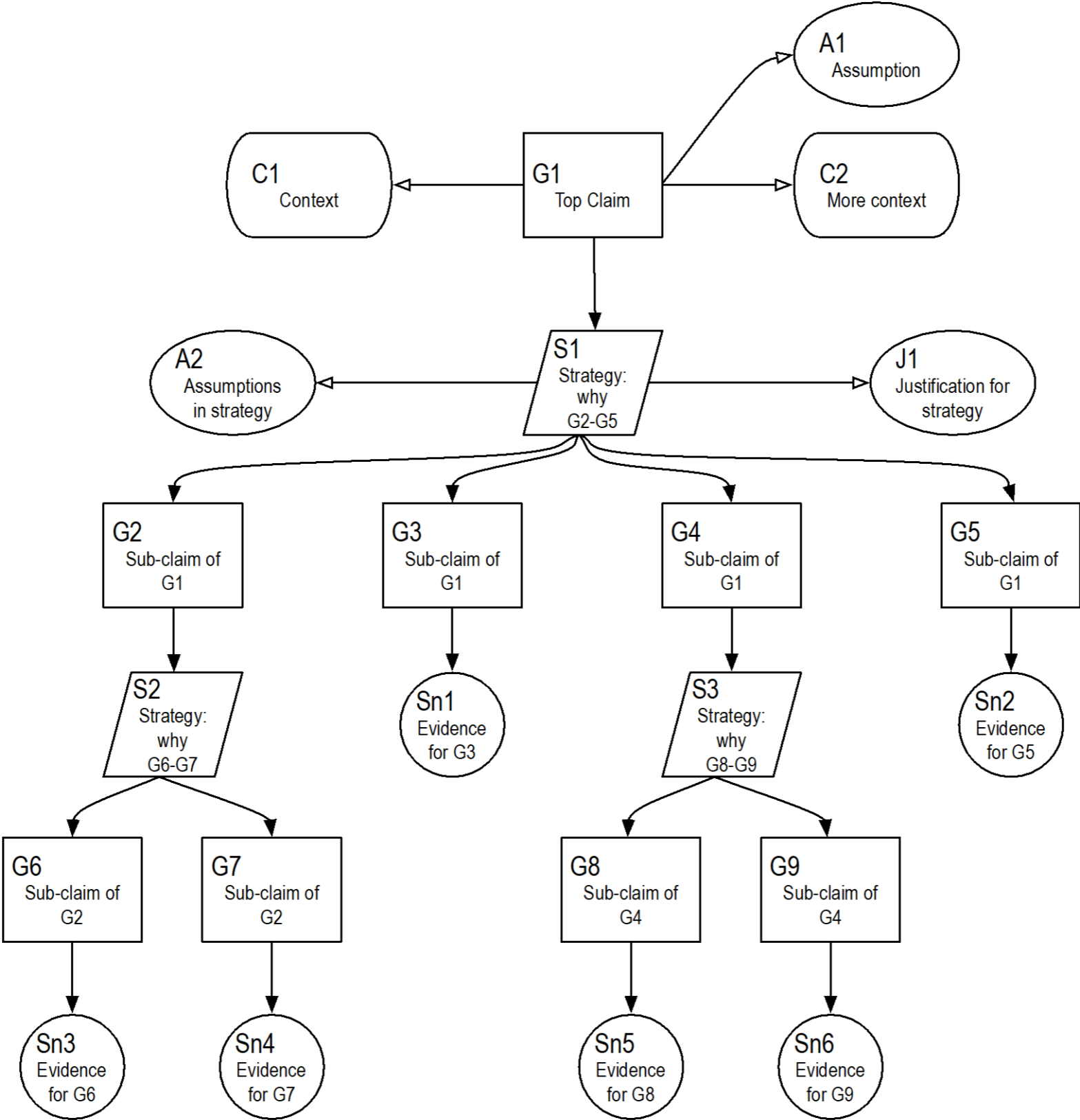
Claim 1.2.1: Primary protection system developed to IL 4.

Evidence 1.2.1: Process evidence of IL 4

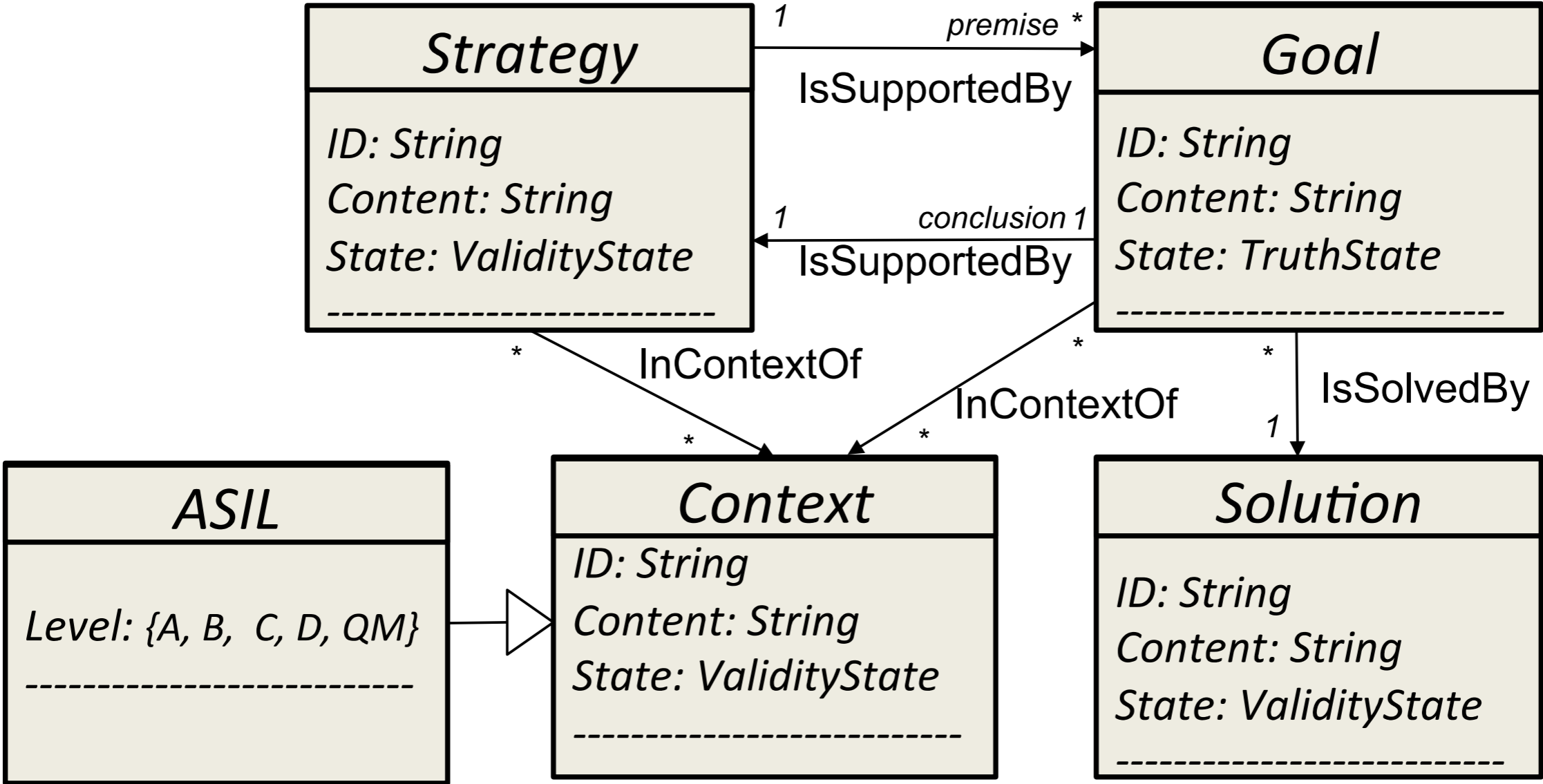
Claim 1.2.2: Secondary protection system developed to IL 2.

Evidence 1.2.2: Process evidence of IL 2.

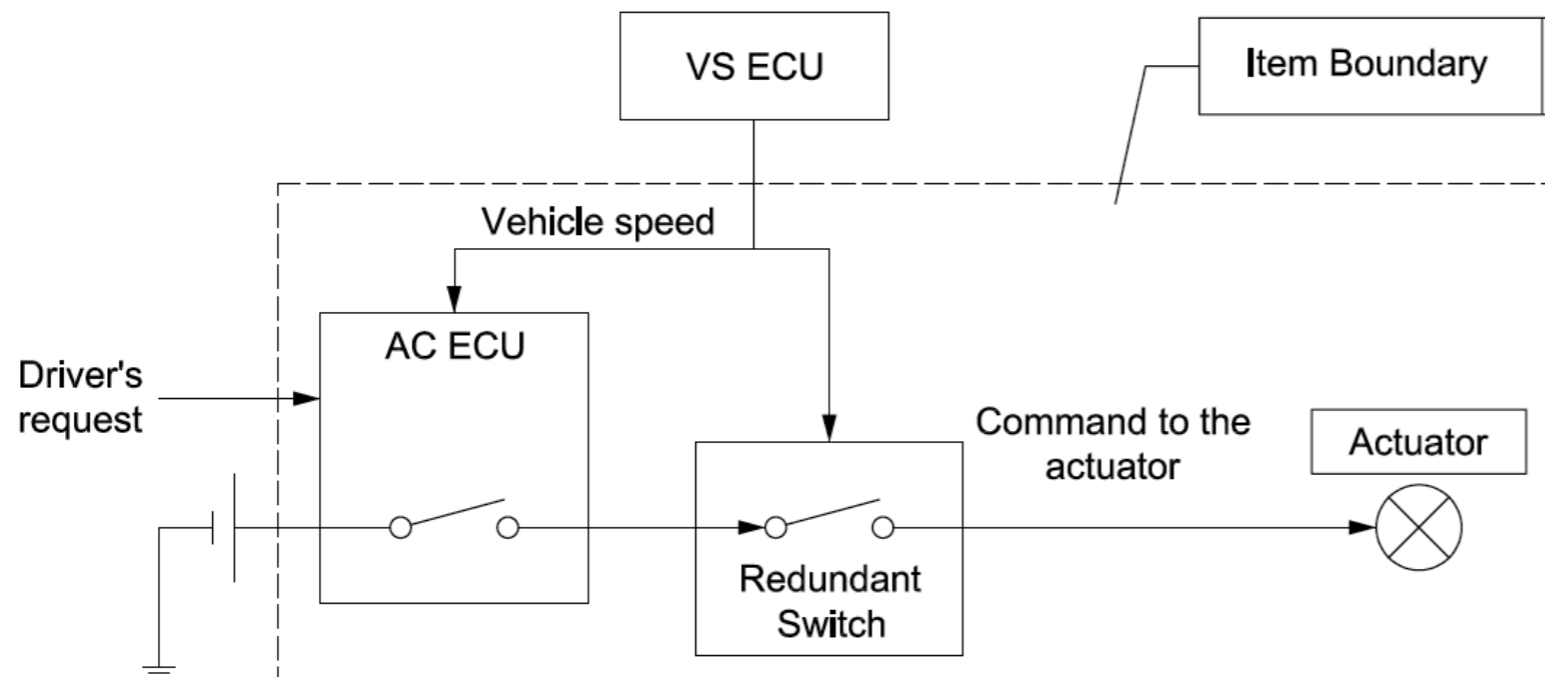
Safety (Assurance) Case Types: Graphical



