CI: AEBD-HW-DSLab

Automatic Emergency Break - Design
DSLab AEB-Design

August 24, 2021

Organisation:	Distributed Embedded Systems Lab
Working Gruop:	Safety OS Working Group
Author:	Nicholas Mc Guire
Release:	3
Revision:	0
Revision Number:	0.3
Date:	June 15, 2021
Expires:	June 14, 2022
Ref:	IEC 61508 Ed 2
Status:	**Incomplete, Unreviewed, Draft**
Format:	LaTeX
Tracking:	GIT
QA:	initial review, anual review
License:	Creative Commons CC-BY-SA-4.0

Contents

ChangeLog

0.1	Nicholas Mc Guire	2021-05-15	draft structure
0.2	Nicholas Mc Guire	2021-05-26	initial design and ad-hoc decomposition
0.3	Nicholas Mc Guire	2021-07-02	merge of HL results

Reviews:

0.1			

Authorization:

0.1			

Status: incomplete, unreviewed, draft

1 Introduction

This document covers the design of an automatic emergency breaking system designed from scratch. While the functionality is not really new in any meaningful way the key aspect is to develop this Automatic Emergency Break (AEB) with safety built in from the very first step. To achieve this we are using a modified version of Hazard and Operability Study (HAZOP) — Hazard driven Decomposition, Design and Development (HD^3) that strives to extract a design for a safety related system by iterative breadth-first search for all possible hazards. If these hazards can be eliminated by taking design decisions this is done, where this is not possible design extensions to add indications and/or mitigations are proposed in the form of Safety Application Conditions. For the details of the HD^3 process see "Contextualized Safe Functions — building complex safety related systems" (CI: CSF-FFG-OT) and "SIL2 Hazard driven Decomposition, Design and Development" (CI: HD3-SIL2-OSADL)

The AEB introduced strives to resolve the problem of vehicles colliding with pedestrians as well as objects. The focus on pedestrians is motivated by the asymmetry of protection and the limited capability of humans in complex traffic situations (even in slow-traffic) to manage the stochastic motions of surrounding pedestrians that potentially could collide with the vehicles perceived trajectory.

While technically much of this may seem resolved — there are ample prototypes available of such systems — the focus of such initiatives in general seems to have been functionality and demonstration of feasibility. Our goal here is to demonstrate:

- Safe and maintainable design of such a complex element
- Minimization of the system by full contextualization
- Extensive use of pre-existing elements in safety related systems
- Extract, at least partially, safety requirements and a partial solution space for using Artificial Inteligence (AI)/Machine Learning (ML) for safety — a de-facto unaddressed issue at this point.

Explicitly we state that the goal is not to create some highly innovative or new solution to the AEB problem it self.

1.1 Note on Design Notation Used

While designs are often done in these days we are using one of its predecessors called Structured Analysis/Structured Design (SASD) [?]. The key elements used are Data and control context diagrams as well as Data-Flow Diagrams. In addition FSMs and there transition tables along with data-dictionaries are used.

While there are more powerful design notations available we have found them to be sufficient and easy to introduce in the team without the common UML related confusions.

1.2 Note on Target Fault-class

When reading the analysis it sometimes may seem that "obvious issues" are missing. While this of course may be the case — missing a credible deviation is always possible, even in a team effort

— the focus of HD^3 is to pinpoint systematic faults not random faults. Notably at the technology agnostic level. Architectural protections for random faults may be considered at later levels and there is some cross-impact of mitigations for systematic and random faults, but essentially the focus is on systematic (specifically design and requirement level) faults. Thus a perceived incompleteness that results from a random fault is acceptable, if you find an omission related to a credible systematic fault we would like to hear from you.

1.3 Document Structure

The document structure mostly follows the HD^3 analysis flow. Some initial remarks in the this introduction and on references is prepended. In the appendix the full (tabular) analysis results are available. Note that the analysis results are generated from the HD^3Tool and this currently does not include systematic spellchecking nor any grammatical corrections, thus the wording may seem a bit wild sometimes.

For the full session logs visit (TODO: ref-URL for public AEB session repo)

2 Normative references

The following referenced documents are assumed to be available and known, at least at an overview level, for the effective application of this document. If you have not been exposed to International Electrotechnical Commission (a non-profit, non-governmental international standards organization) (IEC) 61508 Ed 2 we recommend to read part 0 of IEC 61508 Ed 1 first and then part 1, 2 and 3 of IEC 61508 Ed 2 in that order with part 4 (Terminology) on your desk — note that terms in IEC 61508 Ed 2 need not be defined as you might expect from your engineering practice so we recommend checking the index of part 4 regularly when reading part 1, 2 and 3.

References stated with version (e.g. IEC 61508-3 Ed 2) only apply to exactly that edition. For undated/non-versioned references, the latest edition of the referenced document as of June 2021 shall be consulted.

- ISO/IEC 51, *Safety aspects — Guidelines for their inclusion in standards*, Edition 3, *IEC*, 2014
- IEC 61508 *Functional safety of electrical/electronic/programmable electronic safety-related systems — part 1,2,3,4 and 7 (0*[1]*, Edition 2, IEC, 2010*
- *IEC 61882 Hazard and operability studies (HAZOP studies) — Application guide, Edition 1, IEC, 2001*

References to specific clauses or tables that are unmodified citations are given in *italics.* When a citation is shortened or partial and there is a possibility of 'going out of context' we will give the reference but not highlight the citation in italics — if unsure consult the normative reference please.

[1]part 0 is available for IEC 61508 Ed 1

3 Design Intent

Any system being built is a new system to some extent. In this case the novelty is quite limited with respect to the functionality it shall provide but rather the novelty we strive for is the design structure. The starting point of a new system is a, sometimes quite informal, statement of an intent. This intent describes the primary purpose, if you will, the selling point, of the new system — with other words the main functionality and value proposition.

For the AEB the design intent is also the starting point of the design process using HD^3. The goal of the design intent statement is to serve as a high-level context for the initial analysis as well as leave sufficient maneuvering space so that the actual solution can emerge from analysis — with other words, the design intent shall not actually limit technology or structure but really only clarify the behavioral/functional characteristics.

The AEB shall continuously sense the environment to detect objects which could be in the vehicles trajectory and result in collision. Based on the policy provided, the AEB shall make the decision on intervening by directly controlling the breaks if collision is determined to be unavoidable given the current estimated trajectory. Any intervention as well as internal failures of the AEB shall be communicated to the driver by audio/visual alarm.

AEB Design Intent - DSLab 2020

Note that we are assuming a driver that is basically in control thus safety considerations could draw on ISO 26262 Ed 2 — most notably the controllability attributes used to de-rate ASIL levels. We will though not make use of this as IEC 61511 Ed 1 clearly states that the ability of a human to take proper decisions strictly depends on the training level as well as on the stress level.

Protection layer	PFD
Human performance (trained, no stress)	$1,0 \times 10^{-2}$ to $1,0 \times 10^{-4}$
Human performance (under stress)	$0,5$ to $1,0$

Table 1: Excerpt from 61511 3 Ed 1 Table F.4 “Typical protection layer (prevention and mitigation) PFDs”

Thus while controllability may contribute to safety of a function if this is a routinely active functionality where drivers attention can (or must) be assumed, the AEB is a safety function that the driver is not directly involved in. In fact it must be assumed that its use would be predominantly to manage situations of stress, insufficient attentiveness (possibly aggravated by insufficient training) and thus the AEB is active in situations where the controllability would be very low (PFD of 0.5) to non-existent (PFD of 1.0). To make matters worse it must be assumed that alarming the driver during a situation in which the AEB would actually be needed to mitigate a hazardous constellation, could back-fire as it would cause additional distraction/stress.

