CI: HD3-SIL2-OSADL

SIL2 Hazard driven decomopsition, design and development

SIL2 HD3

May 5, 2018

Organisation:	Open Source Automation Development Lab
Working Group:	Safety Critical Linux Working Group
Author:	Nicholas Mc Guire

Release:	0
Revision:	13
Revision Number:	0.13
Date:	April 30, 2018
Expires:	—
Ref:	IEC 61508 Ed 2
Status:	**unreviewed, unauthorized, incomplete, Draft**
Format:	LaTeX
Tracking	GIT
QA:	initial review and authorisation
License:	Creative Commons

Contents

1 Introduction **5**
1.1 Defining Complex safety related system . . . 5
1.2 complexity and context . . . 6
1.3 Managing complexity by context . . . 7

2 Scope **9**
2.1 Note on reading . . . 9

3 Normative references **10**

4 Terms and definitions **11**
4.1 Item . . . 11
4.2 Mid Complexity . . . 11
4.3 Derived item . . . 11

5 Overview **13**
5.1 Motivation for HD^3 . . . 13
5.2 Process overview . . . 14
5.3 Generated Artefacts . . . 16
5.4 Roles and Competence . . . 17

6 HD^3 process requirements **19**

7 Formal coverage of Annex-R **20**

8 Related considerations **23**
8.1 Safety life-cycle Management . . . 23
8.2 Relation to 7.X 'Selection' . . . 24
8.3 HD^3 and IEC 61508-3 Ed2 Annex C properties . . . 27
8.4 Tools qualification needs HD^3Tool . . . 28
8.5 Forward/Backwards Traceability . . . 29
8.5.1 Forward traceability . . . 30
8.5.2 Backwards traceability . . . 30
8.6 Process Documentation and Records . . . 31
8.6.1 Sessions . . . 31
8.6.2 Hazard and Operability Study (HAZOP) Tables: Item . . . 31
8.6.3 HAZOP Tables: SAC . . . 32
8.6.4 HAZOP Tables: TODO . . . 32
8.6.5 HAZOP Tables: Selection . . . 32

9 Hazard driven decomposition, design and development (HD^3) **33**
9.1 Short specification (overview) . . . 33

9.2 Method intent . . . 34
9.3 HD^3 context . . . 34
9.4 HD^3 design limitations . . . 35
9.5 HD^3 Proces Specification . . . 35

10 Design intent as starting point **39**
10.0.1 Design intent example . . . 39
10.1 Initial design decomposition . . . 40
10.2 Coliminder - Initial design decomposition . . . 42

11 Item analysis **43**
11.1 HAZOP . . . 44
11.2 HAZOP primer . . . 44
11.3 HAZOP extensions . . . 46
11.3.1 Managing concurrency . . . 47
11.3.2 Integrating security . . . 47
11.4 Recording and Traceability . . . 47

12 Derived analysis layers **49**
12.1 function and scope of derivation . . . 49
12.2 Derivation methods . . . 50
12.2.1 Grouping by functional proximity . . . 51
12.2.2 Grouping Safety Application Condition (SAC)s by criticality and hazard scope 52
12.2.3 Grouping by functionality . . . 54
12.2.4 Grouping by criticality . . . 54
12.2.5 SACs in lower levels . . . 55
12.2.6 Grouping by location in the system software architecture . . . 56
12.3 Conclusions from this first mapping analysis . . . 56
12.3.1 Is this SIL-decomposition in disguise . . . 57

13 Application of HD^3 **58**
13.1 Study Work-flow . . . 58
13.1.1 Study preparation . . . 59
13.1.2 Planing and schedule issues . . . 59
13.1.3 Conducting study sessions . . . 61
13.1.4 Managing databases and communication . . . 61
13.2 System work-flow integration . . . 62
13.2.1 Design recording . . . 62
13.2.2 Maintenance considerations . . . 63

A Guidewords **64**
A.1 Functional Guidewords . . . 64
A.2 Concurreny Guidewords . . . 64
A.3 Security Guidewords . . . 65

B (informative) Summary of procedures for HD^3 **66**
B.1 HD^3 subprocess specifications . . . 66
B.1.1 HD^3 Study precondition checks . . . 66
B.1.2 HD^3 Study planing checks . . . 67
B.1.3 HD^3 Study session specification . . . 67
B.1.4 HD^3 Item review specification . . . 68
B.1.5 HD^3 Study Consolidation specification . . . 68
B.1.6 HD^3 Study Item Modification . . . 68

C (informative) Summary of HD^3 record formats **70**
C.0.7 Session record . . . 70
C.0.8 Item record . . . 70
C.0.9 SAC record . . . 71
C.0.10 TODO record . . . 71
C.0.11 Selection record . . . 72
C.0.12 Study preperation record . . . 72

D (informative) Brief intro of design methods for HD^3 **74**
D.1 Finite State Machine (FSM) . . . 74
D.2 . . . 77

Acronyms **80**

Glossary **80**

ChangeLog

Version	Author	Date	Comment
0.1	Nicholas Mc Guire	June 10 2017	outline, overview notes
0.2	Nicholas Mc Guire	June 13 2017	scope
0.3	Nicholas Mc Guire	June 24 2017	process outline
0.4	Nicholas Mc Guire	July 5 2017	references and competence
0.5	Nicholas Mc Guire	July 9 2017	formal coverage of Annex-R
0.6	Nicholas Mc Guire	Dec 26 2017	Method details - draft
0.7	Nicholas Mc Guire	Jan 8 2018	Applying HD3
0.8	Nicholas Mc Guire	Feb 22 2018	Relationships completed
0.9	Nicholas Mc Guire	Mar 5 2018	Derivation process draft
0.10	Nicholas Mc Guire	Mar 8 2018	Initial design intent with review comments from Wolfgang Vogl
0.11	Nicholas Mc Guire	Mar 12 2018	spellchecking, references
0.12	Nicholas Mc Guire	April 8 2018	Method outline FSM/PSpec and integration into applying HD3
0.13	Nicholas Mc Guire	April 30 2018	Introduction

Reviews:

Version	Reviewer	Date	Comment

Authorization:

Name	Organisation	Date	Signature
Carsten Emde	OSADL		
Nicholas Mc Guire	OpenTech		

Status: Draft: incomplete, unreviewed, unauthorized

1 Introduction

There is a significant shift that has been on-going now for about 20 years. That shift is the loss of "componentization" - the engineering idea was basically: specify the functionality of interest and then aggregate a set of components that provide this functionality - done. The problem with this is that it never really worked - it was at best a close approximation and the deviation was simply accepted. For simple systems or aggregates of simple elements that model of design works tolerably bad. For complex systems it fails miserably. Before looking at why and then go into a possible mitigation methodology for system design - we need to specify what we mean with complexity. Even though every engineer has a metal image of complexity we have no common and possibly not even an adequate general definition.

1.1 Defining Complex safety related system

From a safety perspective we can look at complexity through the available definitions of low-complexity from IEC 61508 Ed 2:

```
3.4.3
low complexity E/E/PE safety-related system
E/E/PE safety-related system (see 3.2.13 and 3.4.1), in which
 -  the failure modes of each individual component are well
    defined;
 -  the behaviour of the system under fault conditions can
    be completely determined.
NOTE Behaviour of the system under fault conditions may
     be determined by analytical and/or test methods.
EXAMPLE A system comprising one or more limit switches, operating,
        possibly via interposing electro-mechanical relays, one or
        more contactors to de-energise an electric motor is a
        low-complexity E/E/PE safety-related system.

[IEC 61508-4 Ed 2 (2010)]
```

This hardly satisfies the system type that we are building today - moreover it exhibits the problem of staring at individual component while trying to grasp properties of systems. With this way of looking at systems even systems that are simply large aggregation of individually simple elements will miserably fail. This led an industry that was plagued by this problem to at least extend the definitions to cover complex systems (without defining complexity). IEC 62061 Ed 1 defines low complex component in Clause 3.2.7 in the same way as IEC 61508 Ed 2 and then adds:

```
3.2.8
complex component
component in which
 - the failure modes are not well-defined; or
 - the behaviour under fault conditions cannot be completely defined

[IEC 61062 Ed 1 (2005)]
```

The definition here is for software-intensive systems running on large processors with physical concurrency - with other words today's super-scalar multi-core hardware with non-deterministic optimizations.

```
complex safety related system
Any aggregation of components where some credible faults
depend on unintended interaction of functionally unrelated components
and where:
 - these failure modes are not well-defined; and
 - the behaviour under fault conditions cannot be derived
   from the component behavior.
```

1.2 complexity and context

There is a strong desire to have basic safety elements — almost system building blocks — that can be evaluated independently of the context - in its most quirky form dubbed Safety Element out-of Context (SEooC) [1]. This may be feasible for simple, low-complexity elements that have fully defined behavior - that is Type-A elements in IEC 61508-4 Ed 3 jargon — roughly 1) all failure modes known and 2) behavior under fault condition understood. This is not achievable for complex elements like a modern multi-core and security capable OS.

There is a tradeoff between contextual specification (effectively being a constraint on the relevant failure-modes) and manageable complexity. Even if we could exhaustively analyze a highly complex element to enumerate all of its failure modes, this would not be a reasonable strategy, if the actual subset in use is very small compared to what the element offers: 1) we would run into unmanageable re-evaluation needs on incident reports and 2) the integration of such complex elements into a system still would stay unmanageable due to the inability to enumerate/analyze all interactions. Safety management is always — maybe even to a large part — a matter of putting the resources available to the best use possible. An "out-of-context" approach though is not minimal — for a low-complexity element it may though still be the simpler way to go as the effort is offseted by the re-use of the element — for complex elements this is not the case, most likely not even long term.

Qualitatively the effort grows exponentially with the elements' complexity (even with a single-fault-hypothesis this should hold — if CCFs are to be taken into account then exponential growth

[1] The fact that something like SEooC (and a few other issues in ISO 26262) made it into a functional safety standard does constitute a systematic failure of the standards development and review process

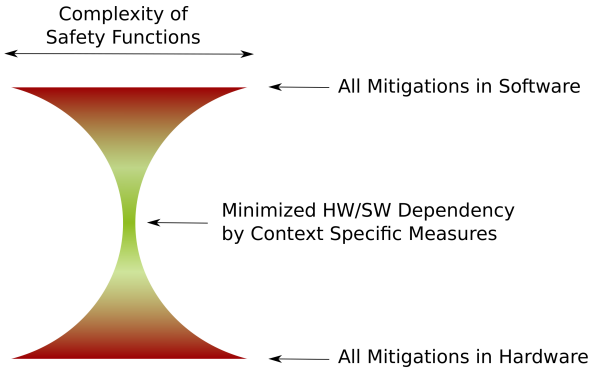


Figure 1: complexity growth dependency on context awareness

is sure) — the only defense, if complex elements are to be used, is to constrain them by maximizing system context awareness in all phases of development most notably for any analytical step.

For highly-complex elements being used in safety related systems our hypothesis is that a management "out-of-context" [2] becomes unmanageable for complex systems and the global mitigation is to re-establish manageability by maximizing the context by a structured, fully contextualized, analysis.

1.3 Managing complexity by context

What we proposed could be called "Safety Element with maximized Context" — not a nicely pronounceable acronym — who would want to use "SEwmC"! The challenge of utilizing complex software elements in safety related systems is regaining control of the complexity by focussing on analytical procedures — testing is close to useless for complex systems with respect to assurance of correctness!

Complexity management has a few aspects that need to be differentiated

- Establishing human manageable system understanding
- Ability to detect faults going beyond analysis scope
- Ensuring that the properties are retained under expected change

The first is the goal of IEC 61508 Ed2 part 1 early system life-cycle phases (Scope, Concept, Hazard Analysis, System safety requirements specification to some level also Allocation to technologies) The second issue is not directly addressed in IEC 61508 Ed2 but can be inferred from the way verification is outlined as well as the completeness attributes are called for in Annex-C — an analysis that did not actually cover the scope of the system quite clearly is of limited use and would not allow achieving a justifiable level of assurance. Finally — and this is maybe an underestimated issue as studies like HES "Out of Control" [**?**] show — maintaining safety in a system that is undergoing significant changes over time (in system change and/or the environment going out-of-analysis scope) must be considered;- we will refer to this as "dynamics of systems".