Assurance Case Modeling with GSN – Goal Structuring Notation

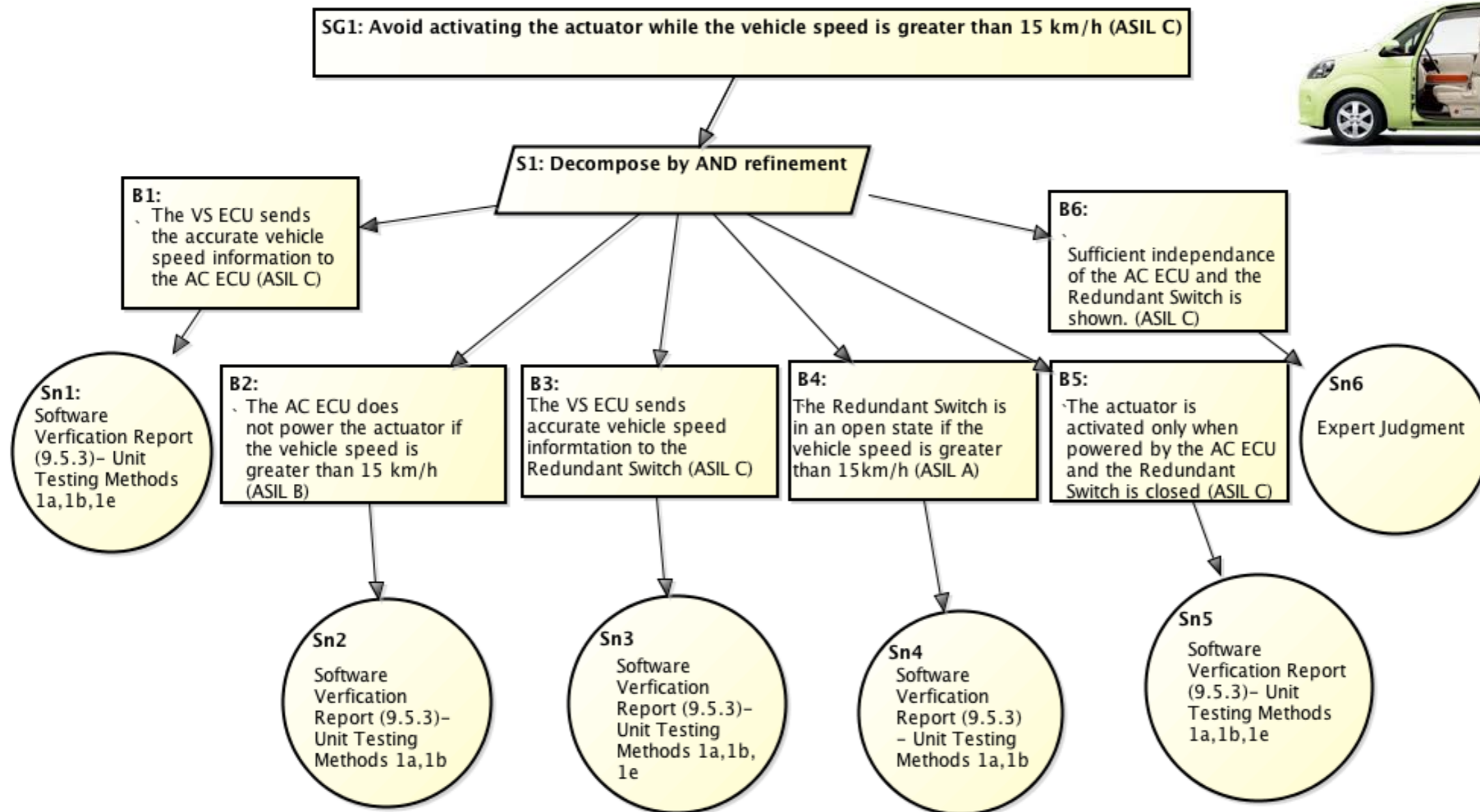


Example: Power Sliding Door System



Safety goal SG1
Avoid activating the actuator when
vehicle speed > 15 kph

Power Sliding Door Safety Case



Incremental development: The answer to the Complexity/Time Crunch?

- Idea: Reuse existing safety assurance arguments for minor design changes (e.g. towing features)
- Sounds promising! What are the results?



A jury has awarded \$3 million in the case of a fatal crash involving this Toyota Camry.

Toyota Unintended Acceleration (UA)

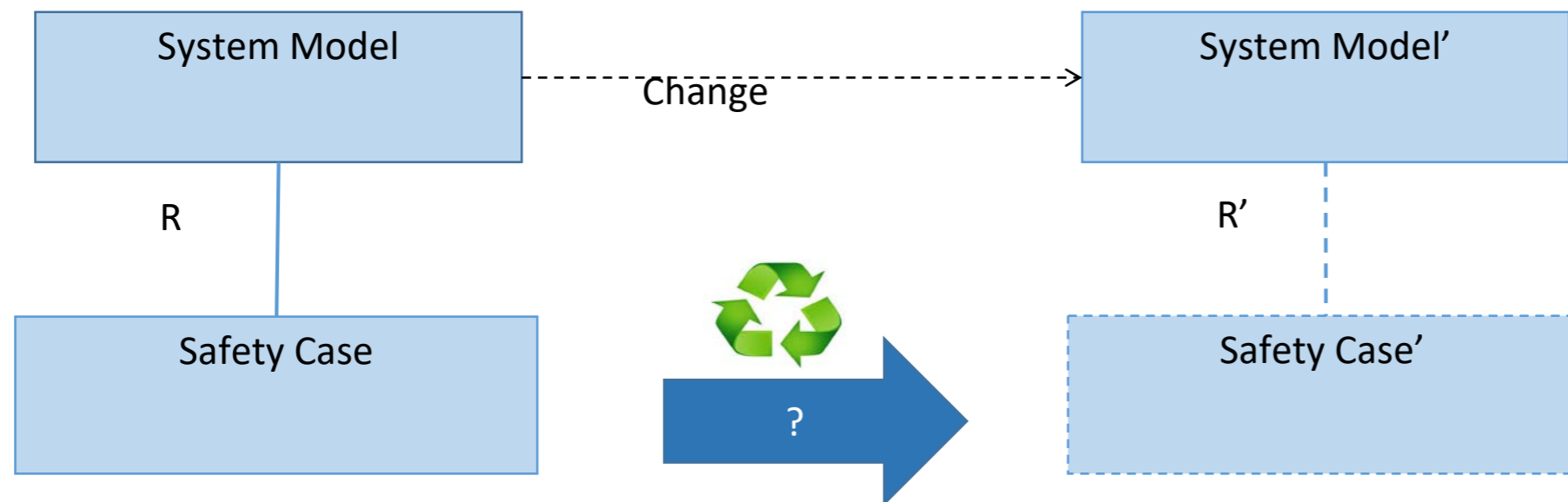
- Brake Echo Check failsafe system for UA only received “brake transitions”
- If your foot is already on the brake ...and then a UA event occurs, you may have to **completely take foot off the brake & reapply** to trigger failsafe system!

2015 Ford Fusion vehicles equipped with a mechanical key and dual screen cluster, 30 minutes after the ignition is turned off, the Body Control Module (BCM) **allows** the **key to be removed** when the transmission is **not in Park**. Part 573 Safety Recall Report 14V-736



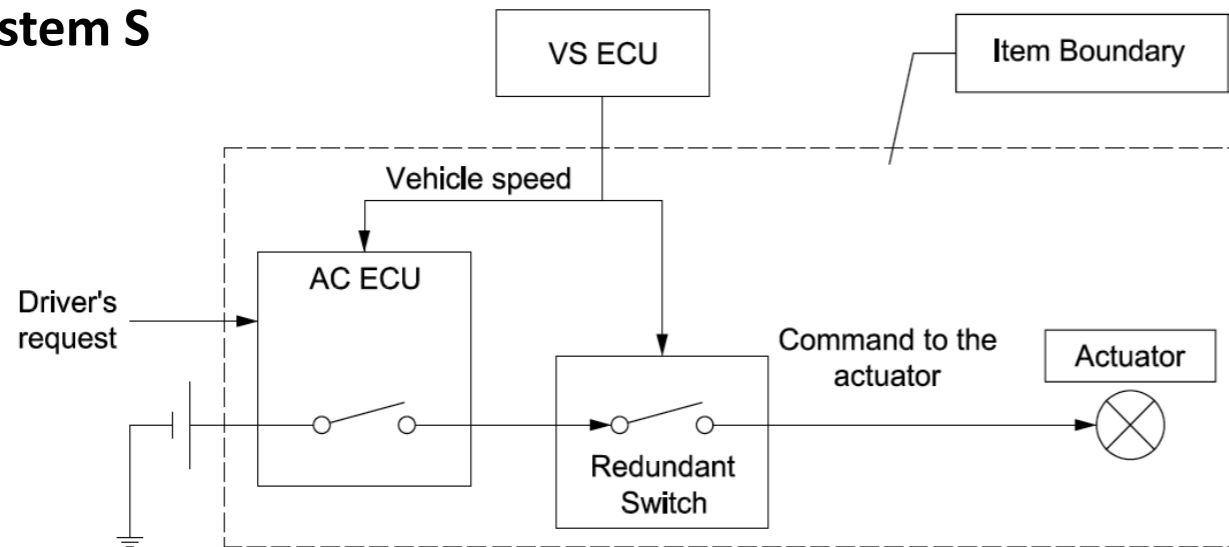
Problem: Support for System and Safety Evolution

- ... correctly
- ... quickly (via scalable automated tool support)
- ... while facilitating product-level and product-line reuse



System change: Removing Redundant Switch in Power Sliding Door (PSD)