3.1 DCD/CCD

The initial Data Context Diagram (DCD)/Control Context Diagram (CCD) is trivial enough that we join them into a single diagram. Essentially this is the initial formal interface for the AEB but this need not be the analytical system boundary ! The reason for this being that assumptions on input/output/impacts need to be made that may well go beyond the formal interfaces — an obvious example would be the consideration of driver response to an unexpected (even worse unjustified) breaking intervention.

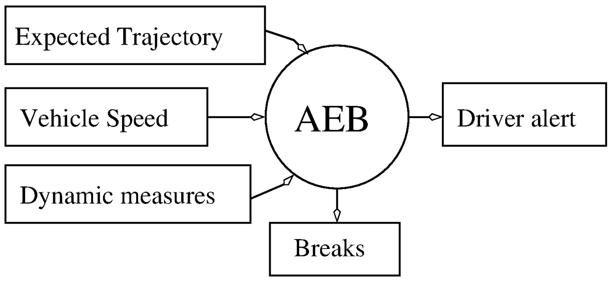

Figure 1: Initial top-level Data/Control Context Diagram

Note that we do not define the inputs/outputs but rather the requirements on the inputs shall emerge during analysis and the options/potentials for tho output(s) would equally emerge during analysis — e.g. the value of "break" and the assurance level (confidence) of the same might be an output potential, if this is then actually used by the receiving end is not in scope of this analysis/design.

3.2 Ad-hoc decomposition

This is a very weakly formalized step in HD^3 where the initial design intent statement is discussed and a first ad-hoc (or if you like expert opinion driven) apportionment of functionality to different logical units is made. Note that logical units here need not imply physical separation or likewise physical integration (an item could end up being split into two physically separated elements if analysis shows such a need).

1. Provide Policy
2. Sense Environment
3. Detect Object
4. Intervention Decision
5. Manage Alarm
6. Control Breaks

The order of items is not entirely random but rather a course grain decomposition of the core functionality that the AEB shall provide — "Manage Alarm" is a bit of an odd-ball here as it is not actually a sequentially integrated item (for a note on that in a moment)

This initial decomposition is now the basis for the first high-level round of sessions. Note that it makes relatively little difference if this first ad-hoc decomposition is complete or not — if the explorative analysis is complete then any missing elements will show up as Safety Application Condition (SAC)s during session and then (via SAC-consolidation) be joined to new logical units. Of course the final set and structure of units (the items of analysis) may differ depending on the starting point. In so far it is helpful to have a "reasonable" starting point for the initial design intent — with other words one does need a team with at least minimal understanding of the domain to reach credible designs. Note that this is not being claimed for our analysis team that is predominantly composed of graduate students of DSLab (Lanzhou University) and thus any results should be taken with adequate skepticism.

Vigilant readers will probably be a bit irritated by the absence of any initialization and shutdown unit. While this could be readily included based on prior experience it turned out in previous analysis to not be necessary as it will "emerge" during -consolidation quit naturally. This is an example of how HD^3 may allow to extract the minimum initialization needs based on those uncovered during the exploration of the intended function. As noted this is a weakly defined process step and one could question if the manage alarm is justified or should have been left to discovery during analysis — it technically (from a HD^3 perspective) would not be wrong to have left it out, this also applies to "Provide Policy", though the later was included to have one "trivial" item to start the analysis with. The other elements are not readily derived from analysis and thus are probably necessary or rather are actively designing this solution e.g. if the "Sense Environment" item would have been omitted then it would have emerged given the lack of any external interface to provide the data. The point here simply is that the starting point of the initial design intent is quite inaccurate but it seems that the methodology can handle this [2].

3.3 Initial FSM

While the initial decomposition informed us of the logical units that we expect (assume) it does not directly give us any indication of the units relations/dependencies. To gauge these relations we use a diagrammatic FSM along with an initial transition table 3. This first version (Figure 2) was the result of a team discussion and was then further refined The data that is passed is described in a very high-level form and should not (yet) be read as input to any software/hardware requirements. The high-level FSM is expected to stabilize at the end of the technology agnostic analysis phase (layers HL and HLD completed).

The transition table is used during the initial analysis to locate possible constraints/limitations to the data being exchanged. The vagueness of specification is to be expected at this point as we have no information on the actual data, just its functional intent, it is expected that by the end of the high-level technology agnostic analysis the data type is more rigorously defined. Thus the vagueness is an inherent property of the top-down analysis method rather than an commission of any sort.

Note the somewhat inconsistent FSM not actually allowing to loop directly — there is no connection from "Control Breaks" to "Sense Environment" — so this is kind of a "one-shot" FSM. Technically this is wrong, but for the purpose of initial analysis it is adequate and a more detailed specification

[2]We do have to note though that this claim is based on three HD^3 runs and a few trial sessions with toy examples only

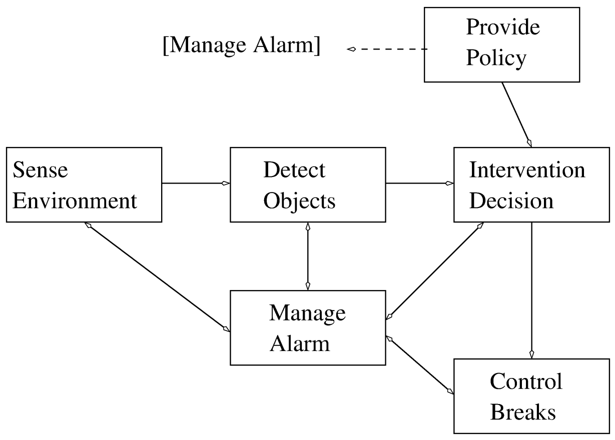

Figure 2: Initial (V1) FSM structure

```
  Provide_policy -> Intervetion_decision : policy_valid | policy_invalid
||
  Sense_env -> Detect_obj                : image_update [taged-image]
  Dectec_obj -> Intervention_decision    : object_list [record]
  Intervention_decision -> Control_breaks: brake | not brake [detailed brake info]
||
  Manage_alarm - - -> ALL                : state_ok | state_failed
  All -> Manage_alarm                    : element_status [enumeration]

[encoding:  -> data,  - - -> signal]
```

Figure 3: Initial (V1) FSM transition table

not yet possible — we simply do not know yet if this will be strictly sequential, overlapping or fully concurrent. For the design phase of course an updated FSM will be mandatory.

The distinction between signals and data is still quite weak here. Effectively signals are just binary data. The simplicity of the structure is not accidental, typically it is assumed that the active working set of concepts we can keep in mind during such an analysis session is in the range of 5 to 7. If the structure were much complexer than this it must be assumed that the analysis would not reach reasonable completeness. If this is not possible in a more complex system then maybe high-level decomposition will be necessary. During this session we experienced this problem during the analysis of "Detect Object" and updated the HD^3 process appropriately.

4 High-Level Design

Before going into the details of the high-level design a short contextualization is provided — for the full account pleas check the $HD^3_Concept$ (CI: CSF-FFG-OT) manual as well as the HD^3 manual (CI: HD3-SIL2-OSADL) in the SIL2LinuxMP repository.