[2] Provided that this ever is a reasonable approach for achieving safety

To satisfy these high-level safety goals the SIL2LinuxMP project is striving to achieve when utilizing GNU/Linux in the context of safety related systems, requires to employ the partially new method outlined in this manual, since we believe that it is suitable to achieve the goal of building a safe system utilizing complex pre-existing software elements. While this method was developed as mitigation of process level deficiencies uncovered during the SIL2LinuxMP project, the method itself actually neither is specific to SIL2LinuxMP nor to using Open Source software in general.

At the same time analysis of past accidents [5] (see table 1) show that the faults are not so much at the level of the components implementation (e.g. coding), but rather high-level faults and maintenance issues seem to dominate. With other words, they happened in those phases where the complexity of the system constitutes the main challenge.

Primary cause by phase	Frequency	Phase %
Inadequate functional req. spec	4	12
Inadequate SIL req. specification	11	32
Total inadequate specification	15	**44**
Total inad. design and impl.	5	**15**
Total inad. install./comm.	2	**6**
Inadequate operation	1	3
Inadequate maintenance	4	12
Total inadequate op/maint.	5	**15**
Inadequate modification	7	20
Inadequate de-commissioning	0	0
Total inad. change after comm	7	**20**

Table 1: HSE, UK, *HSG 238 - Out of control - Why control systems go wrong and how to prevent failure*, 2003

While we are able to manage/design low-complexity components quite well and to some extent complex components, we have little experience to manage complex systems. With managing we mean not only designing and building them but maintaining them with respect to the desired properties while not exceeding the societally tolerable undesired properties. The challenge, thus, is to create a safe system level design while permitting high dynamics of the aggregated elements.

The design methodology proposed to achieve this goal is simply to derive systems from the analysis so that any design decision actually still can be assured to have a complexity that we can manage analytically — testing is nice but not of much use to give any guarantees for any of the corner cases of a complex system [3]. This in essence is the idea behind "Hazard driven Decomposition, Design and Development" (acronym "HD^3") — with the side effect that the result of such an analysis may well be "Do not build it!".

[3] Which I would claim constitute the majority of operational situations for sufficiently complex systems

2 Scope

This document describes Hazard driven Decomposition, Design and Development (HD^3), and provides guidance for applying HD^3 to achieve its main objectives. This document covers the aspects of:

- motivating HD^3 by the overall analytical flow of IEC 61508 Ed 2 and derived standards;
- introduces basic principles;
- providing the procedural specification as a basis to perform a hazard driven system decomposition, design and development;
- defining, where necessary, terms, assumptions either by explicit inclusion or by reference (see normative references);
- defining the basic principles;
- describing the rational for the proposal of this new method;
- providing examples of applying HD^3 to a mid-complexity software intensive system.
- providing an example of integrating HD^3 in the overall safety planing to mitigate shortcomings in pre-existing elements.

Hazard driven decomposition, design and development is an extension of the well established HAZOP method switching the design and analysis phase, while we address issues related to HAZOP in this document we will **not** give an introduction to HAZOP it self, for this we refer you to [4], [6].

2.1 Note on reading

This document introduces the HD^3 method at the technical level as well as giving an overview with respect to sources of assurance and its role in the safety life-cycle. The later is quite formal and might not be of much interest to someone looking for an overview of the method.

Method intro and overview:

2 Scope
4 Terms and definitions
5 Overview
9 Hazard driven decomposition, design and development
10 Design intent as starting point
11 Item analysis
12 Derived analysis layers
13 Application of HD^3

Formal aspects (qualification):

2 Scope
3 Normative references
4 Terms and definitions
5 Overview
6 HD^3 Process requirements
7 Annex R coverage
8 Related considerations
9 Hazard driven decomposition, design and development

Reading structure

3 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.

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 March 2018 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*[4], Edition 2, *IEC*, 2010
- IEC 61882 *Hazard and operability studies (HAZOP studies) — Application guide*, Edition 1, *IEC*, 2001
- IEC 60300-3-1:2003 (German edition 2004), *Dependability management - Part 3-1: Application guide - Analysis techniques for dependability - Guide on methodology*, Edition 2, *IEC*, 2003

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.

[4] part 0 is available for IEC 61508 Ed1

4 Terms and definitions

For generic terms and definitions see Appendix D.2, here we only list those that are specific to this manual or are used in a way that invites confusion.

4.1 Item

The term Item is in use widely with many different meanings. For the use of the work item in this document it refers to HAZOP items, that is a unit of analysis. Notably with functional safety standards that use item in a completely different meaning it is advisable make prominent notice of any use of item in relation to HD^3 that does **not** refer to a 'unit of analysis' in the context of a HAZOP process.

4.2 Mid Complexity

Some systems are not easily classified as Type-A (low complexity) or Type-B (high complexity [5]) systems, but rather they carry properties of both. Low complexity is defined as

> *low complexity Electric/Electronic/Programmable Electronic (E/E/PE) safety-related system E/E/PE safety-related system (see 3.2.13 and 3.4.1), in which*
>
> - *the failure modes of each individual component are well defined;*
> - *the behavior of the system under fault conditions can be completely determined.*
>
> [IEC 61508-4 Ed 2 3.4.3]

Systems where the intended function of the system can be classified as low-complexity but some of the supporting elements can not (e.g.) we will refer to these as mid-complexity systems (or mixed complexity).

4.3 Derived item

Basically re-using the definition from DO 178 B

[5] Note that high-complexity is normatively not defined in IEC 61508 Ed 2 but rather only inferred to not satisfy the assumptions of low-complexity so technically there is an overlap of our mid-complexity definition with IEC 61508 Ed 2 part 4

> *"The DO-178C/DO-278A glossary defines derived requirements as: "Requirements produced by the software development processes which (a) are not directly traceable to higher level requirements, and/or (b) specify behavior beyond that specified by the system requirements or the higher level software requirements."*
>
> [DO 248 derived requirements]

The traceability is indirect via the analysis hierarchy emitting which are technically not present in the top-level design intent (it may be that the top-level design intent does not even note the term safety explicitly).

5 Overview

IEC 61508 and its derived standards build on the premise that safety is a system property and thus system safety is the prime focus. While there are many reasons to strive for component based safety [6] we consider the treatment of elements in isolation as conceptually broken for complex systems.

A number of derived standards address low-complexity systems and reference back to IEC 61508 for high-complexity elements. As an example, representative for the principle approach, we cite IEC 62061 treatment of complex components:

> *The subsystem shall be realized by either selection (see 6.7.3) or design (see 6.7.4) in accordance with its safety requirements specification (see 6.6.2.1.7), taking into account all the requirements of 6.2. Subsystem(s) incorporating complex components shall comply with IEC 61508-2 and IEC 61508-3 as appropriate for the required .*
>
> [IEC 62061 6.7.2.1]

> *Where the design of a subsystem incorporates a complex component (as a subsystem element) which satisfies all relevant requirements of IEC 61508-2 and IEC 61508-3 in relation to the , it can be considered as a low complexity component in the context of a subsystem design since its relevant failure modes, behavior on detection of a fault, rate of failure, and other safety-related information are known. Such components shall only be used in accordance with its specification and the relevant information for use provided by its supplier.*
>
> [IEC 62061 6.7.4.2.3]

The key point being that the overall set of failure modes and behavior under fault-condition of the complex element is **not** assumed to be exhaustively known but only the relevant failure modes are assumed to be known.

5.1 Motivation for HD^3

The intent in IEC 61508 is expressed clearly, what we found not so clearly expressed is the measures and techniques to be employed to achieve this goal in a satisfactory manner with respect to adequate completeness and consistency.

The intent of HD^3 is to provide a systematic approach to developing a complex system from the initiating functional design intent [7] commensurated with the risk potential, through decomposition led by the principle of

[6] sometimes referred to as Safety Engineer out of Context (SEooC)

[7] This is the basic functionality of the system which is apparent without any design documentation - that is the "obvious functionality", e.g. "Measure the number of E.Coli bacteria in water."

- Minimize the safety related role of software [8]
- Risk removal before risk elimination

to achieve the overall intent of a safe system design for complex systems.

To achieve this, an exploratory approach is needed - for this we employ the well known HAZOP method at each of the levels of decomposition - and a process that will allow to retain the hierarchical relation and dependencies of the system deign layers in a traceable (and thus maintainable) manner.

Hazard driven decomposition, design and development strives to evolve a system from its initial design intent through a guided process to a safe system which is ultimately the aggregation of:

1. Directly intended functionality (system design intent)
2. Risk elimination, reduction and detection needs
3. Generic elements providing unspecific base-services and functionality in the context of 1 and 2.

For low complexity systems (type-A systems) it may be reasonable to focus on functionality of software and hardware initially and then add in safety considerations layer-by-layer as an iterative step. Requirements -> requirements review -> update, etc. The problem with this is twofold:

- This hardly leads to minimality
- The overall complexity of generic safe-properties does not scale to high complexity

The two concerns are related - figuratively this can be expressed as:

FIGURE: context sensitivity of effort complexity$_o f_s a f e t y_f u n c t i o n s . s v g$

Thus we assume that the only way to achieve minimality and management of system complexity is to drive the overall life-cycle from a strict hazard perspective.

5.2 Process overview

In traditional system design we are driven by functionality. The design intent of the new system is postulated and is then decomposed, at a functional level, into modules with well-defined interfaces. The effect of this is that any hazard source may follow complex and not necessarily intuitively related, paths through the overall system. Making such hazard related coupling of elements explicit in complex systems is hard. The reasons for this difficulty lies in contextual sensitivity of hazards and in the inherent difference of hazard context and functional context.

The next step in safety related systems is to analyze a design in the context of safety demands. The classical HAZOP method utilized the design descriptions, requirements documents, etc. as input to the HAZOP intending to explore all possible hazard sources present. While this has shown to be effective it is fundamentally not slanted towards the goal of inherent safe design:

[8] see IEC 61508-3 Ed 2 7.4.2.6

Functional safety is just one method of dealing with hazards, and other means for their elimination or reduction, such as inherent safety through design, are of primary importance.

[IEC 61508-0 **Ed 1** 3.2]

To mitigate these issues HD^3 employs the HAZOP method as the means of evolving the system from the initial design intent down to the dependent functions. The input is **not** requirements specification and/or preliminary design documentation, but rather a, often incomplete and naive 'understanding' of the intent for building this new system only. From this top level design intent all elements are emitted **as needed to minimize the risk** associated with providing the intended functionality.

HD^3 is a layered approach that focusses on the decomposition of hazards in the sense that each detected hazard has a trace through the hierarchical decomposition of the system allowing to perform impact analysis at the granularity of a perceived hazard through the set of known contributing, moderating and mitigating elements.

HD^3s layering overview is as follows

Input: Top-level system intent (design intent)

Flow:

Technology agnostic analysis

- Derived technology agnostic functions
- System safety application conditions

Unspecific technology aware analysis

- Derived technology aware functions and services
- Functional safety application conditions

Specific technology aware analysis

- Derived technology aware specific functions
- Dependent technology aware specific helper/wrappers
- Specific safety application conditions

The intent of this phase model is to address the system as a whole at all levels by providing a linked context. The analysis at the detailed level is in a context that can be followed 'back-up' at all steps. This iteration is a necessity not only to stay in system context, but also to review the higher levels of decomposition in the lower-level specific context (and amend it where needed). The flow shown might suggest a waterfall like development life-cycle, it should be clear though that this is an iterative process with the intent to reach the bottom only after the overall hierarchy has achieved **consistency and completeness by systematic iteration** - see IEC 61508-3 Ed 2 .1.1.4 [9]

[9] For those that do not yet have a favorite clause in IEC 61508 Ed 2 we strongly recommend 7.1.1.5 in part 3

5.3 Generated Artefacts

The overall goal of HD^3 is the generation of a system design that addresses all relevant uncovered hazards in a way that continuous assurance of the overall system safety is feasible. The guiding statement for the design of the HD^3 procedure is:

> *The fourth objective of the requirements of this subclause is to design and implement software that fulfils the specified requirements for safety-related software with respect to the required safety integrity level, which is* ***analyzable and verifiable****, and which is* ***capable of being safely modified****.*
>
> [IEC 61508-3 Ed 2 7.4.1.4]

In the context of a high-complexity element being integrated into a system, for which its **relevant** failure modes, behavior on detection of a fault (see note on IEC 62061 above) shall be understandable over the life-time of the modifications to the system, the analysis needs to focus on the path of hazard derivation and resolution, or maintainability will necessarily break during maintenance. It is not feasible to regenerate system context each and every time a system containing complex elements is modified. This is ultimately the problem HD^3 is striving to address.