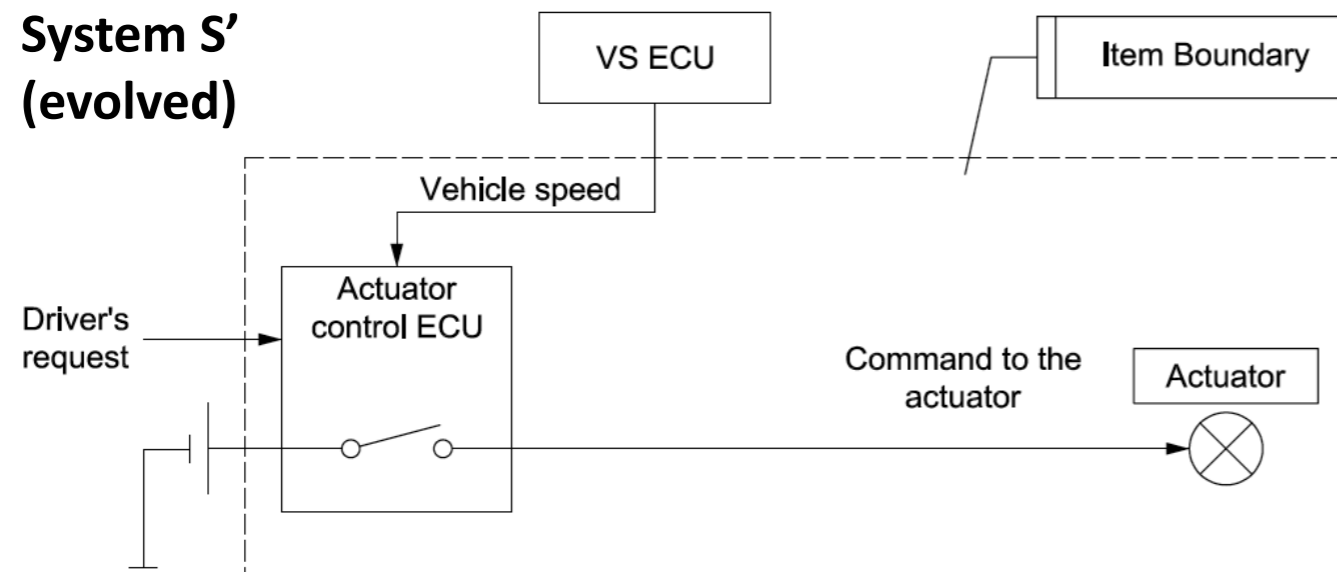
System S



Safety Goal SG1:
Avoid activating the actuator
when vehicle speed > 15 kph

**Δ: removal of
redundant switch**

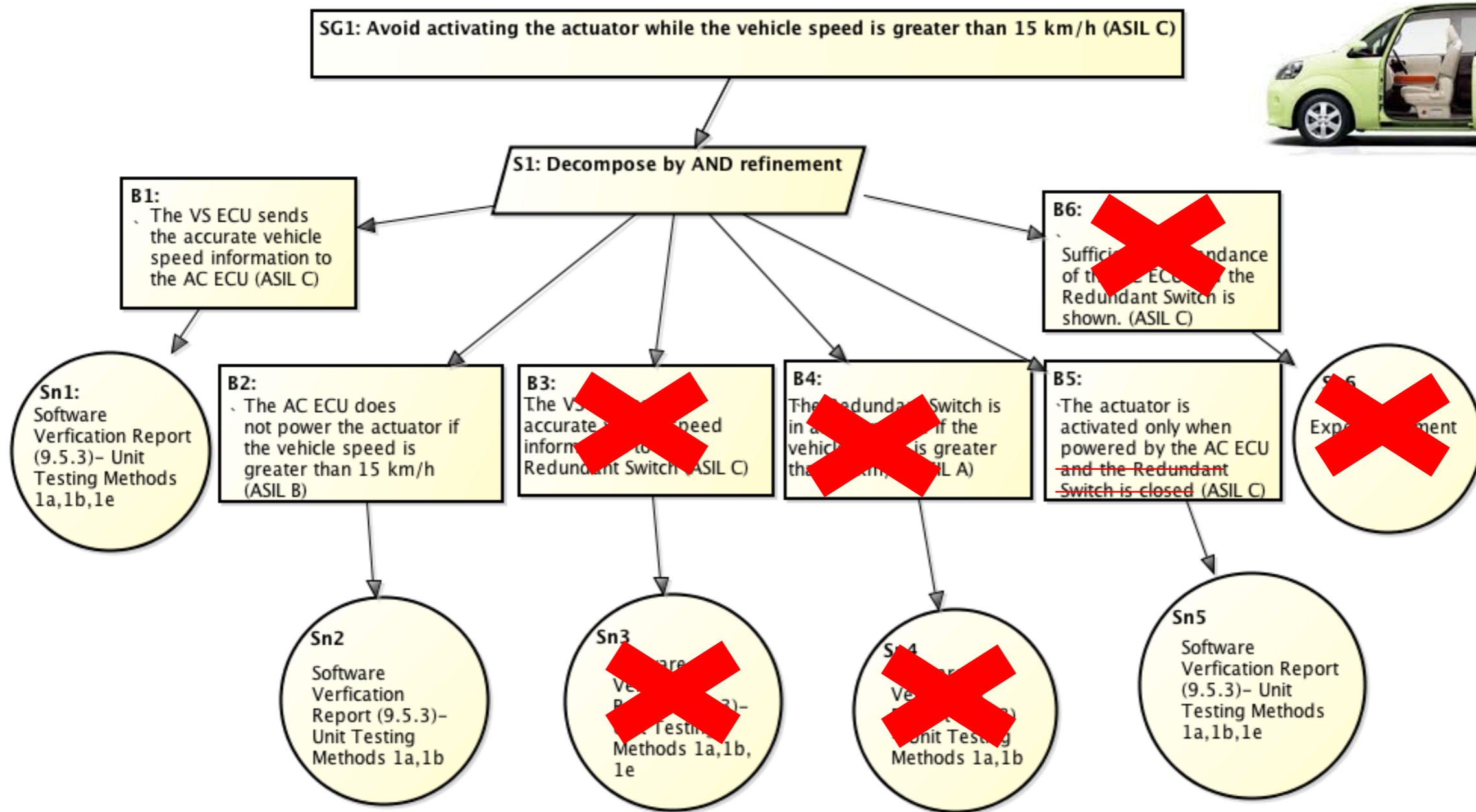
System S' (evolved)



Safety goal remains
the same.

How you achieve it &
*why you believe it
changes.*

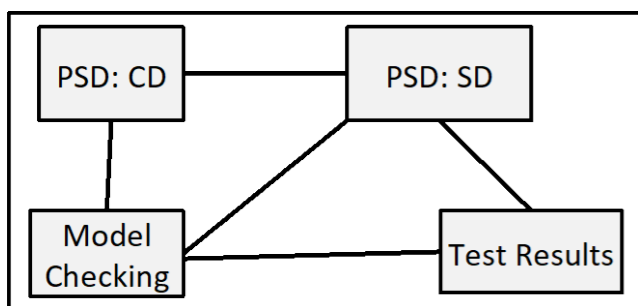
Naïve Evolution of Safety Case



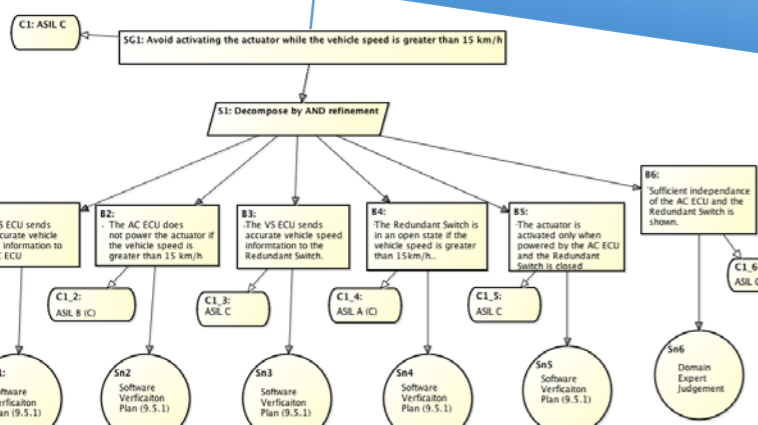
Naïve evolution approach: **Delete** everything related to switch

Solution: Model Based Impact Assessment

System Megamodel



Traceability



Assurance Case

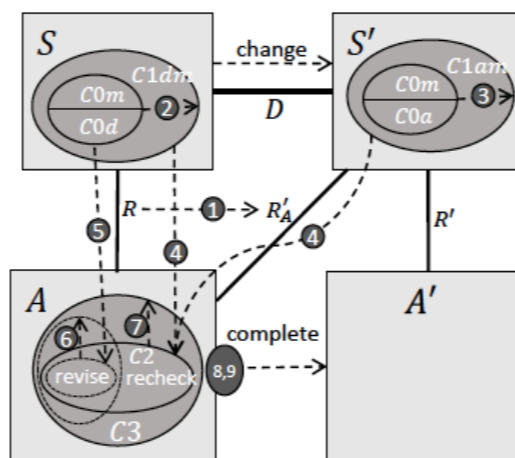
Delta (change)