Life-cycle Position

The high-level analysis is mostly addressing (and hence also covered) by clausses 7.2-7.6 of IEC 61508-1 Ed 2. The overarching goal is to build up sufficient understanding of the system and its environment as well as the overall (system level) safety needs. This is well expressed in the objectives of 7.2-7.6 (which we will not repeat here). The high-level analysis can be mapped to part 1 of IEC 61508 Ed 2 roughly as follows:

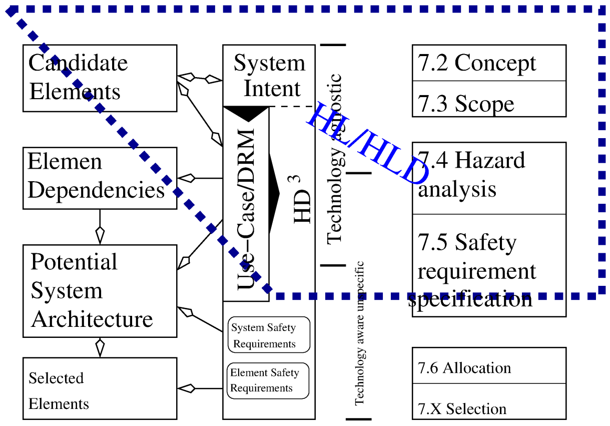

Figure 4: Scope of HL and HLD Layers in the HD^3 process

Normative Coverage

At the normative level the objectives of IEC 61508 1 Ed 2 Clause 4,5 and 6 apply to the HD^3 process (like to any other safety related process, measure, technique or organisation). The objectives that need to be directly addressed are then covered in IEC 61508 1 Ed 2 Clause 7.2 through 7.4 fully and to some extent 7.5 — it is though not expected that the system safety requirements uncovered during technology agnostic (layers HL/HLD) analysis will address all system safety requirements. For obvious reasons the technology aware unspecific requirements are not yet addressed (e.g. those related to usage of software or boolean logig if the same were then later selected).

The coverage is to be ensured by having all team members aware of the objectives stated in IEC 61508 1 Ed 2 which implies a reasonable safety basics background.

Process Dependencies

4.1 Ad-hoc Items

The ad-hoc decomposition was performed by the team starting at the design intent statemet (see Section 3). The initial decomposition emitted 6 items which are then developed in the high-level analysis. The 6 elements are:

1. Provide Policy
2. Sense Environment
3. Detect Object
4. Intervention Decision
5. Manage Alarm
6. Control Breaks

Note that these items may also evolve over time - so an initial analysis starts with an informal specification (see below) and this informal specification may be updated a few times until it is stable. Any such update does need to check on dependencies though. Here we only will present the final versions - the intermediate versions can be found in the git history respectively discussions in the session logs, if of interest.

4.1.1 Initial Item CSpecs

For each of these elements we now introduce the control specification or what we call "mini-spec". This is a relatively informal specification similar to the design intent statement.

Note that the depth of specification varies simply due to the subjective judgement of information needed for the intended unit/item. This does have a strong relation to the actual technical complexity of units but is not in any way formalized. For the high-level analysis the actual mini-specs did change (in general by addition) during the analysis process. For details consult the git logs as well as the irc sessoin logs.

1. Provide Policy:

> *The set of policies used to configure the AEB shall be provided by the provide policy item. Policy related data is assumed to be unchanged during the AEBs operational phase, that is updates (if any) are only while the AEB is not active*
>
> Provide Policy mini-spec - DSLab 2020

2. Sense Environment:

> *"Sense Environment" is a process that provides the data on the environment. It should transfer the information periodicly to the follow-up elements and should be a periodic task with bounded variability. It has the following limits.*
>
> Sense Env mini-spec - DSLab 2020

Limitations:
Design level limitations assumed during high-level analysis.

   a) The range of detection is limited
   b) Changes that are too fast will not be detectable
   c) Changes that are too slow may also not be detectable

3. Detect Objects:

   > *The Detect-Object process takes the data from "Sense-Environment" as well as the external vehicle inputs (trajectory, vehicle speed, basic dynamics meausres) and uses this to convert the information to a set of objects of interest along with basic attributes within a defined time limit.*
   >
   > Detect Object mini-spec - DSLab 2020

   Other inputs:
   The inputs are external to the AEB, where exactly they stem from is not yet determined (e.g. odometer from the vehicle or a AEB internal source would be possible). These inputs need further refinement and possibly addition — for the high-level analysis only these have been considered.
   a) Expected Trajectory
   b) Current vehicle speed

   Attributes:
   These are attributes that we assumed object could have. This is not a technical specification but may serve to extract such a specification in the lower layers of analysis. At the moment it primarily serves to define the limitations to the complexity of real objects taken into account.
   a) Basic: size, orientation, distance, speed
   b) class: vehicles, humans, other
   c) state: moving, non-moving, stationary
   d) relevance: intersection, non-intersection, unknown

   Limitations:
   Design level limitations assumed during high-level analysis.
   a) Ignore all objects below a defined threshold size

4. Intervention Decision:

This process has to decide if there is a potential hazardous environment constelation or not based on the braking policies, object list and the development of the objects (trending) as well as the development of the vehicle (trajectory). In case of a sure (highly probably unavoidable) collision being detected the decision to force-break the vehicle shall be made. Before the brake, it has to alert the driver to make decisions within a safety time buffer. The output is a brake info which conatins detailed parameters such as brake time, touch strenthen etc.

Intervention Decision mini-spec, DSLab 2020

Ad-hoc decomposition:
As the analysis showed that a meaningful interpretation of the guidewords for the overall process was not possible given the relative complexity of the overall task an ad-hoc decomposition into logical units was done. This is though not necessarily the structure/architecture of any actual implementation. The main processes assumed are as follows:

a) Read object list provided (from "Sense Environment")
b) Split list along slected class-set boundaries
c) Calculation of the objects trajectories (projection)
d) Calculation of intersection with expected vehicle trajectory
e) Calculate time to potential collision (or free space violation)
f) Check situation awareness:
   - Break status
   - Speed vs speed-limit
   - Minimum speed
g) Based on crash time to decide on break/no-break with brake info

Limitations:
The assumed limits of the decision process are:
a) Crash time are determined given the data (collision probability determined by given data).
b) Uncertainty of the environment constelation(precision?) (data uncertainty)
c) Possible adverse reason not to engage breaks even if collision seems unavoidable.

Intervention decision shall also notify the driver in the case that an intervention is taking place. TODO: decide if a notification/alarm to driver is also to be issued if the AEB is close to but still below threshhold (similar to a parking assisst warning To Be Done (TBD)).

5. Control Breaks:

Control brakes is a process that obtain the brake info from Intervesion decision to execute specific brake operations. The brake info should be providen periodicly and updated the current one.

Control Breaks mini-spec - DSLab 2020

Limitations:
The detail information such as braking time, braking strengthen and the like is technology aware, so they were not discussed here.

6. Manage Alarm:
(Not yet completed)

Short development statistics
This evaluation was done while HL was not yet completed. Specifically "Manage Alarm" was only half done. Never the less there is some indication that the lower complexity items reuse many SACs while the high-complexity items generate many though still reusing a significant number of existing SACs. It is too early to draw any real conclusions here it is though not going in the wrong direction — that is, if reuse were very low that would indicate that the method has no real advantage over isolated HAZOP session per item. We will have to wait for the complete data to call the case though.

item	guidewords	new SACs	reused SACs	%
Provide Policy	5 (50%)	22	2	9%
Sense Environment	8 (80%)	18	33	82%
Detect Object	8 (80%)	21	42	68%
Intervent Decision	7 (70%)	19	44	55%
Control Breaks	6 (60%)	1	37	45%
Manage Alarm	2 (of 5)	0	29	-

Table 2: Preliminary evaluation of guidword application and SAC reuse

Note that this only pertains to the basic guidewords and did not yet address security or concurrency — insofar this table is an intermediate result and not yet final, not even for HL !

4.1.2 FSM0_HL

The high-level was derived from discussions as well as the analysis of the first elements. The final agreed V3 5 was effectively identical to the initial one (V1) — V2 was a closed loop (fullly sequencial) though we concluded that this is actually not a reasonable assumption given that concurrent and/or overlapping computing can reasonably be expected.