To do so the HD^3 process generates hierarchically dependent and explicitly coupled artefacts at each level of decomposition. Note, that discovery of artefacts is not a waterfall but iterative by the processes nature. Using the procedural layering from above, the following categories of results from the HD^3 process are attained:

1. Technology agnostic analysis
   - Concept validation
     - Safety application conditions
   - Derived concepts/concept extensions
     - Derived safety application conditions
2. Unspecific technology aware analysis
   - Technology selection validation
     - Safety application conditions
   - Derived technology/technology adjustments
     - Derived safety application conditions
3. Specific technology aware analysis
   - Element selection criteria
     - Safety application conditions
   - Derived elements/element specifications

  - Derived safety application conditions
- Dependent elements/element specification
  - Derived dependent safety application conditions

From the overall structure and the relation to the process, under consideration of the iteration needs the application of HD^3 effectively mandates some level of automation. This automation is needed to manage the dependency graphs as well as for the traceability of the artefact evolution.

5.4 Roles and Competence

Given the system level of the analysis and the relatively high dependency between the artefacts emitted from the different layers of decomposition, it is crucial to ensure that the involved actors in the HD^3 process addressing each of the relevant roles in the process, have the necessary competence at their disposal.

As with all system analysis methods, it is neither expected, nor actually desirable, to have a single actor dominate the process. Essentially these issues are the same as for HAZOP sessions which is why we will not repeat the roles and competence needs here in detail (see [6] for a fast-track introduction). The key roles in HD^3 and their core competence:

ID	Role	Competence
0	Study Lead	Authority and disposable resources, knowledge of legal and regulatory framework
1	Recorder	Safety Management, Context understanding, authority to question
2	CM	Tools competence to manage operational tools and related traceability needs, authority to reject contributions
3	Design Manager	System Safety, Design methodology, Safety life-cycle properties
4	Domain Experts(s)	Domain and system environment understanding, knowledge of the available solution space
5	Implementation Engineer(s)	Technology capability understanding notably complexity impact

Table 2: Roles and Competence for the initial HD^3

Note that the life-time of HD^3 is not just the development of the system but essentially it would continue until the decommissioning of the system. The roles listed here now focus on the initial requirements, design and development phases, for maintenance/modification/retrofitting the competence and know-how may change (e.g. knowledge of operational history)

Further note that joining roles may or may not be adequate for a particular project at a given integrity. Joining of roles shall be justified. Depending on the level of independence needed joining the recorder and CM may be adequate - while joining the study-lead and the design manager will most likely always be problematic. The goal of independent roles is to have an umbrella protection against high-level organizational faults like pushing changes due to time-pressure, or to allow pinpointing discrepancies between regulatory needs and actual processes. Resolution of such conflicts may well

be doable in a specific situation **provided it has been detected, documented and resolved with a reviewable justification**—if joining roles prevents such high-level faults from being discovered then these roles should **not** be joined.

6 HD^3 process requirements

Largely these are at the functional safety management and Configuration management level as well as documentation level see IEC 61508-1 Ed 2 Clause 5 and 6 as well as relevant extensions in IEC 61508-2 Ed 2 and IEC 61508-3 Ed 2.

Adequate safety management and configuration management is assumed and not addressed here.

- HD3.1 Roles shall be specified and assigned to specific individuals
- HD3.2 Joining of roles shall be justified
- HD3.3 Table entries shall be consensual within the study team or dis-consent recorded
- HD3.3.1 Unresolved issues shall be assigned as managed TODOs to a specific team member
- HD3.3.2 Resolution of TODOs shall be reviewed by other team-member(s)
- HD3.4 Results shall be versioned so as to ensure that HD^3 results are associated with a defined state of system under study
- HD3.4.1 Item results shall follow a common format
- HD3.4.2 Subdivision of Item properties shall be enumerated
- HD3.4.3 Any mitigations shall be recorded as managed SACs
- HD3.4.4 Mitigations by previously determined SACs shall be recorded by number and a short description (to ensure readability without lookup)
- HD3.5 A minimum level of automation is mandatory to achieve the properties claimed in Table 4
- HD3.6 Attendance to sessions shall be documented in a separate table. (see 8.6 and **??**
- HD3.7 Criticality of mitigations should be allocate at the qualitative level
- HD3.7.1 False positive of functions and false-negatives of inhibition/diagnostics are to be classified as high.
- HD3.7.2 False negatives of functions and false-positives of inhibitions/diagnostics are to be classified as mid provided it is assured that false-positive inhibition/diagnostics will not result in alarm showers.
- HD3.7.3 maintenance impact of functions and diagnostics (e.g. non-recording of trending data) shall be classified as low.
- HD3.7.4 classification that does not follow HD3.7.1,7.2 and 7.3 should be documented.
- TODO: probably not yet complete.

The HD^3 specific requirements are to be considered during the verification of process steps utilizing artefacts of the HD^3 process. The HD^3 specifications need to be taken into consideration for any support tools.

7 Formal coverage of Annex-R

This section contains the formal coverage of the proposed Annex-R addressing the introduction of new tools in the context of a system safety life-cycle. For details see the Annex-QR proposal [10].

- R.2.1:
  Properties are as given in IEC 61508-3 Ed 2 Table C.1, the primary focus is on the coverage of intrinsic design faults of elements (dependent

- R.2.2:
  HD^3 minimum specification is as follows:
  - **Aim:** Provide a system-context methodology for generating a complete and consistent safety-related system design.
  - **Description:**
    HD^3 drives the system design from the initial system intent through a decomposition process guided by exploratory hazard and operations analysis at all levels. The intent is to ensure that the potentials for hazard elimination and hazard severity reduction and/or hazard probability reduction is fully exploited before hazard mitigation is sought. HD^3 has two major element to it
    1. Use of exploratory analysis iteratively applied to each design layer with the starting point of the development process being the "design free" initial system intent.
    2. Continuous recording of context, dependencies (up/down) and derivatives (SACs, selection criteria, derived requirements) ensuring that the hierarchy of the system stays analyzable/verifiable under modification.

  HD^3 is intended for the software safety requirements specification as well as to address software design and development - detailed design at recommendation level R (recommended).

- R.2.3:
  While HD^3 is not yet consolidated at this point the main rational for introduction is the need to keep complex software centric systems maintainable in the context of addressed system hazards. This is seen as critical due to the high rate-of-change introduced in connected systems due to security impacts (see IEC 61508-1 Ed 2 7.4.2.3). If change management were to be driven through a functional decomposition this would mandate system-level re-evaluation of each and every hazard, hazard reduction and mitigation, which we consider unrealistic. With systems including type-B (high-complexity) elements [11] a system design focussed on analysis and re-verification (see IEC 61508-3 Ed 2 7.4.1.4) to ensure safe modifiability.

- R.2.4:
  There is no known conflict with HD^3 (there is in fact a gardening tool named HD^3 !) located with google search .

[10] Note that Annex-QR is **not** a part of IEC 61508 Ed2 but has undergon a reveiw by TueV Rheinland
[11] As IEC 61508-4 Ed 2 does not formally define high-complexity - we refer you to IEC 62061 Ed 1 3.2.8

- R.2.5:
  See Section 9.1
- R.2.6:
  Not yet addressed - both assessor and mandatory competence are not yet determined.
- R.2.7 and 8:
  - Forward and backward traceability as well as semi-formal methods are dependent methods thus need to be selected in coordination with the HD^3 process.
  - Competency see Table 2
  - Integration into Tables of Annex A/B/C - see Table 3 and 4 below

	Technique/Measure	Ref.	SIL1	SIL2	SIL3	SIL4
1a	Semi-formal methods	Table B.7	R	R	HR	HR
1b	formal methods	B.2.2, C.2.4	–	R	R	HR
2	Forward traceability between the system safety requirements and the software safety requirements	C.2.11	R	R	HR	HR
3	Backward traceability between the safety requirements and the perceived safety needs	C.2.11	R	R	HR	HR
4	Computer-aided specification tools to support appropriate techniques/measures above	B.2.4	R	R	HR	HR
5	Hazard driven decomposition, design and development		R	R	NC	NC

Table 3: IEC 61508-3 Ed 2 Table A.1 - Modified

[12] [13]

Note: The properties claimed here are not reasonably achievable without a minimum level of automation, The properties rely on correctness and completeness of traceability which is not achievable by manual process.

Justifying R2:

- Understandability: of safety requirements that are traceable to the hazard analysis in a defined context allow for R2 with the objective acceptance criteria being completeness of all items related to the safety requirement at hand
- Basis for Verification: As HD^3 is effectively a recoding of the overall analytical context at system level the criteria for R2 claims is completeness of study (Study release see Section 9.5 item 10)

[12] R: recommended, HR: highly-recommended, NC: not-considered, –: no prference
[13] rigor R1, R2, R3 as defined in IEC 61508-3 Ed2 C.1.2

	Completeness with respect to the safety needs to be addressed by software	Correctness with respect to the safety needs to be addressed by software	Freedom from intrinsic specification faults, including freedom from ambiguity	Understandability of safety requirements	Freedom from adverse interference of non-safety functions with the safety needs to be addressed by software	Capability of providing a basis for verification and validation
1a	R1	R1/2	R1/2/3	R1/2	-	R2
1b	R1	R1/2/3	R1/2/3	-	-	R3
2	R1	-	-	-	-	-
3	-	R1	-	R1	R1	R1
4	R1/2	R1/2	R2	R1	R1	R1/2
5	R1	R1	R1	R1/2	R1	R1/2

Table 4: IEC 61508-3 Ed 2 Table C.1 modified - contribution by HD^3

- R.2.9:
  To be assessed in the context of the specific method/project - not generically addressable.
- R.2.10:
  It is currently assumed that HD^3 could provide the property coverage outlined in Table 4 at system level, provided that all elements interacting within the system (safe or non-safe) have been covered by the HD^3 process.
- R.2.11:
  To be covered by safety management in the context of the specific project/team.
- R.2.12:
  To be covered by the safety life-cycle management and documentation process in the context of the specific project/organization.

8 Related considerations

While HD^3 is a methodology that is not necessarily bound to other measures or techniques it is related to other safety activities and requirements inherent to any design method (traceability) and any automation efforts (tools qualification).

Most measures/techniques also have an effect on safety life-cycle details and thus on safety-life-cycle management. In the context of pre-existing software elements, which sometimes will be the complex elements building up the bottom-half safetys functions, the IEC 61508 Ed 2 part 1 extension used in SIL2LinuxMP will exhibit some level of dependency on the system decomposition and design addressed by HD^3.

Not all these sections will apply to all systems, as HD^3 is under development in the context of SIL2LinuxMP, thus we consider those issues here the relate to the use of pre-existing elements in the context of Route 3_S.

This section is specific to SIL2LinuxMP to some extent - in so far as it documents the specific role of HD^3 in the context of IEC 61508 Ed2 and the SIL2LinuxMP specific extensions covering selection and Annex-QR as well as the specific context of the current (Coliminder) Use-case.

HD^3 is in development and it takes some failures and derived experience to consolidate and generalize the method. Even if the method targets a general applicability this section should not be considered without taking the specific SIL2LinuxMP context into account.

8.1 Safety life-cycle Management

HD^3 focusses on the early life-cycle phases as outlined in IEC 61508 Ed 2-1. The Goal of HD^3 with respect to preparing the Selection is to identify a minimum subset, of the analyzed specific technology candidate, that is need to achieve the functional safety requirements. While the formal input to HD^3 is a minimum design intent, it is assumed that the development of this design intent to a specific set of functionality along with the related functional safety requirements is guided by adequate domain know-how in the team—it is **not** sufficient to have functional understanding only.

We note that the detailed mapping of IEC 61508 Ed 2 -1 7.2-7.6 + 7.X is covered in the Acceptance Test Criteria (ATC)_HD3 here we only provide the mapping to the objectives of the respective sections of IEC 61508 Ed 2 part1. Specific contribution of HD^3 to objectives of the life-cycle phases of IEC 61508 Ed 2-1 are:

Concept : Conceptual uncertainties are either removed during HD^3 or recorded as (traced) TODOs. The only limitation of HD^3 with respect to the intent of IEC 61508 Ed 2 -1 7.2.1 is that legal aspects are **not** formally covered. Legislation may be an input to the process but probably can be addressed in a dedicated review so as to limit the analytical complexity.

Overall scope definition : Determination of the system boundaries as called for in IEC 61508 Ed 2 - 1 7.3.1.1 is addressed by the scope of the guidewords as well as by the separated treatment of concurrency related effects (implicit coupling). Developing the understanding of the physical elements as called for in IEC 61508 Ed 2 -1 7.3.1.2 is ensured as the element selection takes place in the scope of HD^3 (notably in the transition from technology agnostic to technology aware analysis.