Model Slicers



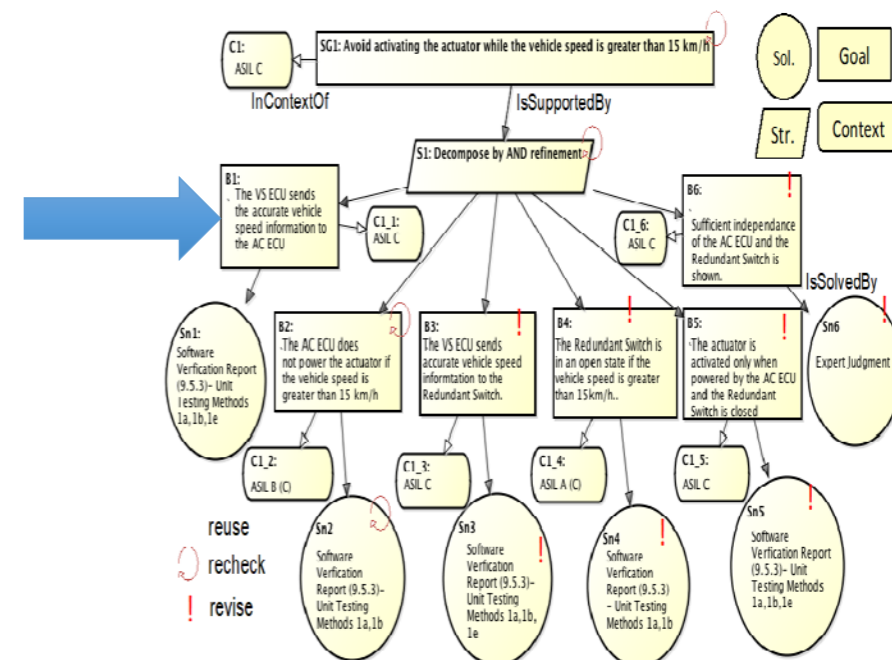
Model-Based Impact Assessment Algorithm



Human-in-the-loop refinement



Annotated Assurance Case

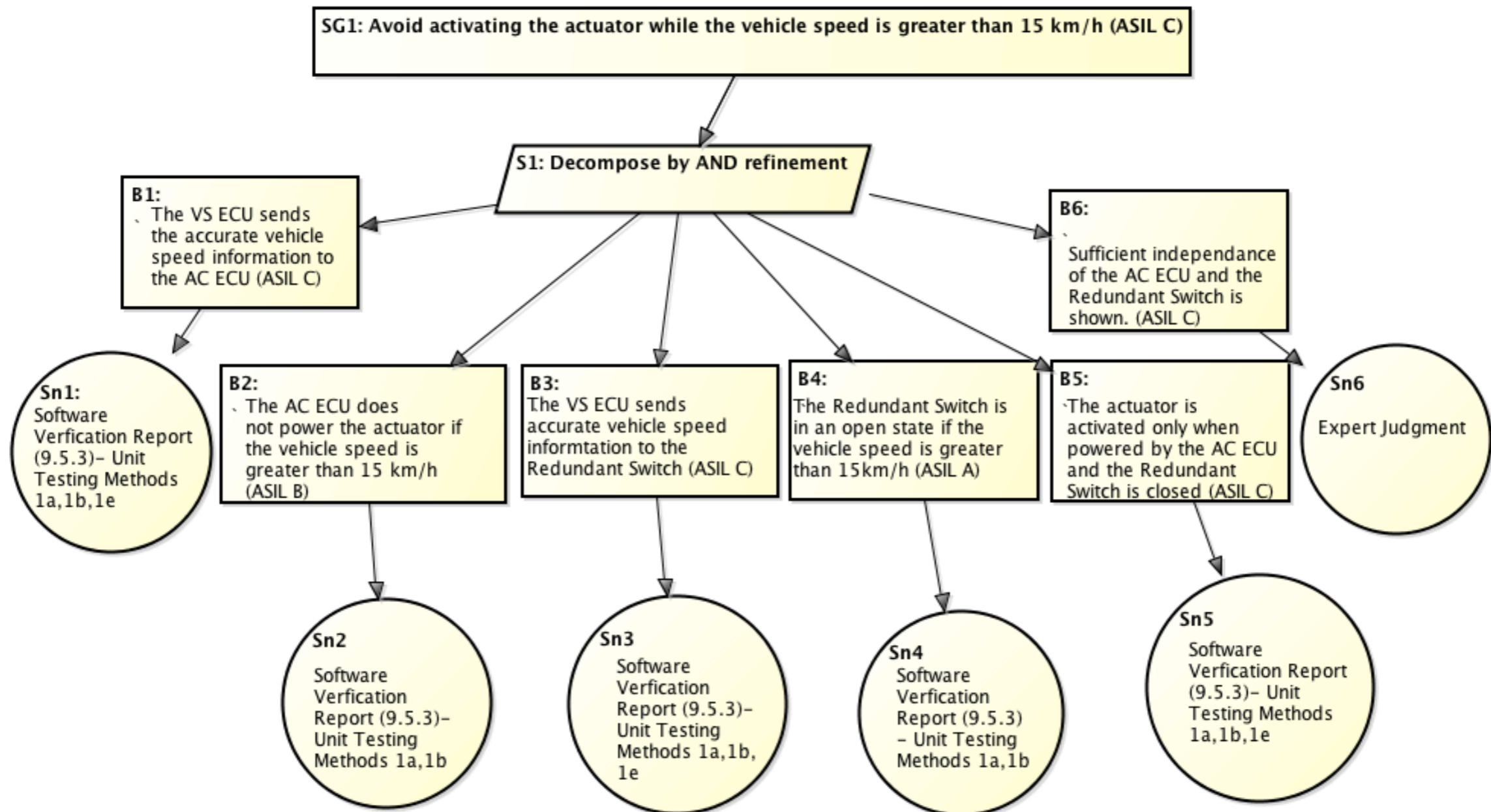


✓ reuse ↻ recheck ! revise

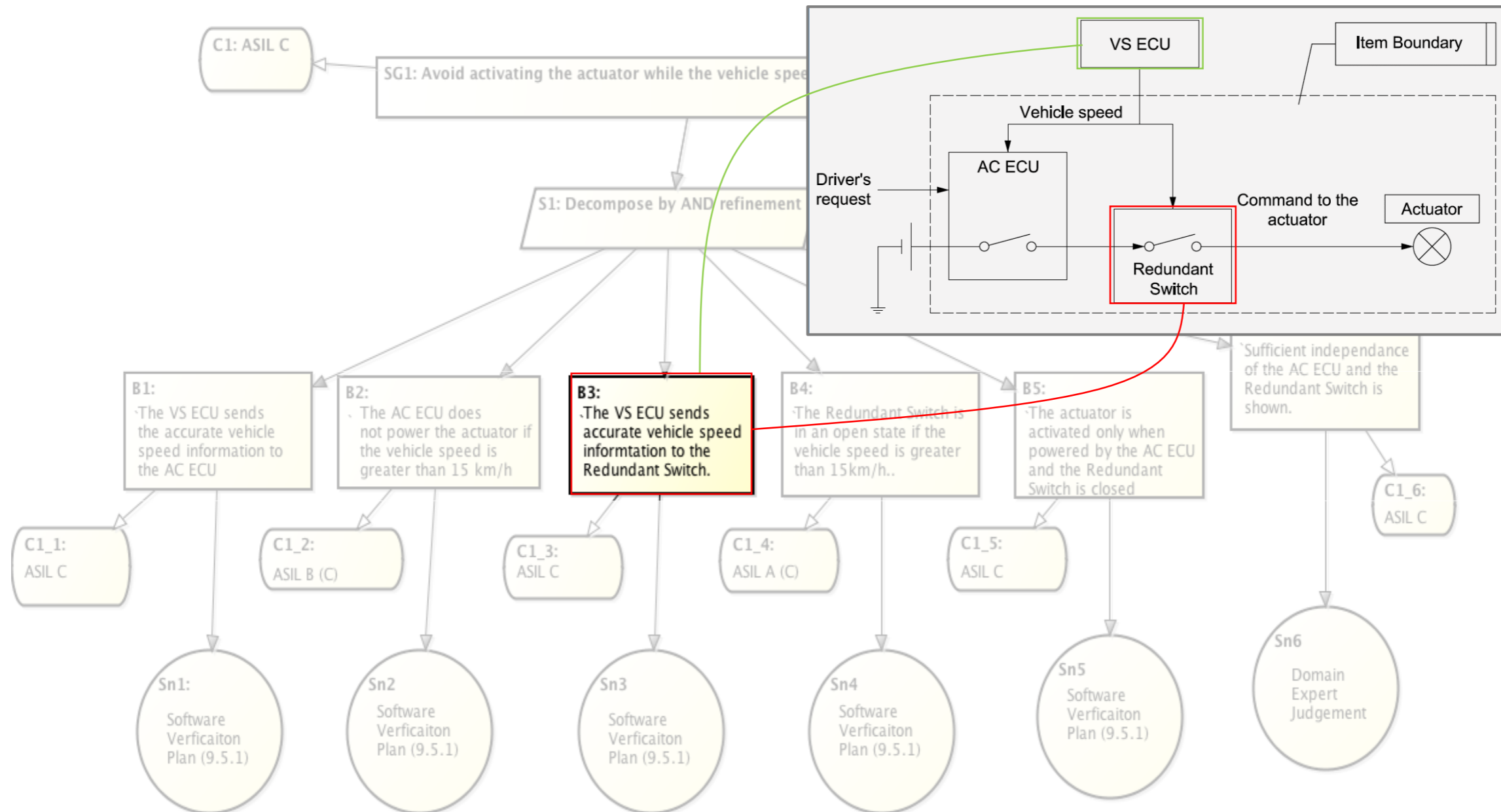
[MODELS16]: original approach

[SafeComp17]: improved approach, assurance case slicer, cost-savings analysis

Begin with original safety case

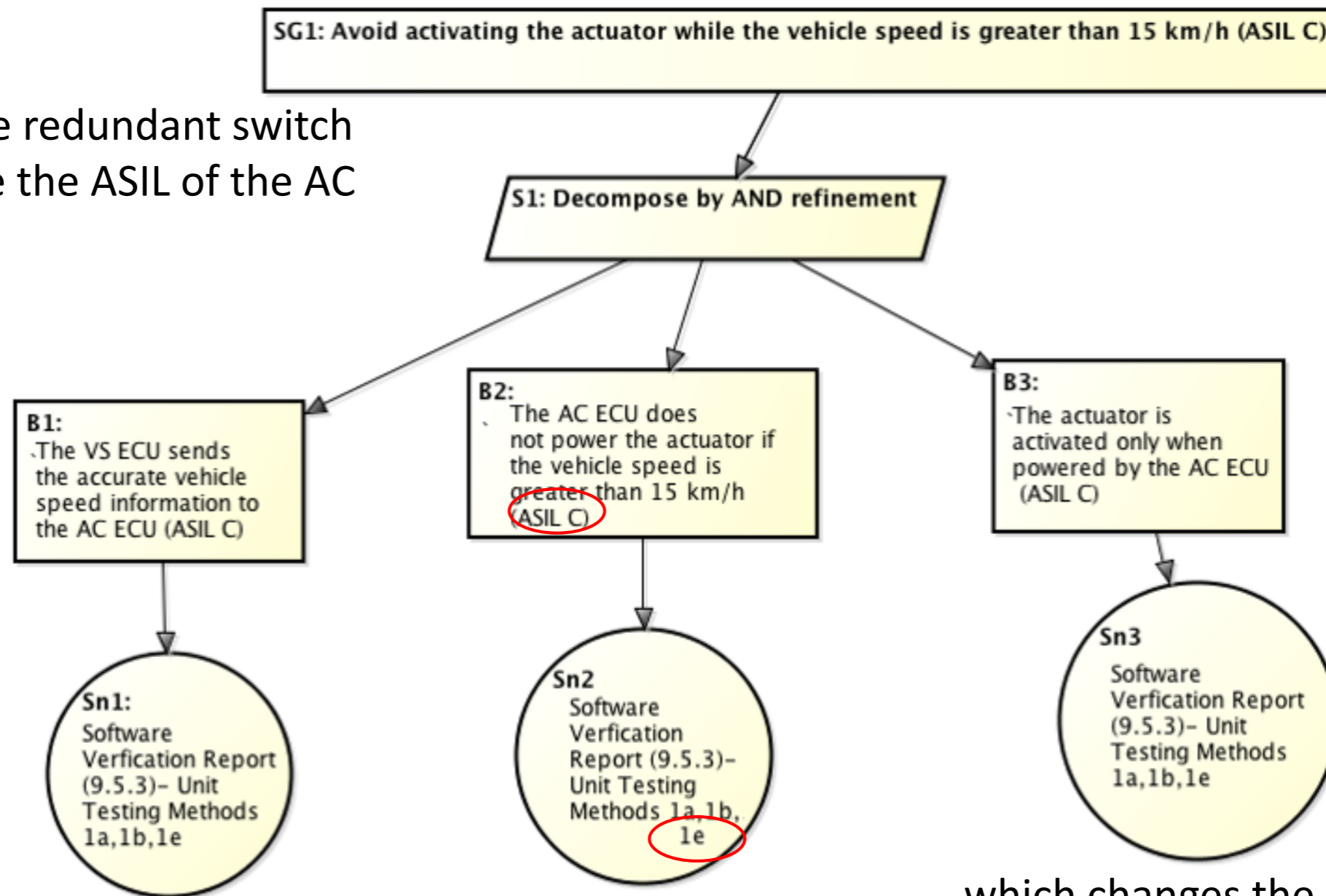


Based on traceability between system and safety case...



Safety informed evolution of safety case (after review and refinement by engineers)

By removing the redundant switch
we have change the ASIL of the AC
ECU to C

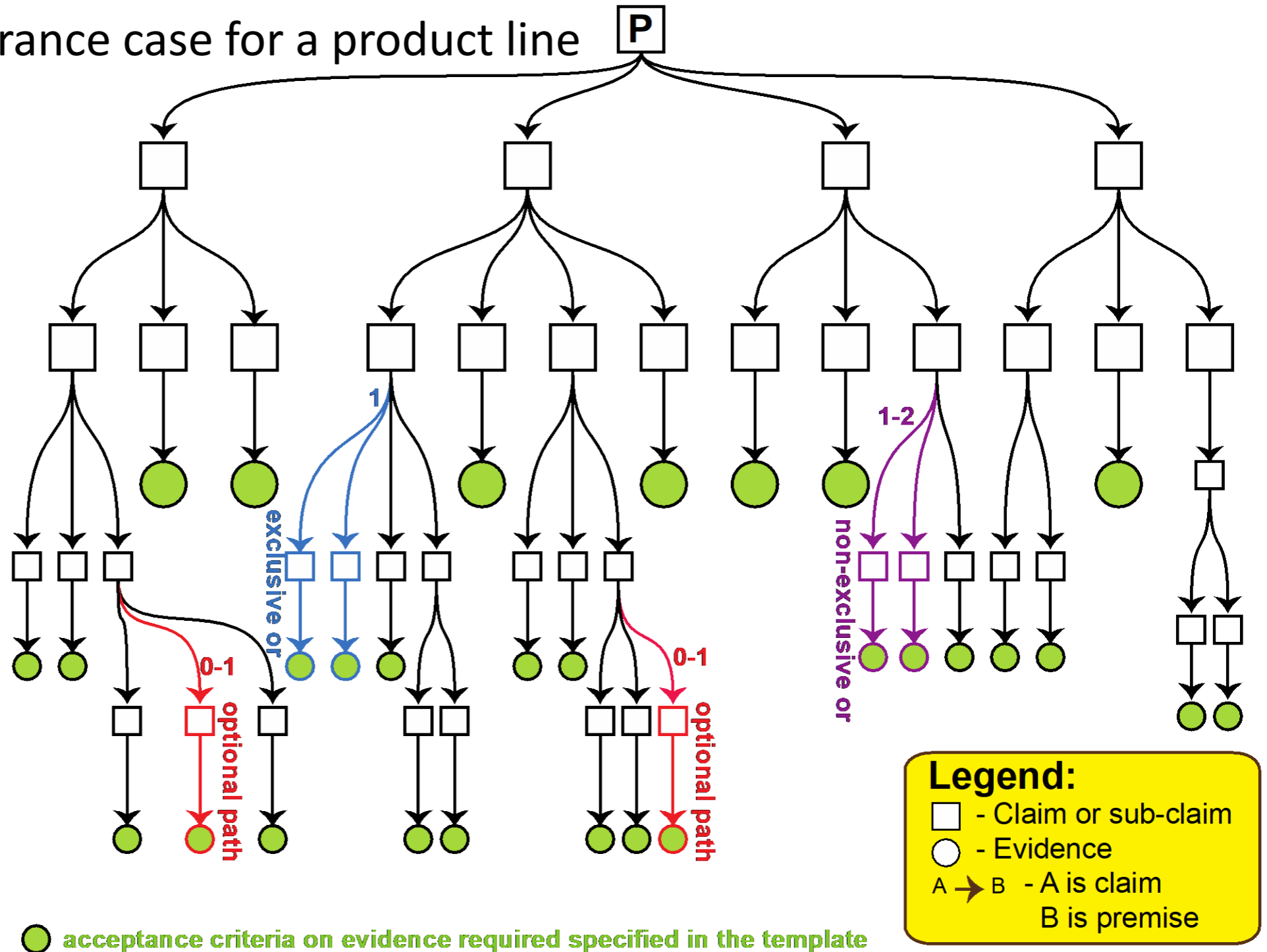


... which changes the acceptance
criteria for the evidence

Assurance Case Templates

- Complete assurance case for a product line **P**

- Complete assurance case produced before development starts
- Optional argument paths
- Evidence nodes specify type of evidence required and acceptance criteria on that evidence
- Requires explicit reasoning (not shown here)
- Try to make it robust with respect to change
- Assume it will be developed by a community in the same way that standards are
- Could replace traditional standards



ISO 26262 Assurance Case Template for ADAS

G
 <ADAS> considered as an ISO 26262 item, delivers the behaviour required, and does not adversely affect the safety of the vehicle, over its expected lifetime, in its intended environment.