Note that there are not transition events given in the current FSM this is simply due to it being at the technology agnostic level and naming events might not make that much sense yet - e.g. humans do not check states of ariables during communication. Aside from this limitation the does not yet contain the derived items (see Section 4.2.4 below)

TODO: V3 Transition Table

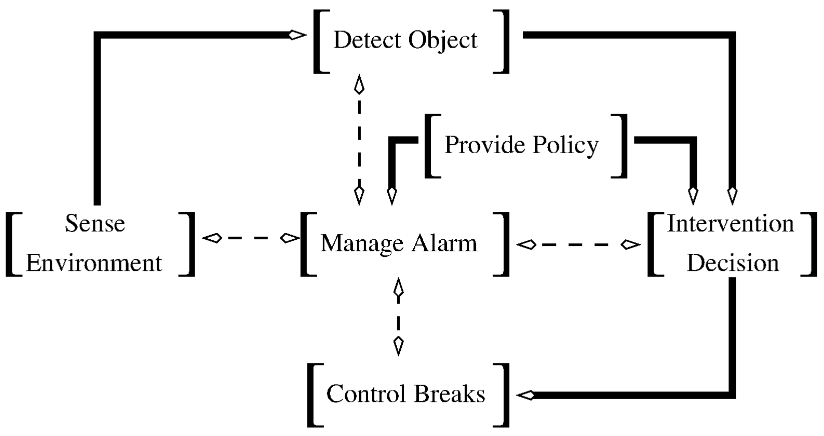

Figure 5: High-level state transition diagram V3 - layer HL

4.1.3 DFD0_HL

4.1.4 Data Dictionary

This initial data dictionary may seem a bit odd at first, typically a FSM would expect to carry state-transition tags. At this high-level the state is implicid insofar as the items will need to establish the validity of transition, at least in part, based on the data communicated. The actual analysis steps will refine these data itmes (e.g. split them up, specify details and attributes) and then (probably) allow to have state variables communicating the overall FSM state.

One note on the “none” entry for “Control Breaks” seems warranted as well: Since this ad-hoc decomposition does not make any (documented) assumptions about context the outputs are not covered in this data dictionary. This also will be changed during analysis (as soon as the appropriate design constraints have emerged).

1. Element: Provide_policy (PP)
   - Name: policy_valid — policy_invalid
     - Description: Communicate validity of available policy configuration (assumed to be pre-operationally defined) to the decision making instance
     - Origin: Provide_policy -¿ Intervetion_decision
     - Type: record
     - Rational: The avilability of a valid policy is a pre-requisite to safe operattion of the emergency breaking system. The provided policy is in an adequate descriptive form to allow decision making or in a form identifiable as ”no policy” with other words an invalid policy hence the type of a record even though valid/invalid may be read as hinting a binary data type.
     - Rate: one-shot (atleast conceptually - this does not exclude the possibility of retries).

  - Location: global shared

2. Element: Sense_environment (SE)
   - Name: image_update
     - Description: Image update indicates to the follow-up element that a new image has been made available
     - Origin: Sense_env -¿ Detect_obj
     - Type: record
     - Rational: The information of an updated image carries different meta data attributes describing the update process and7or updated image. This could include time-stamps, sequence numbering a image name or identification of image source.
     - Rate: periodic
     - Location: shared
   - Name: taged-image
     - Description: The actual image data along with associated meta-data (tag) whereby the tag need not pertain to the image content (logical content) but rather relates to the meta-data considered relevant for processing of the image.
     - Origin: Sense_env -¿ Detect_obj Type: record
     - Rational: The actual complexity of the tag is not yet known and may need extensions to cover different deviations uncovered during analysis the record is a flexible data type. Note that while the record is marked as shared we do assume that it would only reside in the resources of a single entity operating on it.
     - Rate: periodic
     - Location: shared

: Detect_objetcs (DO)
   - Name: object_list
     - Description: The basis for deciding an intervention is the object list and the current assumed trajectory. While the trajectory (and other vehicle status data) is provided by external sources the list of objects is provided by Detect-Object and contains the list of objects along with their attributes.
     - Origin: Detect_obj -¿ Intervention_decision
     - Type: record
     - Rational: The abstraction actual objects and their complex motion/bhaviors to set of their attributes allows to monitor complex elements based on a reduced set of crucial information and so helps keep the decision process simpler and understandable.
     - Rate: periodic
     - Location: shared

: Intervention_decision (ID)

3. Name: collision_risk
   - Description:
   - Origin: Intervention_decision -¿ Control_breaks
   - Type: numeric value
   - Rational:
   - Rate: periodic
   - Location: shared

: Control_breaks (CB)

- Name: None

Element: Manage_alarm (MA)

Name: state_fail—state_ok

- Description: indicate global health of system
- Origin: Manage_alarm -¿ ALL
- Type: Signal
- Rational: state_ok only indicates that the element can operate and there is no additional information to be conveyed - "uncondition continuation" state_fail indicates an error of the element. State:fail mandates active signaling elements while state_ok could be passive (polled on demand)
- Rate: async event
- Location: global

Name: element_status

- Description: indicate the element health status
- Origin: All -¿ Manage_alarm
- Type: Enumeratable
- Rational: element_status shall communicate the element health and in case of not OK a high-level reason - thus communicate one of a possibly set of conditions
- Rate: async event, event driven
- Location: global

Note that this data dictionary is effectively an input to the analysis and not based on the analysis. The rationale has though been updated later during the discussion process, while the analysis actually was under way. As with all design steps there are rarely sharp boundaries and the iterations also imply some level of overlap.

4.2 Derived Items

After the ad-hoc decomposition was driven through the HD^3 analysis we have a collection of SACs. These SACs now may be "local" in hte sense that they apply to one item only and would need to be provided there as reactive safety mechanism [3], or they can be of wider use (system or subsystem level, with other words, shared) and thus need to be provided by some additional item (or in theory by an additional interface to an existing item — but that option would not pop up in high-level design, rather this would be an architectural issue that is addressed later).

In the following sections we outline the SAC consolidation of the AEB and the derived items that result from the proposed allocation.

4.2.1 SAC Consolidation options

The currently still preliminary [4] SAC consolidation process has three options specified. These are:

1. Grouping by functional proximity (coupling/cohesion)
2. Grouping by criticality and hazard scope
3. Grouping by location in the system software architecture

We now iterate over the first two methods — for the full list of SACs and their consolidation see Annex **??**, Annex **??** — the reason the third is not considered is that at this level we do not yet have a system software architecture. The third method really only makes sense in the technology aware unpsecific and specific layers.

We will focus on the applied principles and the conceptual results here for the actual consolidation proposals, see the Annexes.

Grouping by functional proximity

Grouping by functional proximit resulted in two distinct process level groups and three new items as well as allocation of the majority of SACs to the specific items as safety functional requirements.

The proximity of SACs can be evaluted by looking at how often it is referenced in which item. If a SAC appears only in a single item then it is a reasonable candidate for integration as "reactive safety function". Note that this does not imply that it would be tollerable to operate this without any interference protection. Thus later phases of design may result in splitting an item to allow for this protection (non-interference). At the technology agnostic level where we do not yet consider these issues the "local SACs" are simply allocated to the item.

From the initial intent statement and the ad-hoc decomposition we arived at the mini-specs for the initial items that need to be analyzed. During this analysis SACs encode the item specific safety requirements, thus at the end of the SAC consolidation process we not only have derived items (the prime intent of the SAC consolidation) but we also have relatively specific safety functional requirements of the item — in context of the given design intent. These safety functional requirements are relatively precises or rather (ideally) minimal with respect to the design intent. So rather than incorporating generic solutions one limits safety functionality to the specifics of the design intent. This does limit re-use of items, at least initially. If repeated design cylces reveal patterns of use (e.g.