Hazard and risk analysis : Determination of *the hazards,hazardous events and hazardous situations relating to the EUC and the EUC control system* as noted in IEC 61508 Ed 2 -1 7.4.1.1 is fully addressed by HD^3, including reasonably foreseeable circumstances and foreseeable fault conditions, by the explorative nature of the hierarchical HAZOP. The issues of foreseeable misuse is addressed by inclusion of security related guidewords if IEC 61508 Ed 2 -1 7.4.2.3 indicates the need of a threat analysis to be conducted. [7]. The intent expressed in IEC 61508 Ed 2 -1 7.4.1.2 (event sequence) is partially addressed by HD^3 the full extent of event sequences is deferred to later safety life-cycle phase due to the need of knowing the architectural constraints and capabilities. risks (interpreted as severity and probability) as called for in IEC 61508 Ed 2 -1 7.4.1.3 is partially addressed as severity is (qualitatively) recorded but probabilities can not in general be determined until the software architectural capabilities (e.g. SW-LOPA) are fully explore.

Overall safety requirements : As stated in the objectives for the overall safety requirements in IEC 61508 Ed 2 -1 7.5.1 the attributes are the safety functional requirement and the safety integrity requirements. HD^3 addresses the safety functional requirements (most often by decomposition of SACs) but in general will not allow direct allocation of safety integrity requirements. This is only possible after the reliance placed on individual functions is visible from the completed initial HD^3 study where the protection functions can be tallied as a first approximation of their criticality at the system level.

Overall safety requirements allocation : The allocation to the system as stated in IEC 61508 Ed 2 -1 7.6.1 is interpreted as allocation to the respective system element. As the system elements are actually derived during HD^3 this can be provided by the method with the noted limitation on the integrity level. With respect to target failure measures , clearly HD^3 is providing this allocate as call for in IEC 61508 Ed 2 -1 7.6.2 by the systematic derivation of SACs starting at the technology agnostic functional specifications.

Selection : The proposed extension to IEC 61508 Ed2 part 1 7.X.1.1 covers the compliance of selected elements to the safety functional and safety integrity requirements. Again the functional compliance can be extracted from HD^3 directly and base-information for the integrity levels is recorded along the way. The final allocation of safety integrity levels and thus judgement of compliance is not possible before the software architectural constraints have been developed. 7.X.1.2 addresses the coherence of the selected elements respectively the selected safety measures. HD^3 allows judgement of the functional coherence through the hierarchical process of decomposition that is driven by abstract properties and not by specifications of elements in the early phases. Mapping to specific elements and their constraints/limitations occurs in the context of identified hazards and thus, as long as all considered elements satisfy the hazard elimination/mitigation/detection needs - this consistency is an inherent result of HD^3.

8.2 Relation to 7.X 'Selection'

Layering of HD^3 and with that selection during the transition to lower layers, will most likely depend on the global properties of the system under analysis - notably size and complexity. If the current layering scheme will hold is not yet known, the principle methodology though seems robust enough to allow additional iterations into derived layers if the analysis finds this to be necessary.

This section might need to be revised to extract the generic core of the HD^3 method once found stable.

Selection has two basic aspects:

- Element candidates that can satisfy the requirements
- Suitable measures and techniques to provide adequate assurance [14]

These two aspects are further developed below as far as they relate to HD^3.

Selection is directly addressed in the context of HD^3 analysis by extracting minimum criteria for elements that could potentially satisfy the demands of the requested functionality in the context of the uncovered hazards (recorded in the "Selection" table).

The objectives addressed by HD^3 cover compliance and coherence of possible elements during the transition from the technology aware unspecific analysis (e.g. at this point it is known that an OS on a multi-core system is in use) to the technology aware specific analysis (at which point it is clarified that it is a NetBSD on a Octo-core Alpha). The primary contribution that HD^3 provides here is coverage of properties that are in principle **not** covered in a pre-existing software element - notably intrinsic property faults.

A concurrent process not visible in HD^3 is the investigation of potential pre-existing elements and/or options for bespoke elements, to satisfy the demands that manifest them selves during analysis outlined in figure 2 in the right column. Note that the middle column is not addressed by HD^3 but is a concurrent compliance route definition/developmne process.

The development of the element potentials allows for trade-off discussions to take place with respect to best-fit pre-existing elements at the transition from the technology agnostic to the technology aware analysis. This is **not** part of the HD^3 process it self. As an example one may decide to satisfy the technology agnostic demands by a micro-controller without an OS or by a multi-core utilizing multiple OS and virtualization - this decision is **not** part of the HD^3 process never the less it impacts the analysis substantially what system architecture decision is taken.

Any selection decision would though need to conform to the SACs emitted by HD^3 (ultimately at all levels either by direct conformance or by additional mitigations introduced for that purpose - these would show up at the lower technology aware levels during analysis).

The second point were a similar issue arises is during the transition from technology aware unspecific to technology aware specific analysis. At this point the trade-off decision between the different element candidates is taken and is encoded in the selection criteria as well as in a formal decision to e.g. select GNU's Not UNIX! (GNU)/Linux with kernel v4.9.66. Such decisions are directly recorded as TODOs in the table items as well as in the HD^3 document and in most cases will reference external documents or only a summary rational for a selection.

The actual elements then considered during selection is further refined into the system software architecture along the way "down" in the increasing levels of analytical detail as expressed in figure 3. Note that architectural aspects are not unidirectional in the case of pre-existint elements but rather the analysis increasingly takes the detailed properties of slected elements into account [15].

[14] Arguably this implies a third basic aspect of competence - but we leave that as a separate issue of safety management and will not explicitly include it here

[15] Note that we are assuming an approach predominantly using Route 3_S "assessment of non-complient development see IEC 61508-3 Ed2 7.4.2.13.

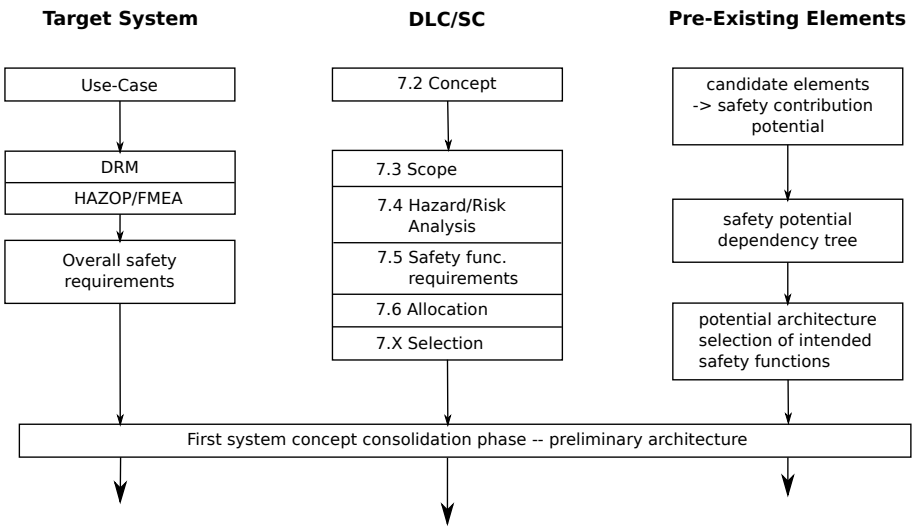


Figure 2: Safety life-cycle position of HD^3 - element selection aspets

Selection of suitable measures and techniques from 61508-3 Ed2 Annex A and Annex B will in general not be sufficient in the case of pre-existing software elements as some attributes inherently can not be covered by selection of the overall element unless one would actually analyze the element to the same level of detail that would be conducted during a bespoke process. Even if this were possible it would not help much as the element would not be generally modifiable (or it would no longer be a pre-existing software element [16] Thus the strategic option enabled by HD^3 is to extract the minimum subset of relevant functionality (documentation, tools and code-base) and so allow an extensive analysis of this **Use-case specific** and therefor fully contextualized logical sub-elements. This provides the basis to achieve reasonable completeness of understanding and in consequence open the possibility of completeness of analysis (modes of operations (61508-3 Ed2 7.2.2.6), input for validation planing (61508-3 Ed2 7.3), detection of safety relevant hardware constraints (61508-3 Ed2 7.2.2.7), constraints on software architecture (61508-3 Ed2 7.4.3), crucial properties to be addressed in validation planing (61508-3 Ed2 7.4.6) and element (module) testing (61508-3 Ed2 7.4.7). For a complex element this completeness, and most notably, consistency can not be achieved if the element is simply considered as a monolithic whole — equally it can not be simply subdivided arbitrarily (e.g. by abstract functional relation) as this would not allow analysis of the relevant interaction. HD^3 thus is the method used to extract the relevant functional subset, including its dependencies and interactions so as to make the analytical scope manageable (see also 61508-3 Ed2 Annex C.5).

In this minimum subset context, the addressing of intrinsic requirements design and implementation issues is feasible. It is explicitly noted that this is **not** creating a out-of-context analysis in any way - in fact the assumption is that such an out-of-context or context-free analysis of even mildly complex software elements is technically infeasible.

The specific properties being addressed in IEC 61508-3 Ed2 Annex C that we consider uncovered

[16] Minor modification e.g. at the code-level can. in principle, be justified but no change at the requirements or design level

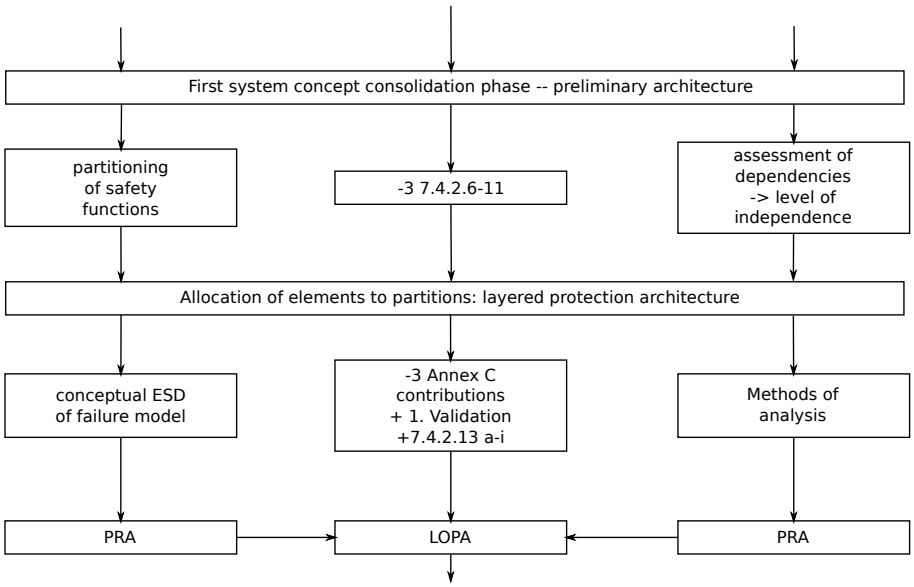


Figure 3: Safety life-cycle impact of HD^3 - architectural aspects

by application of measures and techniques from Annex A and B in the case of pre-existing elements are:

The treatment of intrinsic faults of a pre-existing element mandates a level of analysis that ensures, with reasonable probability, that these weaknesses will be uncovered. To actually address these weaknesses it may be necessary to

- Constraint the selection;
- Provide wrappers that actively mitigate the issue; or
- Implement a monitoring element to respond to deviation from expectation [17]

These mitigations would then show up in the HD^3 analysis in the respective mitigations (SACs). It will commonly be the case that a mitigation turns out to be relevant for multiple deviations from design intent. This is recorded in HD^3 by the reference count of the respective SAC and thus allows a first assignment of criticality to a mitigation measure. It should be noted that this is though not a direct quantification but only a qualitative indication of mitigation criticality.