SR

Strategy:

G can be decomposed into

1. Functional safety concept verified. (GS)
2. <ADAS> complies with Functional safety requirements and is released for production (26262). (GR)
3. Production and maintenance processes. (GPM)
4. Configuration management. (GC)
5. Change management. (GCM)
6. <ADAS> not expected to violate documented assumptions (GA)

Reasoning:

Premise: GS, GR, GPM, GC, GCM and GA are true. Claim: G is true

i) These 6 premises cover all the major premises in 26262 (See SRi)

ii) GR as in 26262 has been supplemented by our knowledge of SE. Specifically, general functional requirements must not adversely interact with the safety requirements. (See SRii)

iii) Important component is operational assumptions are not so onerous that drivers are likely not to comply with them, and those related to the environment will also be valid. (See SRiii)

GS

The safety concept of <ADAS> is Verified. This includes that all necessary functional safety requirements are derived from a vehicle level hazard and risk analysis and validated

GR

Implementation of <ADAS> complies with requirements within tolerance. Requirements are unambiguous, complete on input domain and internally consistent. They include all necessary <ADAS> Functional Safety Requirements, derived from HARA

GPM

Safety of the vehicle is maintained throughout its operating life, through compliance with Production Requirements, Service Maintenance Requirements, & Decommissioning Requirements in ISO 26262.

GC

Configuration Management complies with ISO 26262.

GCM

Change Management complies with ISO 26262.

GA

<ADAS> operational assumptions are documented, and operation of the vehicle in which <ADAS> is installed is not expected to violate these assumptions.

Even your keychain might be part of the problem!

- A 1.6 mm difference in a \$0.57 part!

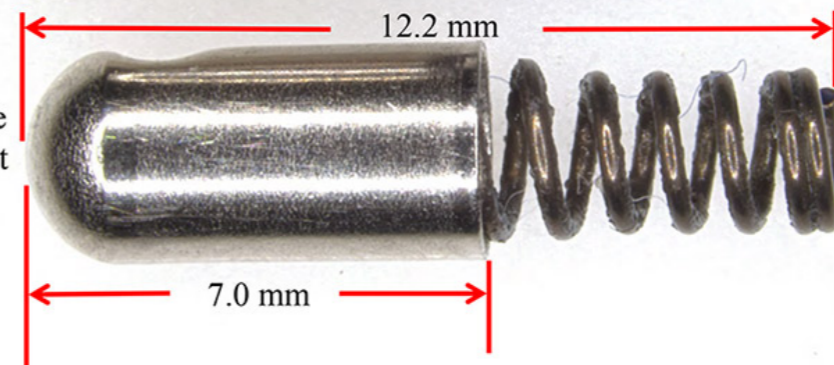


Air Bag Control Module ECU

2005
Model Year



New Service
Replacement
Part

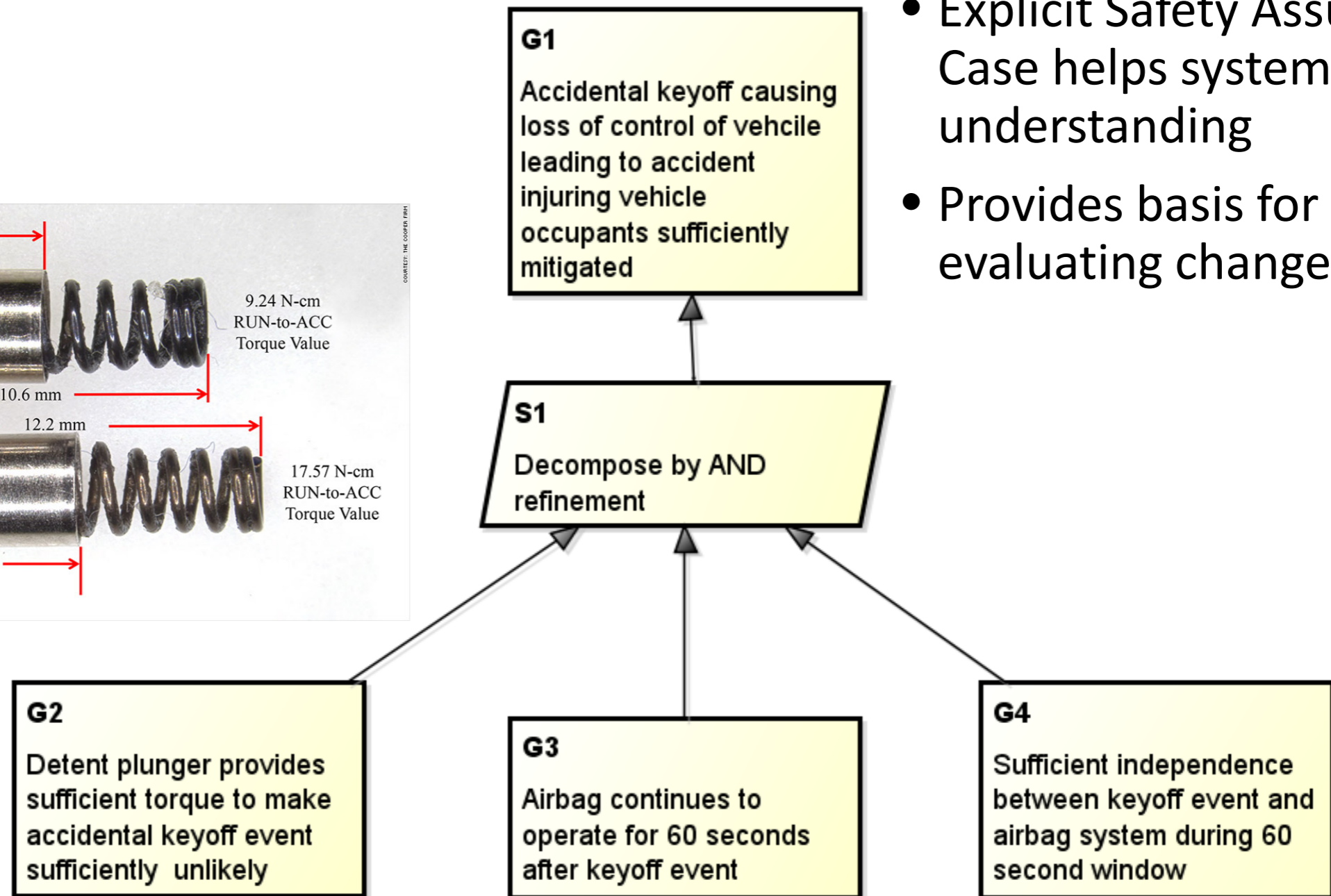
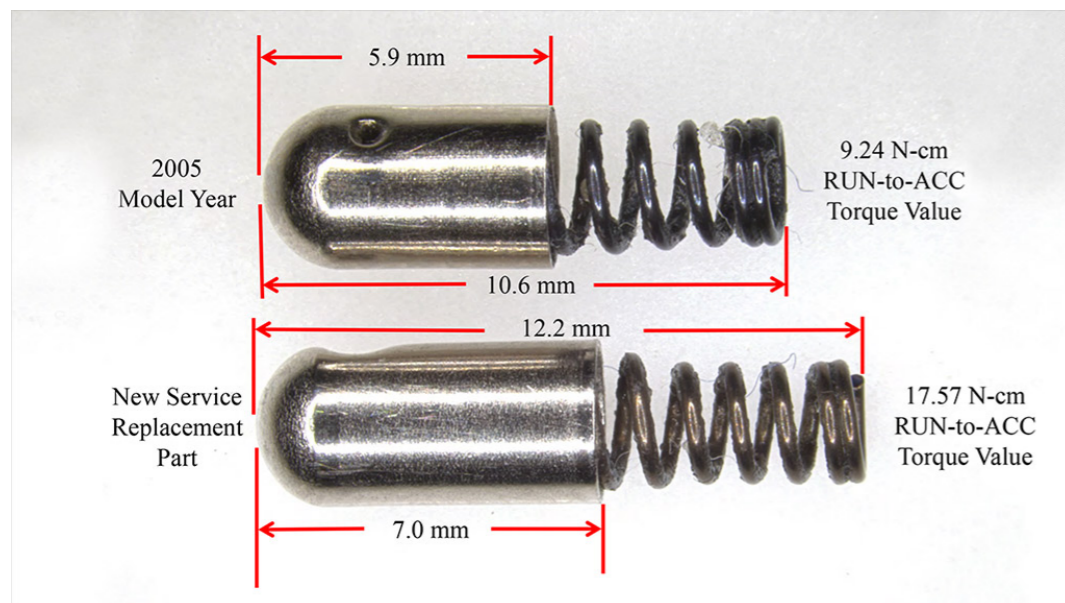


12.2 mm

7.0 mm

17.57 N-cm
RUN-to-ACC
Torque Value

Ignition Switch (keychain) Recall Revisited



- Explicit Safety Assurance Case helps system understanding
- Provides basis for evaluating changes

Toyota Acceleration – from 2014 talk by Philip Koopman

Overview

- Brief history of Toyota UA events
 - Recalls, investigations, lawsuits
 - Fines & jury awards – **\$\$Billions**
- Technical discussion of the problems
 - **This is a Case Study** – what can we learn?
- What does this mean for future automobiles?
 - The bar is raised, at least for now
 - E.g, handling of GM ignition switch & Honda hybrid SW UA

Aug. 28, 2009, San Diego CA, USA

- Toyota Lexus ES 350 sedan
 - UA Reached 100 mph+
- 911 Emergency Phone Call from passenger during event
 - All 4 occupants killed in crash

- Driver:

Mark Saylor, 45 year old male.

Off-duty California Highway Patrol Officer; vehicle inspector.

- Crash was blamed on wrong floor mats causing pedal entrapment
- Brake rotor damage indicated “endured braking”
- This event triggered escalation of investigations dating back to 2002 MY

The New York Times

February 1, 2010



The wreckage of a Lexus ES 350 in which four people died in August after it accelerated out of control.

Gomez Law Firm

http://www.nytimes.com/2010/02/01/business/01toyota.html?pagewanted=all&_r=0

<http://www.autoblog.com/2009/10/26/nhtsa-releases-new-info-about-crash-that-prompted-toyota-floor-mat/>

Recall and Investigation

(Brakes might not mitigate open throttle – more later)

- **Floor mat recalls**
 - Sept. 2007 recall to fasten floor mats
 - Wider recall Oct./Nov. 2009 after Saylor mishap
- **Sticky gas pedal recall**
 - Jan. 2010 and onward
- **Congressional investigation**
 - Toyota President testifies to US Congress, Feb. 2010
 - April 2010: Economic loss class action venue selected

<http://www.cnn.com/2010/POLITICS/02/24/toyota.hearing.updates/>

http://money.cnn.com/autos/storysupplement/toyota_timeline/

<http://articles.latimes.com/2010/feb/18/business/la-fi-toyota-exponent18-2010feb18>