[3]e.g. checking a CRC and retrying in case of CRS mismatch would be a typical reactive mechanism

[4]We simply do not yet have sufficient experience to declare one of these routs to be the correct one

how time is managed) it may be reasonable to later extract semi-generic elements even in complex systems, but this is speculative at this point.

The new derived items startup, shutdown/safe-halt, System-timing/monitoring are the result of grouping SACs by cohesion (perceived technical and functional similarity) based on expert opinion — which is somewhat limited and could result in re-allocation, e.g. by severity based grouping. This is not a step that is fully determined by objective criteria and thus must be taken as somewhat uncertain. The design team should stay open for re-assessing allocation if the later analysis steps show accumulations of dependencies that could raise the severity to very high levels. Specifically in the case of this anylyisis this probably could apply to the system-timing-and-monitoring item — for now we leave it as one item, if analysis shows that joining these two functions results in interdependencies (e.g. monitoring only being reliable if timing is reliable but this is assured by the same item !) then it may be warranted to split it up into two items. For this analysis we started with these three derived items only.

The analysis of startup and shutdown/safe-halt is quite streight foward. Basically we again derive a mini-spec and then iterate using the guidewords. For specific questions during analysis (e.g. if a deviation is credible) one can inspec the origin as well as the use of the respective SACs that formed this item and so constraint the scope to the specific context. It is important to see it in this relation as the analysis of derived items may otherwise seem incomplete if one looks at it from a general perspective. The system-timing-and-monitoring item is a bit more involved as this item has two groups of SACs integrated: timing and monitoring safety functional needs. Thus during analysis it is reasonable to introduce a definition of timing and monitoring that covers the use-cases (again by inspection of origin and use of the respective SACs) and then iterate the analysis along these two main safety functional services being provided. This is somewhat comparable to the ad-hoc decomposition of complex items (e.g. Detec-Object) but driven by analysis results rather than engineering judgement.

We now introduce the three derived items that are covered by the High-Level Derived (HLD) analysis. Note that the list of SACs technically also includes **all** of the "UP" or process SACs that is those SACs that pertain to the process, organisation or qualification. Here we only list the technical SACs explicidly, which includes a certain inconsistency insofar as meta-data-is-verified was included without that there was a specific analytical trigger to do so - this is an example of "pure" engineering judgement — with other words we would simply expect that this would be "obvious". This also shows that HD^3 is a guided but not a mechanized analysis and it does heavily depend on engineering judgement ("brains-on analysis").

Due to scheduling problems the analysis of the derived items was done off-line. That is a proposal sent out by email and commented by team members and so iteratively consolidated. This process has not been formalized yet (and might actually not be that usable - lets see) so we include review findings here along with the analysis summary.

Grouping by criticality and hazard scope

Since the current HD^3 process does not include the criticality allocation yet (it has been preliminary formalized but not yet integrated into the tool) we have not yet run the alternative allocation. This is on our TODO list.

4.2.2 SAC Consolidation proposal - Consolidated items

The initial ad-hoc items analysis emitted a large number of SACs, some of these are “local” that is item specific and not re-used anywhere else (thus effectively safety specific design extensions). Others are problem specific and re-used infrequently, thus potentially also reactive safety functions assigned to the specific item and not generalized in a separate functionality (but this is of course a design decision based on engineering judgement and not directly on measurable facts).

In this section we consolidate the ad-hoc items, if necessary adjusting the mini-spec in the light of the safety related findings (SACs) and list the respective SACs in context. During consolidation the design intent and safety functional requirements (from the SACs) are reviewed and any findings reported.

1. Updated Mini-spec Provide-Policy:

> *The set of **system** policies used to configure the AEB **items** shall be provided by the Provide-Policy (PP) item. Policy related data is assumed to be unchanged during the AEBs operational phase, that is **no** updates **shall take place** while the AEB is active, The PP shall use well defined formats for data and meta-data as well as suitable encoding of safety related content including unique identifiers for all contents. Any data is to be verified and failure of verification including excessive delay (timeout) during provision, shall issue an appropriate alarm*
>
> Provide Policy - DSLab 2020

.

Referenced SACs:

ID	Short SAC
HLS 1	breaking-policy-configuration-prior-to-any-use
HLS 2	config-failure-issues-alarm
HLS 4	breaking-policy-shall-be-defined
HLS 6	meta-data-is-verified
HLS 10	well-defined-information-formats-including-meta-data
HLS 11	acknowledgement-is-checked
HLS 14	encoding-of-safety-related-content-verified-before-use
HLS 15	safety-content-carry-unique-encoding
HLS 22	specified-timeout

Functional Requirements

a) PP shall provide all static policies for AEB operations to other items

Safety Functional Requirements

a) All policy information is pre-defined and static during runtime (HLS_1)

b) Failure to successfully configure (initialize) PP within specified time shall issue an alarm (HLS, HLS_22)

c) failure to verify meta-data, policy information, encoding or unique IDs shall be treated as config failures (HLS_6, HLS_10, HLS_11, HLS_14, HLS_15)

d) Breaking policy shall be defined (HLS_4) **TODO: amend this by the full list of known policies**

Functional flow for Provide-Policy:

```
Provide-Policy {
  Retrieve DATA from permanent storage

  Verify META-DATA

  Provide POLICY in shared storage

  Notify System/Time-Monitor
}
```

Functionality: Basic set of functionalities required for PP:

a) Permanent storage - policy content and meta-data

b) Temporary/shared dynamic-storage - provision of policies to items

c) Notification - status and alarms

d) Timeservices - time and timeouts

Mapping to technologies:

a) Filesystem

b) Shared memory

c) Signals

d) Timers and Clocks

Allocation: This profile can be provided by IEEE 1003.1 POSIX compliant OS (TODO: check if we could map it to a 1003.13 profile)

2. Updated Mini-spec Sense-Environment:

> *"Sense Environment" is a process that gathers and provides data on the environment. It should transfer the information periodicly to the follow-up elements and should be a periodic task with bounded variability of execution time. It has the following limits.*
>
> Sense-Environment - DSLab 2020

Limitations:
Design level limitations assumed during high-level analysis.

a) The range of detection is limited
b) Changes that are too fast will not be detectable
c) Changes that are too slow may also not be detectable

Referenced SACs:

ID	Short SAC
HLS 6	meta-data-is-verified
HLS 24	input-threhold-is-checked
HLS 26	violation-of-threshold-transits-to-safe-state
HLS 27	runtime-calibrate-of-environment
HLS 28	sensor-limits-are-independently-recorded
HLS 31	sensor-off-line-testing
HLS 33	periodic-off-line-re-calibration
HLS 34	processes-used-are-well-defined
HLS 36	meta-data-is-pre-configured
HLS 59	input-images-are-time-stamped

Functional Requirements

a) The environment shall be optically recorded in a form permitting automatic processing of image data

b) Optical records of the environment shall be generated periodically and timestamped

c) Data shall be provided to the followup element(s)

Safety Functional Requirements

aCamera on-line calibration using recorded off-line testing/calibration results shall take place at runtime (startup) Image thresholds shall be checked at runtime Meta-data and communication is runtime verified (TODO: why do we have no SAC here for communication failures ? did we miss that ?)