8.3 HD^3 and IEC 61508-3 Ed2 Annex C properties

Annex QR covers the systematic tailoring of measures and techniques as well as the introduction of new measures and techniques. In general it seems that IEC 61508 Ed2 focussed all its measures and techniques on the bespoke development path (Route 1_S). The problem for a development concerned primarily with an "Assessment of non-compliant development" (Route 3_S) is that some of the tables in

[17] Note that this is not necessarily the same as deviation from specification in the case of a pre-existing element

C.1	Freedom from intrinsic specification faults, including freedom from ambiguity
C.2	Freedom from intrinsic design faults, Simplicity and understandability
C.4	Freedom from intrinsic design faults, Simplicity and understandability
C.5	Precisely defined testing configuration
C.6	Repeatability
C.6	Precisely defined integration configuration
C.7	Repeatability
C.7	Precisely defined validation configuration
C.8	Freedom from introduction of intrinsic design faults (Note: we would prefere the term activation of intrinsic design faults here)
C.8	Avoidance of unwanted behaviour (Note: with respect to unwanted behavior rsulting from the interaction of aggregated elements)
C.10	The ability to modify the functional safety assessment after change without the need for extensive re-work of the assessment (Note that we expect many complex pre-existing elements to have high change rates)

Table 5: Contributions of HD^3 to phase properties (IEC 61508-3 Annex-C)

IEC 61508-3 Ed2 Annex C no longer faithfully reflect the capabilities of the measures and techniques recommended.

For pre-existing elements these shortcomings must be mitigated in a systematic way and one of the methods is to identify these shortcomings and actively manage them by "boiling down" the complex element to the absolute minimum subset (maximum contextualization) so as to allow effective uncovering of any of these intrinsic fault classes. This only can be successful if the analysis is so well focussed that the "mind-set" of the initial design can be anticipated. With other words; to a level where the design decisions are consistent with the needs of the safety-related system or any such inconsistencies are uncovered and mitigated in sequel.

Note that HD^3 does not have a direct relation to Annex-QR but rather that HD^3 has potentials that can help mitigate some of the shortcomings of recommended measures and techniques for pre-existing elements if applied at the necessary level of rigor and thus, we currently believe, has potentials to mitigate some of the principle issues that arise when applying IEC 61508-3 Ed2 Annex C to pre-existing element notably with respect to intrinsic faults.

8.4 Tools qualification needs HD^3Tool

HD^3 is a process and a related tool the later is to undergo tool qualification. The current assumption is that tools supporting HD^3 would need to be at least T2. As the HD^3 is essentially emitting requirements the process is in the systems critical path. The overall process is layered and includes consolidation steps (during transitions to derived phases or next lower level of abstraction) and thus we assume that most faults would emerge during this implicit review [18].

[18] This actually occurred a number of times during the HD^3 sessions and commonly resulted in TODOs being emitted unless it was trivial/local inconsistencies

So while tool support for the initial process would arguably be of a lower qualification level (T1) The qualification need as T2 tool stems primarily from the use in later life-cycle phases, most notably retrofitting and upgrading. For these phases we must assume that the actors would need to rely almost exclusively on the data recorded and the interdependency presented by the tool as it must be assumed that these phases would not necessarily be conducted from the original staff and even if so, given the temporal distance, memory would not be reliable. Thus during these later phases the tool managed information would be on the critical path with respect to design decisions and analysis of any changes or upgrades. The actual rational and analysis step still is process driven and in this sense the tools do not emit requirements directly but would contribute to the emitted software element specification/changes including detection of conflict/defects and thus it would be [19].

8.5 Forward/Backwards Traceability

Forward/backwards traceability is mentioned in IEC 61508-3 Ed2 in many of the tables of Annex-A. This forward-backwards traceability is somewhat different than the stated aim in IEC 61508-7 C.2.11 *To maintain consistency between life-cycle stages.* due to the generative nature of HD^3 where the phase artefacts are internally (that is within the HD^3 process respectively tool) connected rather than externally.

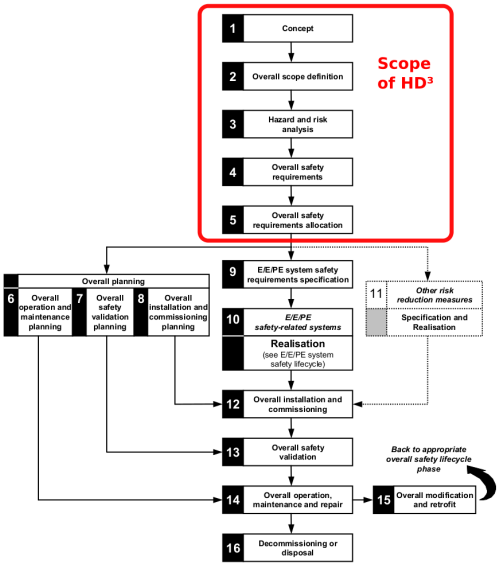


Figure 4: scope of HD^3 in context IEC 61508-1 Ed 2

The traceability between the requirements and design down to detailed design is inherent in the HD^3 method and the tool support adequately specified to achieve this traceability. Technically it is not a forward/backwards traceability but a sub-tree (up/down) traceability which is though equivalent from the intent of forward/backwards traceability.

In a nutshell HD^3 provide traceability from

[19] see IEC 61508-4 Ed2 3.2.11 software off-line support tool

```
Analysis to phase-artifacts
phase-artifacts to Analysis
```

which is elaborated on in the following two sections.

8.5.1 Forward traceability

IEC 61507-7 Ed 2 C.2.11 states that *Forward traceability is broadly concerned with checking that a requirement is adequately addressed in later life-cycle stages* enumerating the points in the system life-cycle where forward traceability vitally contributes to the safety life-cycle:

1. *from the system safety requirements to the software safety requirements;*
2. *from the Software Safety Requirements Specification, to the software architecture;*
3. *from the Software Safety Requirements Specification, to the software design;*
4. *from the Software Design Specification, to the module and integration test specifications;*
5. *from the system and software design requirements for hardware/software integration, to the hardware/software integration test specifications;*
6. *from the Software Safety Requirements Specification, to the software safety validation plan;*
7. *from the Software Safety Requirements Specification, to the software modification plan (including re-verification and revalidation);*
8. *from the Software Design Specification, to the software verification (including data verification) plan;*

HD^3 contributes significantly to all and in most cases can cover the needs completely. Design decisions may be recorded in companion design documents (currently we use elements of SASD - which seems natural given the analysis focus). The direct requirements are emitted from HD^3 as SACs and as Selection-Criteria.

A notable exception is the traceability to the assessment noted in IEC 61507-7 Ed 2 C.2.11 as the last point of forward traceability: *from the requirements of IEC 61508-3 Ed2 Clause 8, to the plan for software functional safety assessment*, though we do believe that HD^3 related information and notably the structuring of the phase related data will contribute to the functional safety assessment even if not formally integrated in the HD^3 scheme.

8.5.2 Backwards traceability

Backwards traceability has three main aspects in the context of IEC 61508 Ed2 1) management of iterations during development which commonly mandates re-opening previous phases, 2) during verification where phase dependency and transitions relations are checked (see IEC 61508-3 Ed2 7.9.2.6) and 3) during operational activities related to maintenance, retrofitting or incident analysis. Given the notorious contribution of maintenance and retrofitting to hazards the ability to inspect in-context based on HD^3 records seems a significant benefit of the method.

IEC 61507-7 Ed 2 C.2.11 lists the following phase relations that backwards traceability is enabling.

1. *from the safety requirements, to the perceived safety needs;*
2. *from the software architecture, to the Software Safety Requirements Specification;*
3. *from the software detailed design to the software architecture;*
4. *from the software code to the software detailed design;*
5. *from the software safety validation plan, to the Software Safety Requirements Specification;*
6. *from the software modification plan, to the Software Safety Requirements Specification;*
7. *from the software verification (including data verification) plan, to the Software Design Specification.*

Traceability from code to detailed design is technically infeasible for most cases of pre-existing software as there is no formal detailed design (in any case not in a traced manner) HD^3 can not "fix" that, but it can provide the necessary basis for focussed inspection and testing. Further the phases of modification is formally not yet being treated and thus in the current phase HD^3 has not analyzed the suitability of provisions to cover these issues - we do expect significant contribution though - notably with respect to supporting change-impact-analysis. Traceability from verification back to artefacts of design in general are not part of HD^3 but traceability capabilities will quite clearly contribute there as well, the formal backwards traceability for that phase though would lie in the verification documentation (which is mandatory anyway - see IEC 61508-3 Ed2 7.9.2.4).

8.6 Process Documentation and Records

8.6.1 Sessions

For each session, the attendance shall be documented with at least the data as given in

The intended usage is to have one entry per session. So if for whatever reason the analysis of an item is not completed, then this shall be recorded in a new entry, and the reason for the abortion of the first session shall be documented in the "notes" field.

Furthermore, if an item has to be re-opened then the entry for this session shall contain a note on why it was necessary to re-open that item.

If was used for the session then the IRC log should be referenced in the session attendance record.

8.6.2 HAZOP Tables: Item

For each layer of abstraction that was subject to a HAZOP study the respective table is to be created and stored in a suitable repository or data-base, Each shall be documented with at least the data as given in - preferably in a seperate file per item — preferably in structured ASCII text.

The motivation for calling for separate files and the use of American Standard Code for Information Interchange (ASCII) is to ensure long-term availability of data and easy searchability. Tool-dependencies should be avoided given that this data will be critical to maintain safety of the system until final decommisioning.

Note that some keywords (e.g. /) need to be regulated to allow automated post-processing - this has not yet been fully specified.

8.6.3 HAZOP Tables: SAC

For each SAC emitted during analysis atleast the data noted in shall be recorded. Records should be in structured ASCII text for long-term availability and easy of post-processing (counting and linking for a first estimate of severity respectively change impact)

SACs are **not** necessarily active mitigations, they can be constraints (to allow removal of hazards) or design/operations requirements to allow elimination of hazards (e.g. operation only by qualified personell).

SACs are not static during HD^3 sessions and only at the end of a complete analysis will they have been consolidated to a point where they can be managed by traditional Change Control Board (CCB). Change of SACs during HD^3 sessions mandate a review of possible impact on other items already referencing the affected SAC.

8.6.4 HAZOP Tables: TODO

For each TODO emitted during analysis atleast the data noted in shall be recorded. While TODOs technically would not need to be long-term available once closed we recommend using a structured ASCII format with one TODO per file anyway as resolution of TODOs caries a lot of relevant context information and thus we expect such records to be of value during maintenance and notably modification.

8.6.5 HAZOP Tables: Selection

During analysis of items in the context of potential pre-existing elements under consideration (short list) constraints on the elements may emerge. These constraints or demands on a possible candidate element shall be documented with atleast the data noted in .

Note that the selection criteria iteratively influence the design notably in the lower layers of abstraction. Thus the selection table is also input to session preperation (atleast in the technology aware specific analysis this should be mandatory).

9 Hazard driven decomposition, design and development (HD^3)

9.1 Short specification (overview)

The specification fields are derived from IEC 61508-7 Ed 2 and amended to satisfy the minimum needs of the proposed Annex-R (see R.2.5)

- **Aim:** Provide a system-context methodology for generating a complete and consistent safety-related system design.
- **Description:**
  HD^3 drives the system design from the initial system intent through a decomposition process guided by exploratory hazard and operations analysis at iteratively refined levels of abstraction — as defined in the HD^3 study — ideally the layering will reach the lowest design level e.g. -specifications/function-calls. The intent is to ensure that the

  potentials for hazard elimination and hazard severity reduction and/or hazard probability reduction is fully exploited before hazard mitigation is even considered. HD^3 has two major element to it

  1. Use of exploratory analysis iteratively applied to each design layer with the starting point of the development process being the "design free" initial system intent.
  2. Continuous recording of context, dependencies (up/down) and derivatives (SACs, selection criteria, derived requirements) ensuring that the hierarchy of the system stays analyzable/verifiable under modification.
- **References:**
  1. IEC 61508-3 7.4
  2. IEC 61508-3 Annex A.1
  3. TODO...
- **Technical references:**
  1. TODO: IEC 61882,
  2. Felix Redmill: System Safety: HAZOP and Software HAZOP, April 1999
- **Limitations:**
  1. Effectiveness unverified as of now
  2. Applicability to higher-complexity not yet clarified, at this point it is assumed to be applicable based on method review of current reference project only.
  3. Level of automation still in flux
- **Assessment:** Not yet applicable given the state of the method development. Submission to CA, once reasonably consolidated is considered mandatory.

9.2 Method intent

HD^3 is designed for the use in safety related systems that include high-complexity, type-B, components. For these components IEC 61508 Ed 2 and derivatives assumes that the fault conditions are **not** generically enumerable and the behavior under fault condition is **not** known for all fault conditions. The intent of a HD^3 process is to ensure that:

- Design dependencies in context of the design intent are detected and developed in system context.
- Possible deviations from design intent **relevant** for the system at hand are uncovered.
- The impact/effect of such deviations from design intent is systematically explored.
- Potential resolutions (not necessarily mitigations) are found **in context**.
- Derived items and or constraints are directly associated with the 'initiating' hazard-analysis 'event' and thus re-opening and or impact analysis can be conducted.