May 25,
2010

Toyota "Unintended Acceleration" Has Killed 89



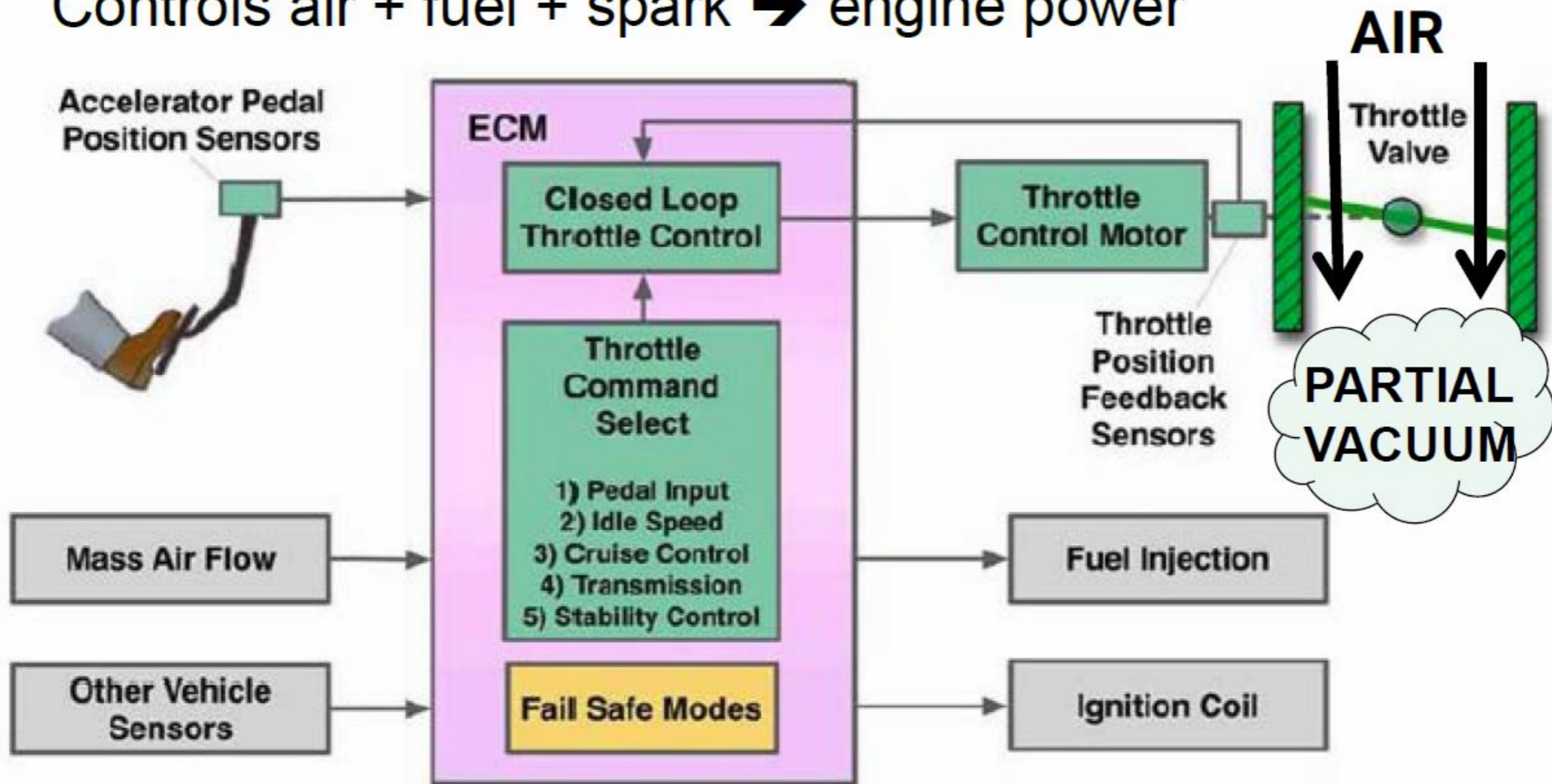
A 2005 Toyota Prius, which was in an accident, is seen at a police station in Harrison, New York, Wednesday, March 10, 2010. The driver of the Toyota Prius told police that the car accelerated on its own, then lurched down a driveway, across a road and into a stone wall. (AP Photo/Seth Wenig) / **AP PHOTO/SETH WENIG**

Unintended acceleration in Toyota vehicles may have been involved in the deaths of 89 people over the past decade, upgrading the number of deaths possibly linked to the massive recalls, the government said Tuesday.

The National Highway Traffic Safety Administration said that from 2000 to mid-May, it had received more than 6,200 complaints involving sudden acceleration in Toyota vehicles. The reports include 89 deaths and 57 injuries over the same period. Previously, 52 deaths had been suspected of being connected to the problem. <http://www.cbsnews.com/news/toyota-unintended-acceleration-has-killed-89/>

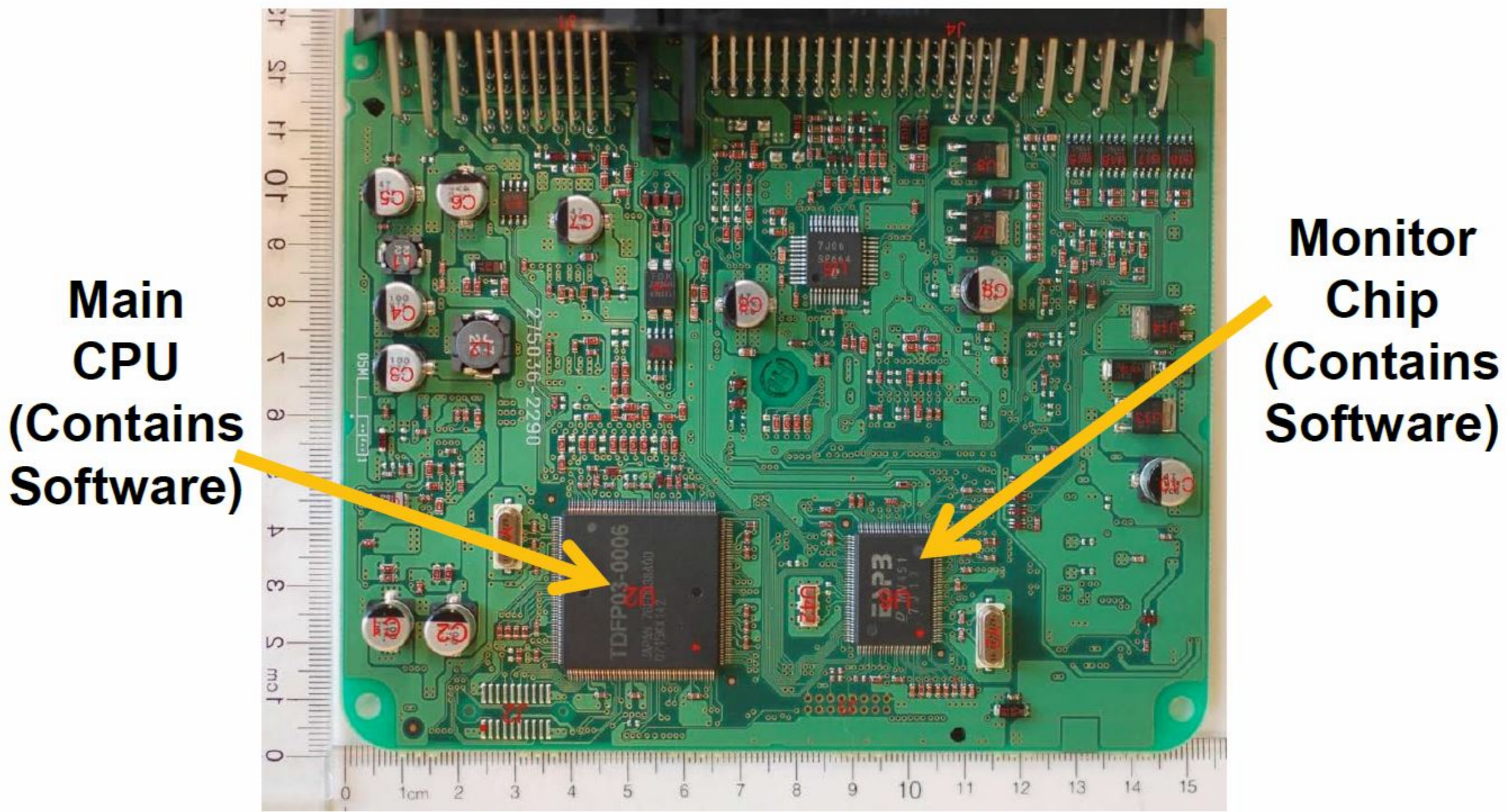
NASA investigation

- NASA team investigates UA (2010-2011)
 - Including **Electronic Throttle Control System (ETCS)**
 - Controls air + fuel + spark → engine power



[NASA UA Report Fig 4.0-1]

Toyota 2008 Electronic Traction Control System (ETCS) – Two CPUs



http://m.eet.com/media/1201063/Toyota_ECM.jpg

ETCS is Safety-Critical

- If **driver pumps brakes**, loses vacuum power-assist
 - With depleted vacuum, holding against WOT requires average of **→ → 175 pounds of force on brake pedal** across vehicles tested [NHTSA data]
 - With vacuum it's only 15.0 - 43.6 pounds force
[http://www.nhtsa.gov/staticfiles/nvs/pdf/NHTSA-Toyota_vehicle_characterization.pdf]
- A software defect could command UA, for example via Wide Open Throttle (WOT)
 - **The brakes will not necessarily stop the car**
[Consumer reports: <http://www.youtube.com/watch?v=VZZNR9O3xZM>]
 - **Potential to command WOT matters for safety**
 - Not just whether there is an actual bug in that does that
 - Drivers will not necessarily perform countermeasures ([NASA UA Report, p. 66]: **shift to neutral**; key-off while moving)

NASA Conclusions

- NASA didn't find a "smoking gun"
 - Tight timeline & limited information [Bookout 2013-10-14AM 39:18-40:8]
 - **Did *not* exonerate system**

Proof for the hypothesis that the ETCS-i caused the large throttle opening UAs as described in submitted VOQs could not be found with the hardware and software testing performed.

Because proof that the ETCS-i caused the reported UAs was not found does not mean it could not occur. However, the testing and analysis described in this report did not find that TMC ETCS-i electronics are a likely cause of large throttle openings as described in the VOQs.

[NASA UA Report. Executive Summary]

- But, U.S. Transportation Secretary Ray LaHood said, "We enlisted the best and brightest engineers to study Toyota's electronics systems, and the verdict is in. There is no electronic-based cause for unintended high-speed acceleration in Toyotas."

<http://www.nhtsa.gov/PR/DOT-16-11>

\$1.6B Economic Loss Class Action

- “Lawsuit pursues claims for breach of warranties, unjust enrichment, and violations of various state consumer protection statutes, among other claims.”
 - <https://www.toyotaelsettlement.com/>
 - 2002 through 2010 models of Toyota vehicles
 - Toyota denies claims; settled for \$1.6 Billion in Dec. 2012
 - Brake override firmware update for in some **recent** models

Please be advised that the Brake Override System installation is now available for the following make and model vehicles:

Toyota Models	Model Years	Deadline
4Runner	2003-2009	3/16/16
Corolla	2009-2010	8/7/15
Corolla Matrix	2009-2010	8/7/15
Highlander	2008-2010	12/11/15
Land Cruiser	2008-2010	8/7/15
RAV4	2006-2010	12/11/15
Tundra	2007-2010	3/16/16

Lexus Models	Model Years	Deadline
LX	2008-2010	8/7/15
RX	2010	8/7/15

<https://www.toyotaelsettlement.com/>
24 Nov 2014

Bookout/Schwarz Trial

- October 2013, Oklahoma
 - Fatal 2007 crash of a 2005 Toyota Camry
 - Neither floor mat nor sticky pedal recalls cover this MY; no “fixes” announced
- Toyota blamed driver error for crash
 - Mr. Arora (Exponent) testified as Toyota software expert
 - “[Toyota’s counsel] theorized that Bookout mistakenly pumped the gas pedal instead of the brake, and by the time she realized her mistake and pressed the brake, it was too late to avoid the crash” [\[http://bigstory.ap.org/article/oklahoma-jury-considers-toyota-acceleration-case\]](http://bigstory.ap.org/article/oklahoma-jury-considers-toyota-acceleration-case)
- Plaintiffs blamed ETCS
 - Dr. Koopman & Mr. Barr testified as software experts
 - Testified about defective safety architecture & software defects
 - 150 feet of skid marks implied open throttle while braking



[\[http://money.cnn.com/2013/10/25/news/companies/toyota-crash-verdict/\]](http://money.cnn.com/2013/10/25/news/companies/toyota-crash-verdict/)

The Bookout/Schwarz Results

- **Jury awarded \$3 million compensation**
 - Key point in trial was whether ETCS design defects caused the fatal crash
 - To this day, **Toyota disputes that their ETCS is flawed**
 - \$1.5M each to Bookout and Schwarz estate
 - Toyota settled before jury could consider awarding **additional, punitive damages**
- **Subsequent Federal trials put on hold**
 - Only ETCS software/safety case to actually go to trial
 - Remaining Federal trials deferred
 - Mass settlements proceeding during 2014
 - **Hundreds of cases pending being settled as of summer 2014**

US Criminal Investigation

- **“Toyota Is Fined \$1.2 Billion for Concealing Safety Defects”** — March 19, 2014
 - Four-year investigation by US Attorney General
 - Related to **floor mats & sticky throttle pedals only**
- **“TOYOTA misled U.S. consumers by concealing and making deceptive statements about two safety-related issues affecting its vehicles, each of which caused a type of unintended acceleration.”** [DoJ Statement of Facts]
 - Deferred prosecution for three years in exchange for fine and continuing independent review of its safety processes.
 - Toyota said in a statement that it had made **fundamental changes** in its corporate structure and **internal safety controls** since the government started its investigation four years ago.