Functional flow for Item Sense-Environment:

b)

```
Sense_Environment {
  Calibrate CAMERA

  Record IMAGES in dynamic storage

  Check THRESHOLD(s) of IMAGE-DATA

  Timestamp IMAGE

  Transfer IMAGE-DATA/META-DATA

  Update STATUS
}
```

Functionality Basic set of functionalities required for XX:

a) Environment sensing - camera(s)

b) Device access - camera(s)

c) Temporary/dynamics storage - image data and meta-data

d) Data communication - image provision to followup element

e) Timeservices - timestamp

f) Notification - status and alarm

Mapping to technologies:

a) Cameras

b) Device abstraction

c) Dynamic memory

d) Signals

e) Sockets (local network)

f) Timers and Clocks

Allocation This profile can be provided by an:

- IEEE 1003.1 POSIX compliant OS;

- A camera specific driver, and;
- The camera hardware.

Architecture note: The camera it self really needs to be treated as a separate item here and for the analysis of “ Sense-Environment” the target is the functionalities allocated to software.

TODO: introduce the Camera as separate item of analysis

3. Updated Mini-spec Detect Object::

> *The Detect-Object process takes the image data from “Sense-Environment” as well as the external vehicle inputs (trajectory, vehicle speed, basic dynamics meausres) and uses these, after runtime threshold checking, to convert the information received from “Sense-Environment”, to a list of objects along with basic attributes, within a defined time limit. Plausibilisation of the objects by thresholds, limits to change rates (dynamics) in context (traffic situation) based on a startup verified list, are continuously performed. In addition trending is performed for tuntime verification and data is timestamped before providing down-stream.*
>
> Detect Object mini-spec - DSLab 2021

Other inputs:
The inputs are external to the AEB, where exactly they stem from is not yet determined (e.g. odometer from the vehicle or a AEB internal plausibilization source would be possible). These inputs need further refinement and possibly addition. considered.

a) Expected Trajectory
b) Current vehicle speed

Attributes:
These are attributes that we assumed object could have. This is not a technical specification but may serve to extract such a specification in the lower layers of analysis. At the moment it primarily serves to define the limitations to the complexity of real objects taken into account.

a) Basic: size, orientation, distance, speed
b) class: vehicles, humans, other
c) state: moving, non-moving, stationary
d) relevance: intersection, non-intersection, unknown

Limitations:
Design level limitations assumed during Technology-Aware Unspecific analysis.

a) Ignore all objects below a defined size threshold
b) Limiting classification to vehicles, humans and other

ID	Short SAC
HLS 6	meta-data-is-verified
HLS 11	acknowledgement-is-checked
HLS 41	object-list-change-limit-exeeded
HLS 42	correct-initial-object-list-at-startup-verified
HLS 43	object-dynamics-threshold-continuously-estimated
HLS 44	check-for-object-change-after-stabilization-phase
HLS 45	object-list-plausibilisation-against-traffic-situation
HLS 46	limited-objects-list-dynamics
HLS 47	eccess-object-list-length-detected
HLS 48	excess-dynamics-invalidates-list
HLS 49	safe-state-transition-on-list-length-violation
HLS 51	failed-plausibilisation-invalidates-list
HLS 52	tracking-of-objects-over-time
HLS 53	defined-limit-for-reclassification-of-objects
HLS 54	multiple-independent-sources-of-object-classifiers
HLS 55	check-for-object-trending-change
HLS 56	Defined-threshold-of-trending-variability
HLS 57	defined-threshold-of-object-variability
HLS 59	input-images-are-time-stamped
HLS 61	input-time-stamps-are-verified-for-progress

Referenced SACs:

Functional Requirements

a) Detect-Object shall detect all objects greater than the defined size threshold classifying them as vehicles, humans or other.

b) Objects shall be tracked over time (HLS_52)

c) Basic object attributes shall be extracted and assigned to each classified object.

d) The list of verified objects shall be provided to the follow-up element.

Safety Functional Requirements

a) Continuous estimation of dynamics, plausibilisation, and reclassification thresholds shall be provided (HLS_43, HLS_45, HLS_46, HLS_51, HLS_53, HLS_57).

b) Object and object list thresholds shall be runtime verified (absolute length, dynamic , reclassification) (HLS_41, HLS_44, HLS_47, HLS_48, HLS_49, HLS_55, HLS_56).

c) Startup object list and meta-data is verified (HLS_6, HLS_42)

d) Timestamps images shall be used to verify progress (HLS_59, HLS_61)

e) Down-stream hand-over shall be verified (HLS_11)

f) Multiple independent sources of object classifiers shall be employed (HLS_54)

Functional flow for Item Detec-Object:

```
Detect_Object {
  Verify initial DATA and OBJECT_LIST

  Classify IMAGES in dynamic storage extracting OBJECT_LIST

  Track OBJECTS over time [TODO: review - this seems misplaced]

  Verify Progress (timestamps)

  Verify THRESHOLD(s) and Limit(s) of OBJECT_LIST

  Verify AGREEMENT of independent classifications

  Transfer OBJECT_LIST

  Update STATUS
}
```

Functionality Basic set of functionalities required for Detect-Object:

a) Temporary/dynamics storage - image-data, meta-data and object list

b) Data communication - object-list provision to followup element

c) Timeservices - timestamp/clocks for progress verification

d) Concurrency - for parallel execution of independent classifiers

e) Data-Analysis/Classification - object classification

f) Notification - status and alarm

Mapping to technologies:

a) Dynamic memory

b) Signals

c) Sockets (local network)

d) Timers and Clocks

e) Machine-Learning

f) Processes/Threads and thread/process management

Allocation This profile can be provided by an IEEE 1003.1 POSIX compliant OS. Machine Learning (ML) could be provided by HW and SW-midleware with a POSIX complient interface on top of the POSIX compoliant OS.

4. Updated Mini-spec Intervention-Decision:

This process has to decide if there is a potential hazardous environment constelation (probably collision) or not based on the braking policies, object list (provided by "Detect Object") and the development of the objects (trending) as well as the expected/projected vehicle trajectory. In case of a sure (highly probably/unavoidable) collision being detected the decision to force-break the vehicle shall be made considering any active intervention by the driver. Before initializing forced braking the break status (engaged see HL_36) shall be checked and the driver has to be alerted to make a decisions within a safety time buffer. The output is a brake info which conatins detailed parameters such as maximum permissible braking time, required breaking strenthen, etc.

Intervention-Decision - DSLab 2021

Ad-hoc decomposition:
The initial ad-hoc decomposition basiscally is retained with minor updates.

a) Read and verify object list provided by "Sense Environment"
b) Split list along slected class-set boundaries
c) Calculation of the critical objects trajectories (projection)
d) Calculation of intersection with expected vehicle trajectory
e) Calculate time to potential collision (or free space violation)
f) Check situation awareness (driver engaged break/reduces speed)
  - Break status
  - Gas-pedal status
  - Speed vs speed-limit
  - Minimum speed (e.g. highway)
  - Context (traffic situation)
g) Based on crash time to decide on break/no-break with brake info

Other inputs:
Additional inputs are needed for checking possible driver engagement

a) Current gas-bedal position (indication of situational awareness)

Limitations:
The assumed limits of the decision process are:

a) Crash time are estimated given the data (collision probability determined by available data).
b) Uncertainty of the environment constelation/traffic-situation (data uncertainty)
c) DESCOPED: Possible adverse reason not to engage breaks even if collision seems unavoidable.

Intervention decision shall also notify the driver in the case that an intervention is taking place.