HD^3 is a system analysis process that strives to emit a complete and consistent design and sub-element requirements allowing to build an overall safe system.

9.3 HD^3 context

HD^3 is only meaningful in system context, it can not be applied to an element that is not embedded in a suitable context specification (see IEC 61508-1 Ed 2 7.2-7.5 which essentially contains the requirements for the system analysis context).

Complex systems are typically made manageable, despite human cognitive limitations, through divide & conquer strategies. This is nothing new, there are a number of analytical methods (Failure Mode and Effects Analysis (FMEA), , , etc.) that provide analysis of specific events or modes in a very narrow context (e.g. the specific failure mode) - HD^3 creates a system hazard context by allowing to link the levels of analysis in a systematic manner. It is though important to state that there is no automation that ensures consistency and completeness – this is at the mercy of the executing engineers. In this sense HD^3 is a managed analytical context suitable for initial designs as well as design changes and incident related impact analysis (which is a special form of design change actually). Notably the common issue of time-pressure during incident analysis of complex systems suggests that the design refinement should record the design hierarchy in a form that occurred failures can be traced to (ideally) all dependent components **and design decisions**. If HD^3 can actually achieve this is not yet answered due to lack of practical experience with the applying the method - it is though a design goal for the method. The context of the HD^3 process is formed by the distribution of roles and competence of the participating staff - there is no inherent property of the methodology that can ensure completeness and consistency of context and thus also no inherent guarantee of completeness and consistency of analysis. What HD^3 is anticipating is a systematic content specification as well as recording and procedural specification to foster a technical environment that allows achieving completeness and consistency of the system design.

9.4 HD^3 design limitations

HD^3 is not able, and not anticipating, the emittance of context-independent items or sub-elements. All analysis is strictly **in-context** and any change of context is, at least initially, to be treated as system-level change.

HD^3 can only structure what is present in the team running the HD^3 process - the determination of suitable competency and necessary resources is in the hands of the umbrella safety-management process. If resources or competence is inadequate then the results of HD^3 will be inconsistent and/or incomplete – probably both – and the overall results meaningless in the context of determining the achievement of tolerable residual system risk.

While we believe that there may be automateable indicators of incompleteness and inconsistency, we are quite sure that there is not affirmative indication of consistency and completeness - thus ultimately HD^3 builds on engineering judgement and competence.

9.5 HD^3 Proces Specification

1. Study pre-conditions (see section B.1.1):
   - System intent definition (informal) available.
   - Team competence available (see Table 2).
   - Long-term commitment of responsible organization – HD^3 is a life-cycle framework and might not be adequate if terminated before decommissioning at least if it is to provide the claimed properties - see Table 4.
   - Sufficient time available - from our initial experience we currently assume that no more than two items per day can be handled with adequate concentration of team members.
   - Consistency of team - while exchanging individual roles is doable (that is the whole point of formal definition of roles) it is assumed that such role-changes would be affecting individual roles and not whole sets. Team level inconsistency will result in invalid or unusable analytical results and/or emitted documents.
   - Suitable Configuration Management System (CMS) [20] to manage changes in a traceable manner.
   - Suitable communication channels (e.g. mailing list) to post change proposals for review.
2. Study Planing : (see section B.1.2) This is largely an overlap of those planing efforts outlined for HAZOP studies - see Technical References 2 - specific to HD^3 are the use of:
   - Text-based and thus diff-able file formats so that individual changes can be traced to engineer:data:intent(commit message).
   - Suitable asynchronous communication channel as pure fact-to-face sessions are too limited both organizationally as well as with respect to traceability.
   - Emitted artefacts: HD^3 emits sets of artefacts that must be managed, notably text-based recording may induce issues of duplication and/or inconsistency if not managed in a structured way.

[20] git - what else ?

- Design documents generated concurrently must be managed in a traceable manner - if technologies are used for design records that are not known to all team members they need to be introduced during sessions. Preferably only a pre-determined limited set of representations is used for design recording.

3. Study start : The intent to build a safety related system or a safety related version/variant of an existing non-safe system (prototype) is the starting point (see also section **??**)
   - Buildup of joint understanding: e.g. by domain expert introducing the overal system intent, the known system context, the skill-set expected by participating engineers (developers).
   - Team level (recorded) agreement on initial design intent statement (see section 10).
   - If found adequate - initial design decomposition (see section 10.1)
   - Trial element: Run to get acquainted with tool support and overall work-flow - either from the initial design intent statement or an element from the initial design decomposition. This first item analysis need not be completed if all team-members are good with the methodology and should be discarded.
   - Session preparation: It is mandatory that participants prepare for sessions by reviewing of relevant material, this material should be coordinated by the study leader (see section C.0.12).
4. Study session: Analysis of a specific item as team effort (see B.1.3)
   - Session preparation by participants confirmed
   - Status of relevant deferred items (TODOs) is to be checked.
   - Iteration over guidewords for selected items
   - Continuous recording of results and emitted artefacts including recording of any dissent
   - Postponed tasks are to be recorded with an owner (e.g. as TODO tags)
5. Item reviews: review of recorded item in repository as individuals asynchronous effort (see B.1.4)
   - Change proposals submitted to communication channel (e.g. mailing list) in a reviewable form.
   - Change proposals are Acked (take as is), Reviewed by (take with noted minor changes), Nacked (rejected entirely), or discussed interactively (traceable) until consent is achieved.
   - Change proposals accepted are applied by study CM to the respective CMS.
6. Item reopening: amendment/change of previous study item based on later study findings (see (see B.1.6)
   - Finding is recorded in the item at which the change was discovered Where a finding impacts a different item the affected item is also change/update. The rational is that

the "initiating event" of the finding should be recorded to allow discovery of procedural shortcomings [21].

- Origin of finding is recorded in the changed item (e.g. in the commit message).
- Impact of the amendment/change is to be checked by study team [22]

7. Item defect: change of previous study item based on study external events (e.g. failure report, field incident).
   - Root cause of the external event shall be determined and the affected items analysis reviewed as to why this case was missed
   - Related items may also need to be reviewed (e.g. if "As well as" in one item is found to be incomplete, this issue may affect other "As well as" cases)
   - External events should be recorded in relation to the study documentation - e.g. in a "external findings" table or the like. Rational: to ensure that the externally induced changes are available for any impact analysis during the lifetime of the HD^3 document.
8. Item modification: change of previous study item based on intended change of system design-change or feature update/extension (see B.1.6)
   - 1st level impact analysis shall clarify if the hierarchy of the study items holds for the proposed item change. If a parameter of a API is added this might be handled locally or may impact other items due to the API call providing some protection mechanism to other items (e.g. change of mlockall() parameters). The impact hierarchy shall be recorded.
   - 2nd level impact analysis shall clarify the impact of the proposed change on all items in the hierarchy found in the 1st level impact analysis.
   - 3ed level impact analysis shall handle/integrate any new SACs into the appropriate derived levels of analysis. Note that introduction of new SACs may change the lower-level hierarchical structure.
9. Session consolidation: (see B.1.5)TODO
10. Releasing Study: Releasing the study is bound to:
    - A defined (traceable) set of artefacts associated with the specific study results (tables).
    - Authorization of study results and labeled artefacts (revision see IEC 61508-3 Ed 2 Clause 5)
    - Notification of involved parties: as a minimum system safety manager and certification authority, and in case a system is still under development, engineering staff.
11. Closing Study: A HD^3 is formally closed with decommissioning of the system under study. In case where closing the study happens while a system is under development or active deployment

[21] e.g. the discovery of systematic misinterpretation of a guideword or a general concept level misunderstanding in the study team.

[22] for trivial changes, e.g. fix of typo or moving content that accidentally ended up in the wrong field no consent should be mandated.

in a safety-related context, the impact of the discontinuation of the study on the overall system safety properties are to be documented and justified.

10 Design intent as starting point

Typically safety related systems are designed based on functional and non-functional requirements - possibly from a customer. If the non-functional requirements include safety requirements that is in principle a good thing - provided the rational for the requirements are also given (which is a rare thing). The approach taken in HD^3 is to focus on the intent and derive what is needed through all hazards that may jeopardize this intent. This implies that safety needs to be a top-level intent - if not what are we up to anyway ?

The next step in the traditional DLC would be to start a refinement of requirements and then enter into some form of (probably preliminary) design. This implies that the requirements were completed without design or with some high-level design/trade-off studies maybe - commonly focussing on functionality: thats where we see one of the root-causes for design issues, but much more importantly this functionality focussed approach tends to emit requirements and design that is structured along functional relations and not around safety — safety is then patched on afterwards. The effect of this after-the-fact review and amendment, is that the complexity grows because most safety mechanisms introduced in the context of an existing design are focussed on mitigation **not** on elimination. Further the design rarely is minimal with respect to the safety scope simply because the structuring (e.g. partitioning) is driven by functionality the criticality is only detected in later phases and thus would call for a re-design if mixed-criticality would show up in the design — which is almost assured for complex software systems.

To mitigate this HD^3 focusses on the initial design intent as the analysis starting point **only**. The goal is to agree on the initial design intent statement and the use exploration **driven by hazards** to develop the safest possible structure:

- Eliminating as many hazards along the way as possible
- Allocating suitable mitigations at multiple levels where necessary
- Developing and recording the entire hazard hierarchy to ensure maintainability of system-safety properties during modification [23].

10.0.1 Design intent example

The initial design intent - for the Colimander Use-Case - was a simple one liner:

> *Measure and report E.Coli -specific enzymatic activity per volume of sample (indicating the level of fecal contamination) in at most 15 minutes, with defined and verifiable level of assurance.*
>
> [Colimander Design Intent]

[23] Our expectation is that in complex software centric safety related systems modification and retrofitting will by far more common than in traditional safety-related systems - provisions for this paradigm change we think are critical

All design decisions and ultimately the implementation, should be directly traceable to this top level intent. The goal is simply

1. Everything the design intent calls for shall be present in the development result
2. Everything present in the development result shall be justified based on the derivation from the initial design intent.

If this is not achieved in a complex system then it almost instantly loses maintainability. The mitigation in HD^3 is a hierarchical and fully traceable analysis that systematically drives the decomposition along with the rationals. Only by holding intent, design and rational together can a safely maintainable system be attained.

10.1 Initial design decomposition

In some systems directly starting at the initial design intent may be feasible. Basically exposing the stated initial design intent to the exploratory analysis focussed on deviations from intent - with other words the initial design intent simply is treated as a singleton requirements specification. Since the initial design intent will in general need to incorporate multiple aspects - in the case of the above Coliminder one can immediately extract:

1. Measure of E.Coli-specific enzymatic activity
2. Reporting of contamination
3. Time supervision (ensuring $\leq 15m$)
4. Runtime assurance

While this can serve directly as the top level items for the analysis it seems reasonable that an engineering team would have a somewhat more accurate idea of the realization. In case of the Coliminder Use-Case this was the initial description of the measurement process decomposed at a level that it could be manually executed.

1. Take "average" water sample
2. Put 5.5ml sample in a clean bottle
3. Dosage puffer (400ul) and substrate (20ul)
4. Stir sample with controlled speed to get homogeneous mix up and heat controlled at reaction temperature (so as not to kill the bacteria $\rightarrow$ No thermal hot-spots)
5. Measure fluorescence response over time (while stirring and maintaining reaction temperature) $\rightarrow$ slope
6. Calculate mMFU

Note that this functional decomposition already lacks the non-functional properties of the system — timeliness, assurance — it even missed the reporting (which presumably was seen "hidden" in the *Calculate mMFU*).

A first high-level review of this decomposition against the initial design intent would uncover those missing items and result in a first design decomposition of:

- Take "average" water sample
- Put 5.5ml sample in a clean bottle
- Dosage puffer (400ul) and substrate (20ul)
- Stir sample with controlled speed to get homogeneous mix up and heat controlled at reaction temperature (so as not to kill the bacteria → No thermal hot-spots)
- Measure fluorescence response of enzymatic activity over time (while stirring and maintaining reaction temperature) → read off slope
- Calculate mMFU
- Time supervision (ensuring $t \leq 15min$)
- Runtime verification of correctness (assurance)
- Report contamination

With these steps the entire initial design intent can be achieved and the specification is not yet at a granularity that it would demand extensive design efforts or specific methods. Note that the level of detail is not very constant (e.g. the "Stir sample...' is already noting some (obviously) known issues. Ideally the level of detail would be homogeneous but in reality this is rarely the case as soon as a domain expert is in the room and has her anecdotal experience to offer [24].