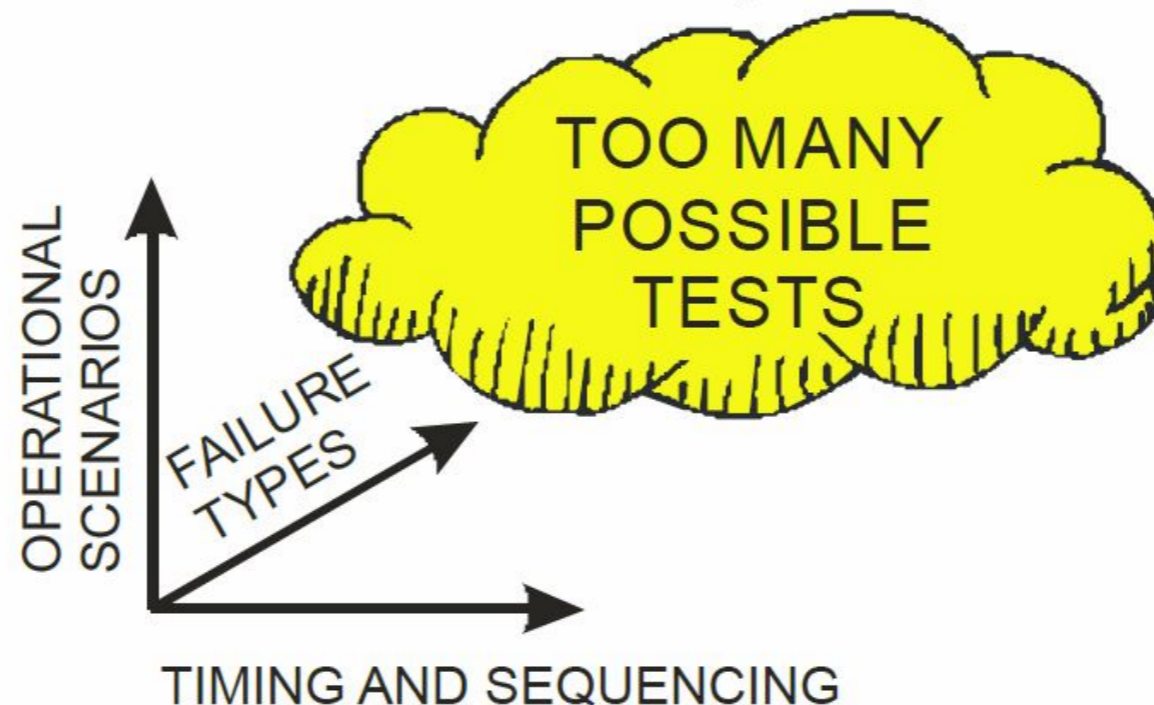
<http://www.nytimes.com/2014/03/20/business/toyota-reaches-1-2-billion-settlement-in-criminal-inquiry.html>

The Technical Point of View

- NASA did not find a smoking gun but found plenty of questionable practices, involving both hardware and software
- Jury found that ETCS defects caused a death
- We will look only at the software portion of the stack

Didn't Vehicle Testing Make it Safe?

- Vehicle level testing is useful and important
 - Can find unexpected component interactions
- But, it is impracticable to test everything at the vehicle level
 - Too many possible operating conditions, timing sequences
 - Too many possible faults, which might be intermittent
 - Combinations of component failures + memory corruption patterns
 - Multiple software defects activated by a sequence of operations



Testing Is Not Enough To Establish Safety

- Toyota **tested about 35 million miles at system level**
 - Plus 11 million hours module level software testing
([NASA report p. 20], covering 2005-2010 period)
 - In 2010 Toyota sold 2.1 million vehicles [Toyota annual report]
- **Total testing is perhaps 1-2 hours per vehicle produced**
 - Fleet will see thousands of times more field exposure
 - Vehicle testing simply can't find all uncommon failures

IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 19, NO. 1, JANUARY 1993

The Infeasibility of Quantifying the Reliability of Life-Critical Real-Time Software

Ricky W. Butler and George B. Finelli

life-testing of ultrareliable software is infeasible (i.e.,
to quantify $10^{-8}/\text{h}$ failure rate requires more than 10^8 h of
testing),

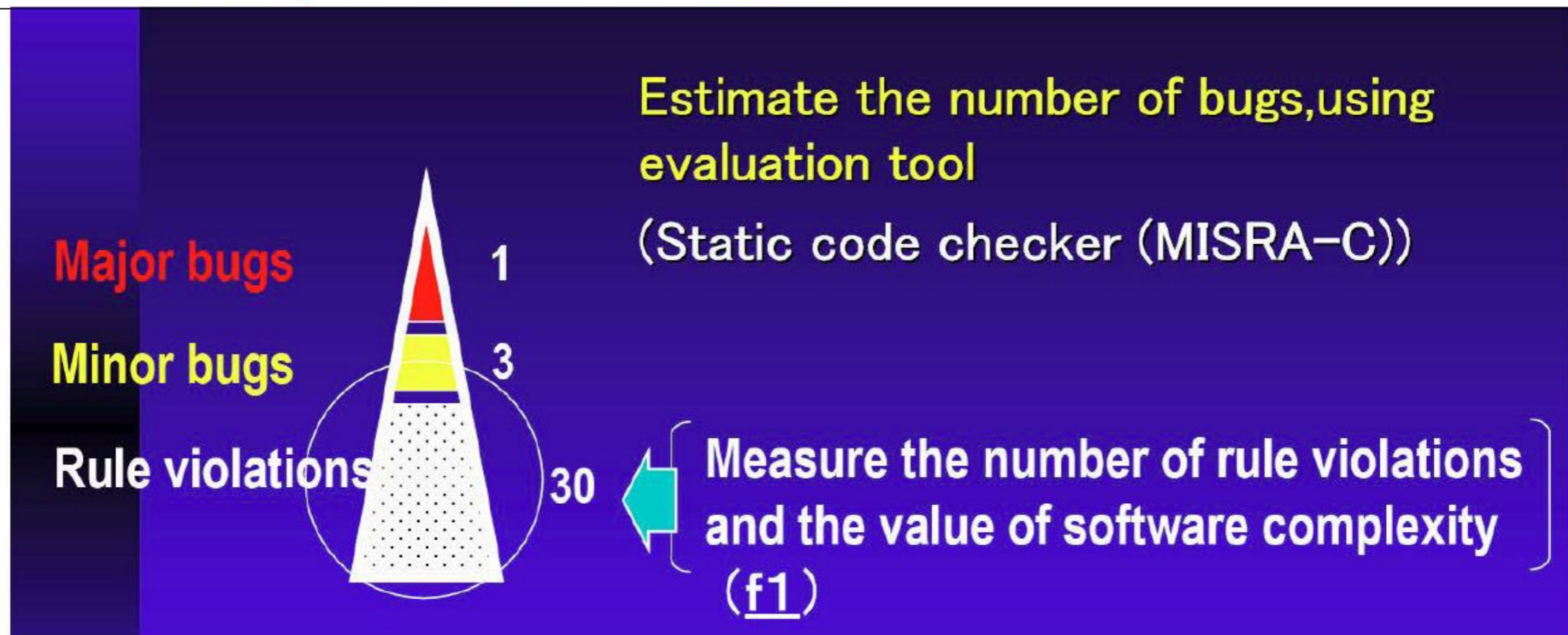
[Butler 1993, p. 10]

ETCS should probably be ASIL-C

		S1	S2	S3
C1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	ASIL A
	E4	QM	ASIL A	ASIL B
C2	E1	QM	QM	QM
	E2	QM	QM	ASIL A
	E3	QM	ASIL A	ASIL B
	E4	ASIL A	ASIL B	ASIL C
C3	E1	QM	QM	ASIL A
	E2	QM	ASIL A	ASIL B
	E3	ASIL A	ASIL B	ASIL C
	E4	ASIL B	ASIL C	ASIL D

What About Software Bugs

- For example what if MISRA C rules are violated?
 - MISRA C is a safer subset of C programming language developed for automotive use



Toyota data on infotainment software shows an expected **one “major bug” for every 30 coding rule violations**

Toyota Source Code Quality

- Coding style: rules for code formatting & language use
 - Toyota claimed 50% MISRA C overlap for coding rules
 - But only 11 MISRA C rules out of MISRA C in the Toyota coding rules
 - Toyota did not always follow its own coding rules
 - For example, 105 out of 343 “switch” keywords without “default”
[e.g., NASA App. A pp. 21-23]
 - Reason given for not using MISRA C rules for 2002 MY:
Its coding rules pre-date 1998 MISRA C [NASA App A p. 28]
- 35 MISRA C rules that NASA tools could readily check:
 - 14 of 35 rules violated; 7,134 violations
 - Mostly macro use and use of #undef [NASA App. A. p. 29]
- Mike Barr’s team found **80,000 violations** of MISRA C
[Bookout 2013-10-14PM 44:19-45:2]

NASA ETCS Static Analysis Results

- NASA used several static analysis tools [NASA App. A, pp. 25-31]
 - Toyota did not claim warning-free on these, but results informative
- Coverity:
 - 97 - declared but not referenced
 - 5 - include recursion
- Codesonar:
 - 2272 - global variable declared with different types
 - 333 - cast alters value
 - 99 - condition contains side-effect
 - 64 - multiple declaration of a global
 - 22 - uninitialized variable
- Uno:
 - 89 - possibly uninitialized variable
 - 2 - Array of 16 bytes initialized with 17 bytes

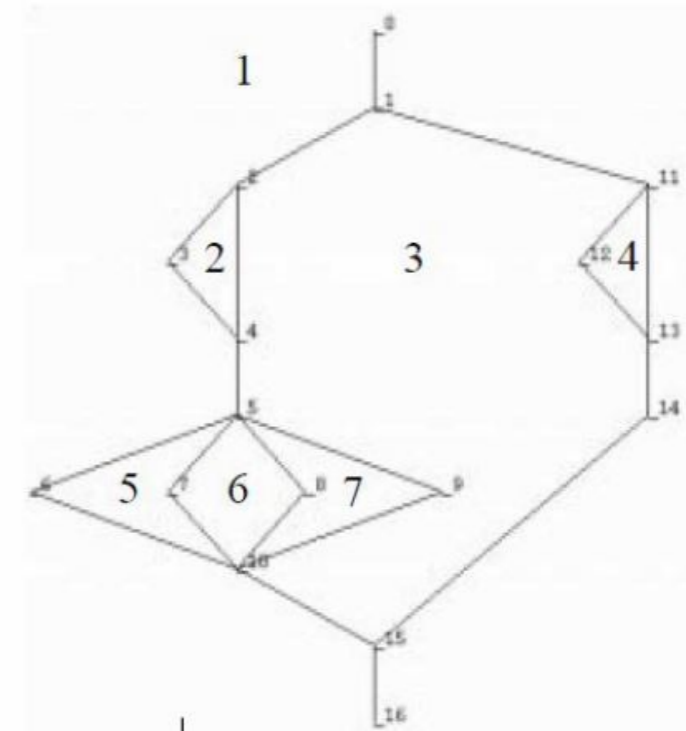
Code Complexity

“Spaghetti code”:

Incomprehensible code due to unnecessary coupling, jumps, gotos, or high complexity



- McCabe Cyclomatic Complexity metric
 - Number of “eyes” in flow control graph
 - Unit tests harder with complex graph
 - **Over 50 is considered “untestable”**
- Toyota ETCS code:
 - 67 functions with complexity over 50
 - **Throttle angle function complexity = 146;**
1300 lines long, no unit test plan
[Bookout 2013-10-14 31:10-32:23; 32:15-23]



Complexity=7

[NIST 500-235, 1996, pp. 28-29]