Referenced SACs:

ID	Short SAC
HLS 6	meta-data-is-verified
HLS 11	acknowledgement-is-checked
HLS 63	processing-timeout-as-indicator
HLS 81	estimate-the-probability-of-completing-the-full-list
HLS 62	risk-probability-change-limit-exceeded
HLS 64	inconsistency-detection
HLS 67	stability-over-time-of-trajectories
HLS 75	on-line-comparison/verification-to-formal-model

Functional Requirements Note that the functional requirements may sound like safety requirements which they actually are at the system level. For the purpose of this specification the functional requirements pertain to the requirements derived from the intended (item) functionality and the functional safety requirements pertain to those (additional) capabilities introduced to assure the safety of the intended functionality.

a) Intervention-Decision shall estimate (trend) trajectories of relevant objects based on object information provided by Detect-Object as well as the estimated/expected vehicle trajectory.

b) Intervention-Decision shall detect probably collisions and quantify the risk (collision probability). as break-info.

c) In case of assessing a high probability of collision with a relevant object, Intervention-Decision shall evaluate the drivers allertness and if found insufficient, initiate a forced breaking while alarming the driver of the potential collision

Safety Functional Requirements

a) Supervise computational progress (HLS_63, HLS_81)

b) Runtime consistency and stability check (HLS_62, HLS_64, HLS_67)

c) Model based plausibilisation (HLS_75)

d) Verification of data and communication (HLS_6, $HLS_1 1$)

Functional flow for Item Intervention-Decision:

```
Intervention_Decision {
  Receive OBJECT_LIST

  Estimate TRAJECTORY of relevant OBJECT(s) and VEHICLE

  Trend OBJECTS over time

  Estimate RISK_OF_COLLISION

  Check DRIVER_ALLERTED

  Decide FORCED_BREAKING

  Transfer BREAK_INFO

  Update STATUS
}
```

Functionality Basic set of functionalities required for Intervention-Decisions intended functionality (this does not cover safety functional needs).

a) Temporary/dynamics storage - object lists and intermediate trending information.

b) Estimators

c) Decision-making

d) Data communication - break-info

e) Notification - status and alarm

Mapping to technologies:

a) Dynamic memory

b) Numeric estimators

c) Machine-Learning and/or Statistics

d) Sockets (local network)

e) Signals

Note that time related needs (timeout supervision) shows up in the safety functional requirements but not in the intended function. This may be a point to cross-check if there is any ommision.

Allocation This profile can be clearly provided by an IEEE 1003.1 POSIX compliant OS. ML/Statistics and Numberic estimators may be allocated to HW in part and/or software midleware elements.

5. Updated Mini-spec Control-Breaks:

   TODO

6. Updated Mini-spec Manage-Alarm:

   TODO

Consolidation findings

4.2.3 SAC Consolidation proposal - Derived items

Joining these initial groupings as well as mitigating some of the perceived weaknesses of the initial analysis (possible ommissions due to assuming previously defined SACs being in place - see also the findings from HL analysis in Annex **??**.

1. Mini-spec Startup:

   > *The startup shall conditionally setup (AEB Activated) and, where necessary initiailze, all resources that are necessary for operation of the AEB. A successful completion of initialization shall be signaled to the driver, failure to initialize shall place the AEB in a safe (inactive) state.*
   >
   > Startup, DSLab 2021

   Allocated SACs:

   Note that SAC 68 is not directly addressed by “Startup” but only to be used as a calibration input, that is an untrained system may need more stringent limits than a well trained system. Thus it may seem a bit misleading to refere to this SAC as ”allocated to startup” but this is the limit of the chosen abstraction and simply must be handled by sound engineering.

ID	Short SAC
6	meta-data-is-verified
60	startup-sanity-check
68	initial-training-of-AEB-by-observing-drivers

Analysis of "Startup" reveald only two new SACs and it introduced a constraint on initialization order that was not formulized as a SAC (though it maybe should have been). Since Startup also needs the ability to transit to a safe state any resources that this may need would necessarily be initilialized first. Following this timing would need to be initialized since timing is used in a number of places as indicator/trigger, and finally "the rest" whereby no specific order emerged and thus simply the functional order seems to be a reasonable (and understandable) order - thus: Startup order (which probably should be a SAC) is:

a) Startup it self

b) Safe Halt

c) Monitoring and timing

d) Everything else... simply using functional order (see FSM)

New SACs:

ID	Short SAC
HDS 1	resource-usage-monitored
HDS 2	excess-resourcs-usage-transits-to-safe-state
HDS 4	deviation-from-execution-order-transits-to-safe-state

For the new SACs introduced and the detailed analysis see the details in the Annex.

Referenced SACs:

ID	Short SAC
HLS 12	safe-state-transition
HLS 30	inconsistent-configuration-transits-to-safe-state

Note that all other SACs called for during analyisis of Startup were UP SACs which are applied at the process level and not item specific — this list does not imply that other SACs are not relevant for Startup see Section 4.3.

Review findings

- One ommission was detected: no possibility for the driver to disable the AEB !
- Note that HLS_12 can not be directly applied as it is not ensured during startup that Safe-Halt is already initialized Thus safe-state-trransition may need an appropriate extension to ensure it is effective even without pre-initialized Shutdown/Safe-Halt module.

2. Mini-spec Shutdwon/Safe-halt:

> *On detection of a critical deviation from intent, failure to initialize or deliberate deactivation of AEB by the driver, the AEB system properly, the AEB shall transit to a safe (inactive) state and signal status to the driver (failed or disabled). Safe state transition due to critical deviations from intent (errors/failures) shall be made known to the driver (e.g. logged).*
>
> Shutdown/Safe-halt, DSLab 2021

Allocated SACs:

ID	Short SAC
HLS 6	meta-data-is-verified
HLS 12	safe-state-transition
HLS 13	safe-state-defined
HLS 80	missing-part-threshold-check

Note that HLS_80 sounds very specific but actually it is quite generic as it is monitoring if individual steps completed sufficiently often so as to not shut down the system on a single (spurious) violation (which will diminish quality but not invalidate results completely — actually no result is really derived from a single cycle).

Including HLS_13 safe-state-defined is may seem a bit odd as well but essentailly how ever this item were implemented it would be defining the safet state, this does though not imply that it need not be specified in some way elsewhere.

Shutdwon/Safe-halt has an implicit limit as it can not call for safe-state transition in case of error if it is not properly initialized or if this error is internal to Shutdown/Safe-halt it self. This issue did not emerge during analysis of mitigations though it would be expected. Put differently Safe-Halt needs to re-define the safe-state (see HLD_4 deviation-from-execution-order-transits-to-safe-state) to cover very early failures and mandate the initialization order to cover later errors. The issue here is that if startup failed to initialize properly then we can not rely on the availability of Safe-halt

New SACs:

ID	Short SAC
HDS 6	defined-item-status
HDS 7	check-global-status-with-status-of-item

Referenced SACs:

ID	Short SAC
HLS 33	periodic-off-line-re-calibration

Review findings

- Possible ommision of spurious safe state transition (e.g. via “Early” or “Before”. Internal failure mitigation in the case of “safe-halt” might need to resort to deliverabe live-lock (e.g. infinite looping),
- HLS_33 needs to be adjusted to cover this case as its current definition is exclusively in the context of “Sense Environment” and would not make much sense in the given context. This review of HLS_33 seems to have been missed during analysis (or at least not recorded).

3. Mini-spec System/time monitor:

> *On detection of a critical deviation from intent, failure to initialize or deliberate deactivation of AEB by the driver, the AEB system properly, the AEB shall transit to a safe (inactive) state and signal status to the driver (failed or disabled). Safe state transition due to critical deviations from intent (errors/failures) shall be made known to the driver (e.g. logged).*
>
> System/time monitor, DSLab 2021

Allocated SACs:

The item has two related but distinct functionalities. The one group is time related and the other is time-independent monitoring (content monitoring in some form). This is also reflected in the analysis.