The simples path forward now is to define the 9 functionalities as our top level items for the initial technology agnostic analysis. In our case we concluded that the runtime verification should include the time supervision and thus the explicit time-supervision can be dropped as top-level item and further the runtime verification is not taken as an initial item but as a constraint on all other items allowing to emit suitable SACs to achieve the goal.

This elimination of items is not a formal step in any way - this is engineering judgement if it makes sense or not to include a top-level item or not and w are not able to give a hard criteria for when to do so and when not - what motivated us here is that the initial design decomposition should be functional focussed. Our final initial design decomposition thus include 7 items and those are reduced to the functionality - the constraints and limitations are left out - they will re-apear during analysis (which mandates presence of a domain expert).

[24] Rather than idealizing the initial decomposition we left it as it actually "happened".

10.2 Coliminder - Initial design decomposition

With the above discussed adjustments, the functional design decomposition is now at a level that it could be handed to a human for execution.

1. *Take "average" water sample*
2. *Put 5.5ml sample in a clean bottle*
3. *Dosage puffer (400ul) and substrate (20ul)*
4. *Stir sample at reaction temperature*
5. *Measure fluorescence response over time*
6. *Calculate mMFU*
7. *Report contamination*

[Coliminder - initial design decomposition]

If this person exhibits a diligent and professional work approach she will start asking questions as the specification leaves out too much detail to actually turn immediately to action. All that HD^3 now does is introduce a systematic method to ask these questions to achieve:

- completeness
- consistency

with respect to any possible hazards and their mitigation. Note that it is not the intent to pack the entire context into the initial design decomposition ! This dropping of details is to extract the items - the details are re-introduced during analysis where they should "emerge" in the appropriate hazard context and thus allow proper tracing rather than popping up from nowhere.This analysis is a per item analysis based on a slightly modified HAZOP process, outlined in the next section.

11 Item analysis

Managing complexity typically means divide and conquer. While this is all good the challenge in safety is to implement divide and conquer without losing traceability and notably the ability to modify with a reasonable effort. Specifically Clause 8 notes the relevant properties for assessment:

> *completeness of functional safety assessment with respect to this standard;*
>
> *correctness of functional safety assessment with respect to the design specifications (successful completion);*
>
> *the ability to modify the functional safety assessment after change without the need for extensive re-work of the assessment;*
>
> *repeatability;*
>
> *timeliness;*
>
> *precisely defined configuration.*
>
> [IEC 61508-3 Ed 2 8]

While assessment is not the prime intent of development, considering assessment needs is crucial for highly-complex systems, most notably if they are expecting a significant change/update rate [25]. In response to this our focus is on device and conquer in the context of relatively frequent modifications.

Modifications only can be managed in complex systems if the impact analysis for a finding is doable with reasonable effort - with other words if it can be confined to a readily identifiable subset of items in the overall system **and** the context/scope of these items along with their rational can be extracted with tolerable effort. The HD^3 analysis might seem quite overblown at first — and who known maybe it is — but it strives to satisfy the needs of modification and assessment after modification or the maintenance of complex software systems will fail at the first security vulnerability detected - effectively that would mean before shipping the first device !

As such HD^3 defines a relevant part of the strategy of *validation and modification* in safety planing as called 61508-3 Ed2 6.2.2. More specifically 61508-3 Ed2 notes:

> *If at any phase of the software safety life-cycle, a modification is required to an earlier life-cycle phase, then an impact analysis shall determine (1)*
>
> [IEC 61508-3 Ed2 7.1.2.9]

While this is quite clear and generally handled for the initial implementation it commonly fails during modification after longer periods of operations where the rational and intent, in some cases

[25] Even a conservative estimate would need to assume that any connected complex system would be hit by a or 2 per year

even the involved staff, is no longer accessible. As studies from [5] show even for traditional safety related systems the incidents induced by modification are significant. For a high-complexity safety-related system the defenses we believe necessary is a rigorous hierarchical management of divide and conquer from the very begin to ensure safe (and economically viable) modification.

11.1 HAZOP

HAZOP originated in one of the industries that had complexity problems very early on - the chemical industry. The problems were due to the size of the systems (e.g. refineries) simply making it impossible to reason about faults from a local context - notably also due to humans being part of the control system. The key property of HAZOP is that it is exploratory in the sense that it does not build on prior knowledge of failures/failure-modes but rather takes the intended functionality as the per-item starting point and then drivers the analysis by systematic exploration of the deviations from the intent. This of course assumes that the intent it self is sound - HAZOP it self does not guarantee that. The exploration it self is not by anecdotal knowledge or "experience" as is largely the case with /, but is driven by a set of abstract guidewords that allow covering the potential range of deviations. These guidewords must be interpreted in context and thus domain know-how is typically a key aspect in any effective HAZOP study.

Traditional HAZOP was though excluding concurrency and also security - more on that later. For the overview here we refer you to Redmil [6] book on HAZOP respectively to the appropriate IEC standard IEC 62882 [4] if you are not familiar with HAZOP yet.

11.2 HAZOP primer

In this section we ignore the organizational and competency issues associated with the HAZOP method and focus on the "mechanical" procedure of the item based analysis. Note that the term *item* here refers to the function being analyzed and should not be read as a ISO 26262 Item further function is not a very formal term here - effectively the item describes some functional requirement e.g. "Store data" or "Detect Object" and does not say much about how this function would be implemented. Systematically the HAZOP session then analyzes the possible deviations of *any* possible implementation determining potential causes, consequences and preventive or mitigating measures along the way. The function - typically the name of the item under analysis - is far to vague though to conduct a meaningful analysis - so the first step in a HAZOP session is to deliver a interpretation that is accepted by the entire team in the context of a specific guideword.

An item could be "Take Sample"

1. First define the item e.g. "Take Sample" must be clarified during the session what the context and meaning of Take Sample is - in some cases this may need recording (e.g. in a notes section) if not clear. In our Use-Case Colminder, "Take Sample" means sampling possibly contaminated water from a defined water source.

2. The guidewords is selected - e.g. "No". The usual guidewords are No, Less More, Part-of, Other-than, As-well-as, Early, Late, Before, After. For the scope of these guidewords see annex-A.

3. The problem is "No" again has no meaning without interpretation. So that is the next field. In the context of "Take Sample" vagueness is to be resolved by providing and agreeing on an interpretation - e.g. "No" might be interpreted as "No sample was taken" in the meaning of "nothing sampled" with other words an empty bucket was drawn from the water source.

4. For the given interpretation of the guidewords "No" here possible causes are reviewed by discussion in the team - that the explorative part on the causes. It could be that the river is dry or that the bucket is broken. The goal is to record *reasonable* causes which implies that they may be incomplete [26].

5. With the list of causes at hand we now can ask for possible consequences - quite clearly these consequences are valid only in the context of the analysis. If study members think that the context might not be sufficiently clear from the item and guideword then it needs to be recorded. The criteria to be placed on detection of causes and consequences is *reproducibility*. A refinement of consequence we introduce is to classify consequences as

   - false positives - the function is signaled to have succeeded while it actually failed silently.
   - false negatives - the function indicates failure while in fact the failure state is not known or success has occurred.
   - maintenance - the result of the function are not critical for operations immediately but relevant for detection of systematic deviations by data-analysis (e.g. during maintenance or by system monitoring).

   What severity these classes have is context specific but in any case they form an order of severity for the given level of analysis.

6. Once consequences are enumerated for this particular set of causes determined to be reasonable, the next step is to ask for indications and possible mitigations. If there is no indication then we would not be able to detect this deviation from design intent in which cases it may be necessary to extend the functionality to allow indication or extend the item to allow mitigation. It should be pointed out that it is possible also to eliminate hazards e.g. by constraints on the Function or by delegating the assurance of some pre-condition for success to a different unit - e.g. it is hardly reasonable to expect that someone would try to draw water from a dry river - if it is an automated system this becomes reasonable and for that context then one might resolve the potential hazard by adding an indicator e.g. weighing the bucket after drawing the water. But this just increased the complexity of the system - so maybe one could have an operation constraint that the contamination is only measured if the river is not dry e.g. by some initializing process and then it is not reasonable to assume that a river would suddenly go dry - or the indication might show up in a later step during processing of the water sample anyway - in which case the indication might be omitted if the consequence is assumed to be low (and thus a single protection mechanism sufficient) or one decides to have multiple levels of protection if the consequence of drawing an empty sample would be assumed to be more significant — what this little discussion should show is that focussed on a item **and** context constraint by the guideword **and** domain know-how in the team, the complexity is manageable.

[26] which is why it is so crucial to have rational recorded later so that an analysis can be re-opened with reasonable effort

7. The last step left now for this item is to record any questions as well as SACs decided to be needed or TODOs (with assignment) and then store the dataset in some long-term data-base — long-term as in corresponding with the life-time of the system under analysis.

A randomly chosen sample following the above scheme might look like

```
id: [24]
item: I/O Setup
guideword: Part of
interpretation: Parital access mode and access permission.
cause: incorrect control; inconsistent configuration
consequence: data loss
indication: see UTS\_1020 (storage media is grey-channeled),
  STS\_3014 (access modes pre-determined)
question: NOTE: mount RO but access RW. TODO\_18 (andi): check if
  protection layers (seccomp, cgroups) can check if a file
  shall be used RW but is only used RO.
```

Referenced SACs (e.g. UTS_1020) are given with there short description TODOs needed for clarification are assigned to a particular user and managed in a separate table.

11.3 HAZOP extensions

HD^3 is still work-in-progress and during our initial sessions we found that some minor extensions to HAZOP with respect to recording and handling of issues was needed which extend the usual definitions. We do expect that this will need more refinement over time and is not yet sufficiently consolidated for the general case.

1. Use of "Macros" for names that are very specific to the system and/or specified in some companion design document e.g. "FillMEASUREMENT_CELL" here MEASUREMENT_CELL is assuming a defined specification or meaning.

2. Interpretation field: it seems most HAZOP form do not include a interpretation field and rather than recording it formally it is simply discussed and agreed on during session. The recording in context of the specific item is important for maintainability and later impact analysis (did the interpretation cover the new case ?)

3. Deferred work in the form of (managed) TODOs. A TODO is at least assigned to a responsible person an receives a unique ID. One might find additional information worth encoding (e.g. the layer or SAC at which the TODO was invoked - but we currently were OK with unique numbering of TODOs only).

4. Allocation of qualitative severity class (/ and maintenance) note that in general the actual severity depends not only on the class but also the level of indirection that is, a failure of functionality (without redundancy or monitoring) is more severe as the failure of an indication

"only" — while the former may serve as an immediate initiating event to a hazard the later would "only" fail to mitigate a primary failure.

Technically not an extension to HAZOP in a safety process but a somewhat different role in the way HD^3 uses the CM process. Most HAZOP documentation covers conducting a single study and notes that one should conduct such studies at "appropriate times" but does not explicitly (or implicitly for that mater) note traceability from one study to the next. Essentially this is one of the key issues of HD^3 that - as IEC 61508 Ed2 frequently states the importance of "testing and analysis, and testing in complex systems is of little help, therefor the onus is on analysis - allowing to manage the complexity of the system analytically. The importance of traceability changes quite significantly in HD^3 over traditional HAZOP.

11.3.1 Managing concurrency

As current safety related systems are moving towards multi-core systems the question of concurrency is becoming very much more significant. Not that it did not exist before - even a single-core system with a multi-threading OS on it offer pseudoconcurrency - but with physical concurrency the probabilities dramatically increase and this has implication for safety - concurrency can not be ignored or reduced to "fixing priorities". To address concurrency issues we added guidewords to the HAZOP - interrupted and terminated - and extended the HAZOP process.

The extension is **not** to simply treat these two additional guidewords during session but rather to complete the "serial" analysis using th traditional guidewords and then rerun the item analysis for the concurrent guidewords in a separate session. This, we think, is necessary for the HAZOP team to actually mentally perform the necessary context switch to concurrency issues. A side effect that showed up nicely was that this is an effective review - notably the concurrency considerations may reopen guidewords or call for changes to interpretations of the original analysis.