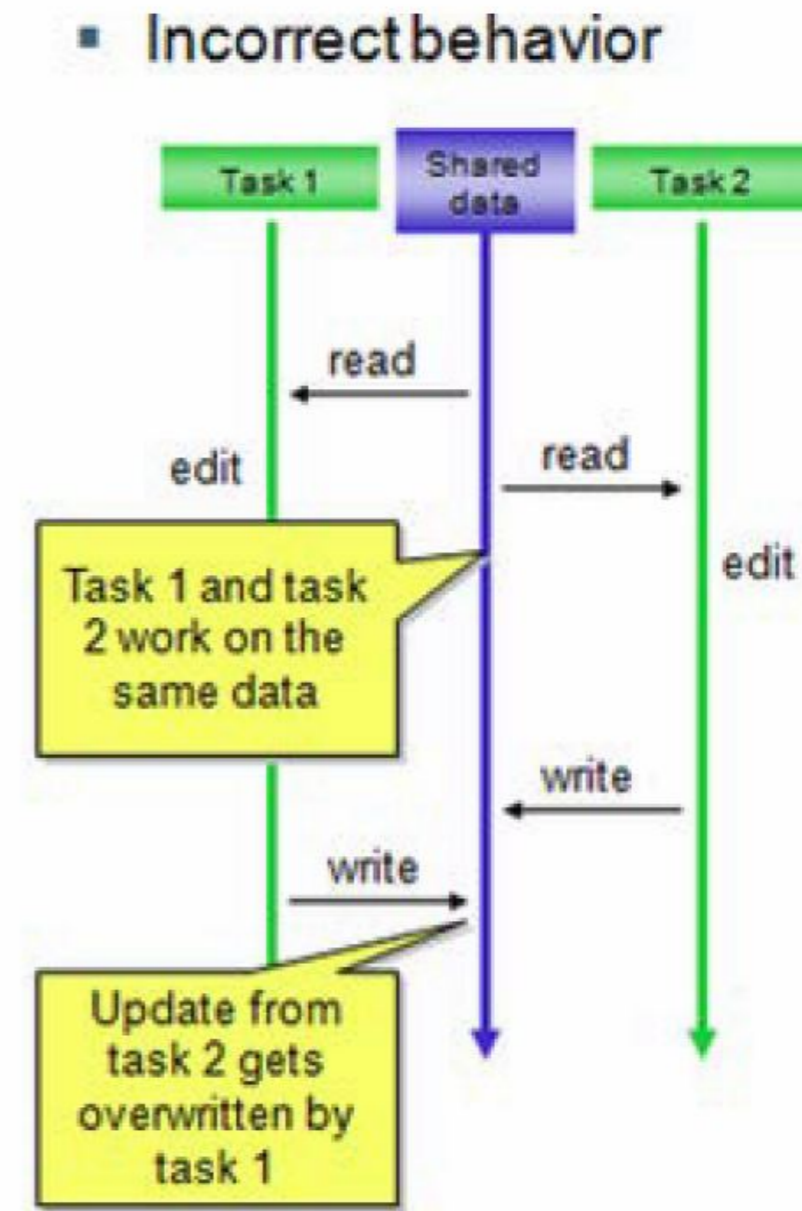
As the number of branches in the module or program rises, the cyclomatic complexity metric rises too. Empirically, numbers less than ten imply reasonable structure, numbers higher than 30 are of questionable structure. **Very high cyclomatic numbers of more than 50 imply the application cannot be tested, while even higher numbers of more than 75 imply that every change may trigger a “bad fix”. This metric is widely used for Quality Assurance and test planning purposes.** [RAC 1996, p.124]

Global Variables in Toyota

- **Ideal number of writeable globals is ZERO**
 - OK to have moderate “const” values and configuration data:
 - Toyota code has: [NASA App. A p. 33]:
 - 4,720 read-only & text variables
 - 11,253 read/write variables
- **ETCS globals command throttle angle**, report engine speed
[Bookout 2013-10-14 PM 29:4-15]
- **Toyota: 9,273 – 11,528 global variables**
[NASA App. A pp. 34, 37]
 - “In the Camry software a majority of all data objects (82%) is declared with unlimited scope and accessible to all executing tasks.”
[NASA App. A, pg. 33]
 - NASA analysis revealed: [NASA App. A, pg. 30]
 - 6,971 instances in which scope *could be* “local static”
 - 1,086 instances in which scope *could be* “file static”

Concurrency Bugs / Race Conditions

- One CPU can have many tasks (e.g., with Real Time Operating System)
 - Tasks take turns, sharing the CPU and memory (“multi-tasking”)
- Concurrency defects often come from incorrect data sharing
 - One way to fix this is to **disable interrupts** before reading to ensure one task reads/writes at a time
 - Defects may be due to subtle timing differences, and are often **difficult to reproduce**



[Wind River]

ETCS Concurrency Bugs

- NASA identified a specific **concurrency defect**

This rule is based on the fact that equal priority tasks cannot interrupt each other. Both can still be interrupted by a higher priority task. If because of this interruption the second task does not complete, and the first task restarts in the next time interval, **it could still overwrite the result of the interrupted second task.**

[NASA App. A pp. 33-34]

- **Nested scheduler unlocks** [Bookout 2013-10-14PM 21:10-22:1]
- **Shared global variables not all “volatile”** [NASA App. A pp. 33-34]
- **Shared globals not always access with interrupts masked**

If two tasks running at different priority levels access the same data, then the lower priority task must use interrupt masking to protect against interference from the higher-priority task.

[NASA App. A pp. 33-34]

This rule is not always followed in the code. In a few cases, the lower priority task merely sets a flag before entering its critical section and the higher priority task checks this flag before accessing the same data.

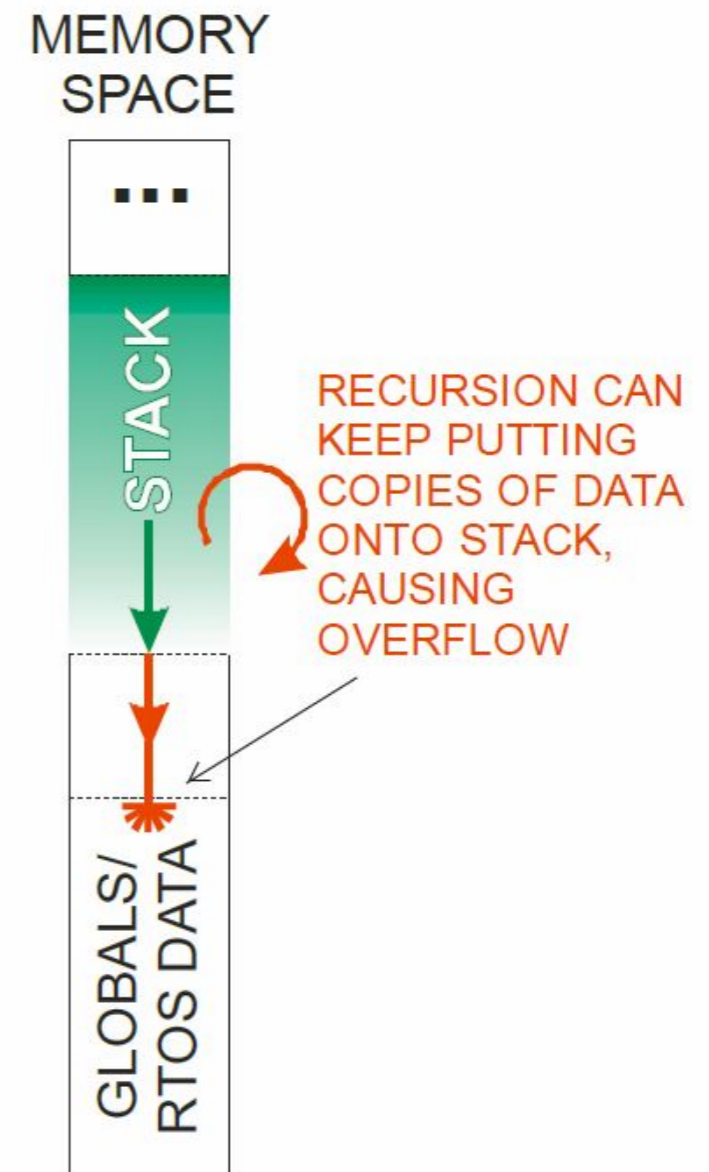
There are cases in the code where tasks of different priority levels (e.g., levels 14, 12, and 4) access the same global variables without using interrupt masks (pattern: read, store local copy, update, write new value). An example is the use of variable `s2s_eafsfb_gaind`, which is declared as a *non-volatile* static variable. This use would appear to be in violation of coding rule 651.

Despite the rigor in the use of the *volatile* qualifier on *constant* data, other shared global variables are not always declared *volatile*. The team counted **11,528** non-constant, shared global variables in the code.

There are only 865 uses of interrupt masking in the code, in 194 different source files. This indicates that access to global variables is not always done under protection of interrupt masks.

Toyota ETCS and Recursion

- Safety rules typically **forbid recursion**
 - Risk of stack overflow
 - Recursion makes it impossible to do the V&V necessary for a system like the ETCS [NASA APP A. p. 129]
- Toyota ETCS uses recursion
 - Stack is 94% full PLUS any recursion [Bookout 2013-10-14PM 35:7-24]
- **No mitigation for stack overflow**
 - Incorrect assumption that overflow always results in a system reset [NASA APP A pg. 130]
 - Memory just past stack is **OSEK RTOS area** [Bookout 2013-10-14 PM p. 39:1-5]



Rule 70 (required): Functions shall not call themselves, either directly or indirectly.

This means that recursive function calls cannot be used in safety-related systems. Recursion carries with it the danger of exceeding available stack space, which can be a serious error. Unless recursion is very tightly controlled, it is not possible to determine before execution what the worst case stack usage could be.

[MISRA C, page 43]

Safety Culture

- “No knowledge at Toyota” for some parts of the “V” process (e.g., module inspections)
 - No independent certification for parts they couldn’t/didn’t check
[Bookout 2013-10-11 PM 32:22-33:19]
- Example of Toyota’s UA investigation philosophy:
 - “In the Toyota system, we have the failsafe, so a software abnormality would not be involved with any kind of UA claim.”
(Employee tasked with examining vehicles with reported UA problems)
[Bookout 2013-10-11 PM 35:6-21]
- 2007 e-mail internal Toyota e-mail says:
 - “In truth **technology such as failsafe is not part of the Toyota’s engineering division’s DNA.**” ...
... “Continuing on as is would not be a good thing.”
[Bookout 2013-10-14 PM 96:20-97:31]

Other Issues

- **Poor isolation of task functions**
 - “Kitchen Sink” “Task X” both computes throttle angle AND is responsible for many of the failsafes (same CPU, same task). **Brake Override function in 2010 MY Camry is in this same task.** [Bookout 2013-10-14 AM 80:5-82:16]
- **OSEK RTOS not certified; 80% CPU load** (> 70% RMA limit)
[Bookout 2013-10-14PM 42:6-25] [NASA App. A p. 119]
- **Many large functions**
 - 200 functions exceeded 75 lines of non-comment code [NASA App. A p. 23]
- **Reviews informal and only on some modules**
[Bookout 2013-10-11 PM 29:24-30:5; 2013-10-14 49:17-21]
- **No formal specifications** [Bookout 2013-10-11 PM 29:24-30:5]
- **No bug tracking system** [Bookout 2013-10-14 PM 49:3-50:23]
- **No configuration management** [Bookout 2013-10-11 PM 30:7-10]

Some Legal Concepts

- The trials have been civil, not criminal
 - “Beyond reasonable doubt” is a criminal standard
 - not applicable in the death & injury lawsuits
 - US typical decision threshold for **civil lawsuit** liability is:
 - **“More likely than not”**
 - For product defects, common ideas are:
 - Was **reasonable care** exercised in the design?
 - For example, were accepted practices followed?
 - **Unreasonably dangerous**/defective for intended use?
 - Would it have been economically feasible to cure the defects?
 - The defects were a **plausible cause of the loss event**
 - Not necessarily the *only* possible cause
- (Laws vary by US State; these are just common ideas)