ID	Short SAC
HLS 6	meta-data-is-verified
HLS 39	runtime-verification-of-time
HLS 63	processing-timeout-as-indicator
HLS 65	continuous-verification
HLS 66	inconsistency-threshold
HLS 70	monitor-scenarios
HLS 78	excessive-low-risk-intervention-failure-takes-AEB-off-line
HLS 79	discrepency-between-DRMs-and-on-line-monitoring-data-takes-AEB-off-line

New SACs:

ID	Short SAC
HDS 8	unique-identification-made-for-safety-related-things
HDS 9	check-the-unique-identification-of-things-before-using

Referenced SACs:

ID	Short SAC
HLS 22	specified-timeout
HLS 27	runtime-calibrate-of-environment
HLS 39	runtime-verification-of-time
HLS 60	start-up-check

Review findings

- It is interesting that the need for unique identifiers was not raised earlier - verification of encoding of safety related data (HLS_14 and context specific also HLS_6) did appear and the unique encoding (HLS_15) did as wsell. As this was done in the off-line process the motivation for introducing the new SACs could not be fully determined — never the less HDS_8/HDS_9 seem mergeable with HLS_14/HLS_15.
- HLS_27 is not properly specified so while it is referenced here as an indication it really needs fixing to be usable
- HLS_39 is a circular reference as this High-Level SAC is actually allocated to this item and shall be consolidated. This may need tool support to catch such cirtulcar references.

- HLS_60 may be a bit too weak for the intended purpose as HLS_60 is checking availability (access) but not usability (e.g. time is advancing on the clock but maybe not reasonably so).

The three new items introduced generated some new SACs but re-used a large number of SACs, at the same time only a small number of process related (UP) SACs were found which indicates that the process related requirements are nicely stabilizing.

item	guidewords	new SACs	reused SACs	%
Startup	5 (50%)	4	26	30%
Safestate-Transit	3 (30%)	4	7	8%
Sys Time/Monitor	7 (70%)	2	36	39%

Table 3: Evaluation of guidword application and SAC reuse in the High-Level Derived Layer

Review findings Some of the UP SACs actually could have been found earler, if these are ommisions or if these really could not show up earlier is not yet clear. There does seem to be certain level of redundancy regarding UP SACs implying that while they would ideally be found at the earliest point in time, the risk of missing an UP SAC is smaller than for DOWN SACs (often item specific). Some of the SACs need to be merged and some really need to be reviewed as their specification is too vague.

4.2.4 FSM0/DFD0 Update

FSM V4:
Version 4 of the Finite-State-Machine diagram incorporates the High-Level Derived items (HD) as well as groups those items that are accessed by all (or almost all) items. Note that Startup formally also could reach Shutdown/Safe-Halt respectively Manage-Alarms but this would be limited to the point after which these had been successfully initialized which is probably a very short time-span, so diagramatically we do not connect Startup directly with these failure response and service providing items but rather assume that Startup would have a basic internal halt (e.g. busy-wait forever) implmented as a minimum local safe-halt.

Note that it well may be reasonable to merge some of the items in any practical implementation. Candidates for merging in this current phase could be:

- Startup and Provide-Policy
- Manage-Alarm and Sys/Time-Monitor

This should be considered during Technology-aware Unspecific (UT) analysis and is not decided here now.

```
Startup
  |
  |     .---------> Detect objetcs >------------.
  |    /              |                          \
  |   |        .-------------------------.        |
  |   |        |        .Provide Policy  |        |
  | Sense      |       /     \           |     Intervention
environment ---| Manage       Sys/time   |------ decision
               | Alarms       Monitor    |        |
               |-------------------------|        |
               | Shutdown/Safe-Halt      |        |
               `-------------------------`        |
                        |                         .
                        |                        /
                    Control breaks <------------
```

DFD0:

The update of the data-flow and control-flow diagram consists of clarification of control and data flows as well as the introduction of logical storage elements (shared data stores).

The DFD (which includes the control-flow elements) is not to be read as a architectural statement — concurrency is not yet considered further any notion of partitioning or spatial/temporal isolation is not yet available. Isolation needs can be expected to emerge from UT analysis (at least the first general isolation needs).

4.2.5 Data Dictionary Extension

Data elements
TODO
Control (signal) elements
TODO
Storage elements
TODO

4.3 Procedural Requirements

SACs that are not allocated to a particular functionality but that are ither process level or generic may be marked as UP. This means that they would be merged at the process level (e.g. coding-style) and not carried down to each and every element explicidly. For the High-Level SACS we decided to group them into two big categories: 1) Process level requirements and 2) Design and coding style. Note that the process level requirements do cover some of the generic safety requirements in IEC 61508 Ed 2 and while this may seem technically not necessary to derive them through this analysis the clear advantage is that we now have an at least semi-quantitative severity assigned to a particular generic measure (e.g. system testing) where normally we simply can not clearly judge how critical the same is. If this can be exploited is an open issue at this point.

Process amendments

System specific process amendments will in general be specializations of generic process demands as found in IEC 61508 Ed 2 for the respective Development Life-Cyle (DLC) phase. None of these process amendments shall be expected to replace generic process requirements (hence the name) even if in some cases it may be that the coincide (e.g. system tests) with demands from IEC 61508 Ed 2. In general they will help specify and to some extent weigh the generic requrements of the standard.

ID	Short SAC
HLS 0	Review-of-policies
HLS 3	defined-review-process
HLS 5	centrally-managed-specifications
HLS 8	Safe-element-behavior-fully-specified
HLS 18	development-process-be-monitored
HLS 35	well-defined-development-process
HLS 37	system-test
HLS 38	well-defined-DLC
HLS 73	well-defined-Design-Reference-Missions
HLS 74	Independent-validation-and-verification
HLS 76	Staged-deployment
HLD 2	follow-up-protection-shall-be-evaluated

Table 4: Process amendment based on High-Level and High-Level-Derived

Design and Implementation amendments

Design requirements from IEC 61508 Ed 2 are generic and are not replaced by the context specific requiremnts — some may address generic needs in part or in whole — in general they are though amendments. While High-Level and High-Level-Derived do not actually include architectural design steps and thus much of the design and implementation amendments will only manifest them selves concretely once the technology has been selected.

Findings: The general findings during the analysis of the derived items is very limited — ideally all general demands would have been found during high-level analysis ! — insofar the finding of HLD_2 and HLD_6 are to be seen as probable ommissions during the high-level analysis..

ID	Short SAC
HLS 7	resources-are-bounded
HLS 9	protocol-shall-be-specified
HLS 10	well-defined-information-formats-including-meta-data
HLS 11	acknowledgement-is-checked
HLS 14	encoding-of-safety-related-content-verified-before-use
HLS 15	safety-content-carry-unique-encoding
HLS 16	communication-uses-a-defined-hand-shake-protocol
HLS 19	information-contains-all-labels-and-settings
HLS 20	Verified-hand-shake-protocol-in-communication
HLS 21	information-receipt-shall-check-for-timeout-occurring
HLS 17	defined-sequence-of-operations-pre-determined
HLS 19	information-contains-all-labels-and-settings
HLS 23	input-timeout-supervision
HLS 25	violation-of-timeout-transits-to-a-safe-state
HLS 29	verification-failure-transits-to-safe-state
HLS 30	inconsistent-configuration-transits-to-safe-state
HLS 69	threshold-check
HLD 6	defined-item-status

Table 5: Design and Implementation amendments based on High-Level and High-Level-Derived

Acronyms

HD^3	Hazard driven Decomposition, Design and Development. 3, 4, 6–10
AEB	Automatic Emergency Break. 3
AI	Artificial Inteligence. 3
CCD	Control Context Diagram. 7
DCD	Data Context Diagram. 7
DLC	Development Life-Cyle. 36
HAZOP	Hazard and Operability Study. 3
IEC	International Electrotechnical Commission (a non-profit, non-governmental international standards organization). 5
ML	Machine Learning. 3
SAC	Safety Application Condition. 8, 18
SASD	Structured Analysis/Structured Design. 3
TBD	To Be Done. 13