11.3.2 Integrating security

While some work on joining safety and security analysis - both being system properties that are connected - is an on-going effort. IEC 61508 Ed2 finally acknowledge the need to address security in a very early phase in *the context of hazard analysis* (see IEC 61508-1 Ed2 7.4.2.3). Andreas Platschek proposed a set of adjusted guidewords detailed in A.3 as well as [7].

Currently $HD^3 has not yet been using the security guidwords and we have no credible experience if this would be effec$
$as such security is not yet considered in$ HD^3.

11.4 Recording and Traceability

Traditional HAZOP sessions were meeting based team sessions recorded during the actual analysis. While it is common to the define follow-up work that is executed later (similar to the TODOs we use), the key issue identified is the way these off-line handlings are managed. The management in the HAZOPTool is based on git and for followup work we thus also used git. To properly manage changes to items, SACs, TODOs and Selections the formal process was to submit the change-request as patch to the dedicated hazop mailing-list and then wait for atleast two other study participatns to acknowledge and/or review the proposed changes befoer CM was ask to merge the changes or

change-sets. This process may seem slow at first but it turns out that the recorded discussions help as it is common that the changes needed are not single instances but reappear at multiple occasions (e.g. in a specific guideword or in multiple guidewords of the same item).

Recording of the sessoins is done in the HAZOPTool that records all of the initial data submitted during session. The discussion it self was not recorded extensively - at most it poped up in Notes and in TODOs. Some experimenting with based sessions turned out to be a bit slower but far simpler to follow or revisit discussions later - we would actually recommend using IRC or a comparable text-based recording of sessions over traditional recording by a dedicated member of the analysis team.

All records are then stored in a repository by tables - we use git but probably any reasonable CMS could do it (not sure if there is an reasonable CMS aside from git though). Keeping things in seperate tables witch each of the items stored in structured ASCII-text file - one-per-item - is a very effective way as it allows to readily search through the files and post-process them with simple tools (e.g. to get basic statisticas on frequency of SACs, length of resolving TODOs and other meta-data baed metrics).

To allow reasonable access to the quite extensive data collection - notably during maintenance and impact analysis this is crucial - the data is structured into a set of tables and then compiled into a final report with a bit of glue logic interspersed for improved readability. The Colminder report is not included in this document [8] and we might add that this first instance of a full system decomposition and design with HD^3 also shows some of the evolution of the method - as such we point out that only the later tables (technology aware specific) actually satisfy the criteria outlined here. A rerun is scheduled but to understand the method and the development this early Use-Case actually seems better than a clean and fault-free (artificial) report.

No method is perfect - no tool is perect - and both method and tool are usless (or worse) if used by people with the wrong attitude/mind-set. HD^3 traceability intends to capture the structure of the systems fault behavior and record the data necessary for maintenance, retrofitting and impact analysis in case of incident reports - the traceability rules can not resolve these needs at best it can provide a structured approach to allow reaching the goal of covering these needs - the rest is based on the enineering capabilities, competency and willingness.

12 Derived analysis layers

The layered approach taken in HD^3 results from the design nature of the method. This is where HD^3 goes beyond a classical HAZOP type approach. Typically analysis results are fed back into the appropriate life-cycle phase. In HD^3 the feedback is much tighter and the life-cycle phase more or less emerges from the HD^3 activities.

The example we will be using is admittedly quite absurd - but it turns out to be better to use absurd examples than real examples where then the discussion focusses on the technicalities of that specific safety related function or the environment rather than on the issue at hand - the process of grouping SACs to emit the derived layers of analysis.

12.1 function and scope of derivation

The starting point of the technology agnostic analysis is the documented initial design intent which is expected to be expressed in natural language and may well be quite informal e.g. "measure bacteria in water". It also may be at a slightly more refined level where the basic steps are already anticipated in a very informal way. This initial design intent is what is developed in the top-level technology agnostic analysis. Any of the hazard elimination SACs determined will show up as constraints in the applications and also need to be considered as constraints for any of the lower level mitigation proposals. Any of the mitigations - effectively the first level of safety functions - specified as SACs are then consolidated at the end of the analysis step and transformed into the derived layer of the analysis. Not all of the SACs need necessarily end up in the derived layer some may go "UP" - forming a basis for operational restrictions - and some may stay in place and ultimately need to be considered at the application level (technology aware specific analysis) even ending up as coding restrictions. Those that can be consolidated reasonably should be and then subjected to the same level of detailed analysis using HD^3 item specifications.

The relevance of this layering is that this allows a first split with respect to criticality. A function that is only there to detect a fault or mitigate a fault is of a lower criticality than the initial function that actively was contributing to the hazard. Further the differentiation between false-positive and false-negative allows to further subdivide criticality. The goal of this early qualitative classification is to make sure that the most critical elements in the system also get the highest level of attention (which is often not the case). In the lower levels hazards may emerge that already have a tentative mitigation (a SAC) in place in a different item and thus this SAC should be referenced. A certain care must be taken not to introduce too large numbers of nearby SACs - thus SACs must be documented along the way and before introducing a new SAC the available SACs should be reviewed - it showed that oftentimes a specific SAC can be generalized with a minor change and thus reused in multiple places - which is preferred over having multiple closely related SACs. This joining of SACs needs to be a documented process - and the method we use in HD^3 is by allocating a TODO (a traced element) to perform such a joining [27].

[27] in rare cases this may mean to re-open an item that had the original SAC as mitigation

12.2 Derivation methods

While UP SACs will also undergo some sort of consolidation, they are not considered here further. SACs that went UP are not part of the technical implementation [28]

The consolidation of SACs has a lot to do with grouping SACs - the grouping problem is not something where there is one correct view. An example: glibc and firefox both are groups of functionality. The commonalities of firefox functions might be something like:

- web-resource access functions
- content rendering and display
- security capabilities
- ...

The description is functionally driven and one might then find such groups of functionally related code-level functions grouped in files. For glibc this could be described with:

- low-level/generic function
- user-space side of system-calls

This is not implying that memset() and lseek() are in any way functionally related though - so the descriptions have two very different viewpoints of the function groups. The consolidation of SACs thus depends a lot on what level of abstraction one applies. Our (current) recommendation would be to try and apply generic abstractions more than functional abstractions for the grouping. This will be visible in all of the consolidation methods. The three approaches we present here are based on quite preliminary experience and we assume that there are system where none of them fit - in that case the only real requirement is to document that new approach, provide a rational and include it in any relevant documentation so that the difference to the methods described here is clear.

The three aspects we will outline for the method used here are:

- Grouping by functional proximity (coupling/cohesion)
- Grouping by criticality and hazard scope
- Grouping by location in the system software architecture

We do expect that one will actually be using a mixture of these methods in practice. Also note that grouping may shift with increasing design insight and allocation/selection decisions (e.g. from kernel to user-space or vice versa).

[28] They may though appear as rational in any technical implementation.

12.2.1 Grouping by functional proximity

The first option for grouping "hints" is the SACs that appear together providing a particular indication/mitigation, a further "hint" is SACs that address the same item - so probably those SACs that a particular element emitted do have a certain inherent relation to each other. There is no one rule (as usual) for determining coupling and cohesion. It does though depend a lot on what abstraction level one uses.

Possible groupings to consider would be

- Abstract consolidation:
  - per criticality - all false-positive protection into one and all false-negative protection into one + maybe helper functions
  - split along indication/mitigation maybe in addition to criticality and/or helper functions.
  - allocate to abstract generic functionality: Timing, Storage Helper, Trending, Monitoring.
- Specific consolidation:
  - per item - so all the Alarm item SACs into one AlarmHelper
  - allocation by functional relation derived from the SACs directly like "Estimate speed/location"

Abstract consolidation may (we really do not have the experience yet) provide a higher re-use factor for other projects, specific consolidation on the other hand is to us easier to understand and that would be the key argument for choosing it. But this should not be seen as an exclusive or disjoint set it well may make sense to split along criticality first and then by item. Multiple levels of consolidation may then result in an optimum.

One can also approach SAC consolidation from the direction

- Bottom up approach

  build a relation tree for all SACs (in theory all-to-all so not too attractive) and group the SACs by commonalities likeappear in the same item appear in the same guideword frequently appear together

- Top down approach

  Define abstract categories and assign the SACs as best fit to those categoriesStorage: everything calling for "record" and/or "trending" could be consolidated into one SAC and its derived function Alarm: everything calling for respond to error-events or propagation of detected error might be put into a common Alarm SAC and its derived function.

The top-down approach seems closer to usual functional decomposition and may have advantages with respect to understandability if the cohesion of the aggregated SACs is high and the interfaces (coupling) natural. The bottom up approach can help uncover systematic problems (unintended coupling) as well as globally missing considerations and thus probably the bottom-up information should be used for any top-down consolidation. One aspect that is particular to the top-down

approach though is that it may allow introducing safety functionality that is common domain know-how even if it had not emerged in the analysis. [29]

There is of course a certain subjectivity to this, and it technically is not relevant for the SACs how they are allocated as long as all are satisfied, but a bad allocation could create a very high (useless) complexity and that is inherently bad. Note also that the cases in the example analysis conducted often readily allowed including a note that some SACs could be joined - in general by assigning a TODO for such joining of SACs so that traceability is ensured, The final decision of joining or not is only decidable if you know your allocation strategy which is why it generally should be deferred and not decided ad-hoc. A counter example of early joining could be that it may make sense to keep to SACs separated that are technically similar simply because they differ in criticality and joining them would increase the complexity of the high-criticality SAC - which might not be the best thing to do.

The criteria for the allocation for a safety related system would be to constraint the complexity of any of the allocations in relation to the criticality - the defense against false-negatives is of higher criticality and thus should conceptually be easier to understand and of lower complexity.

12.2.2 Grouping SACs by criticality and hazard scope

Effectively this builds on the first consolidation round outlined above and as such it is a 2nd Round of consolidation into a set of derived items so the UP features are already taken care of at this point.

All DOWN features are consolidated into derived items, we refer to features not SACs here as SACs may contain multiple functional or non-functional requirements that can thus be allocated to multiple derived items

Derived items are placeholders for design intent extensions and in some cases may also include design intent constraints that might not have any functional implementation in the final system but just a constraint type requirement (e.g. configuration limit or the like).

The original intent had no notion of estimate X or Trending of Y so these are design extensions and they (at the suitable level) will then provide the services or capabilities outlined in the allocated SACs.

As an example we take an item RecordData. There commonly are a number of SACs that emitted data recording needs - so all of those can be consolidated to a core high-level function RecordData. The data recording element then need to be compatible with all the assigned SACs - it will though not actually implement them, e.g. RecordData is concerned with how to record e.g. a timing data element not with how to generate it [30] .

The High-Level derived functions still are technology agnostic - though in some cases this can be highly artificial - for RecordData it is not hard to imagine that a human could record all data items with paper an pencil. The analogy to human operations should not be overdrawn - in some cases it might simply not be possible to come up with a reasonable human executable process - which would indicate a mandatory allocation to an automated entity of unknown technology (electric, mechanical or programmable).

Note also that SACs may be allocated multiple times simply because they fit multiple abstract categories. For each SAC the criticality is basically known as it was introduced to either cover a:

[29] if that happens it would be a probable indication that the analysis is not complete or the necessary domain know-how in the team is insufficient.

[30] the precision of timing information or the format.

1. false negative -> critical
2. false positive -> less critical
3. non operational impact (e.g maintenance) -> low criticality

But all the derived design extensions have conceptually **lower** criticality than the initial functions as (if designed properly) they only can emit false-positive but no longer false-negatives (this implies non-interference here !). When would this assumption be violated ? Here are a few criteria to use (not that we actually already know all of them

1. false negative > false positive
2. multiple use > singe use
3. unique mitigation > mitigation set member
4. multiple items > single item

The first should be clear from above, the second I guess is intuitively clear, as a failure of the SACs implementation would jeopardize multiple indications/mitigations. The third applies to SACs that are only employed in a single instance, if that fails it would only impact a single point in the system [31] - many system are though built to not permit a single point of failure - the "mitigation set member" e.g. could be a set of elements related through Layer of Protection Analysis (LOPA). The fourth is only true for items have a relation to each other that allows a later item to detect a design deviation - if they would share a SAC it could be a potential Common Cause Fault (CCF)

It is important to record this level of criticality to allow effective allocation of efforts with the target of maximum hazard reduction. If a function that is low criticality from the safety perspective takes up too much resources then these are missing in the critical protection functions. For critical protection functions we might decide to use architectural means or LOPA while for less critical the failure rate might be tolerable. This also plays into technological selection decisions - for a low criticality function using AI/ML might be justified simply be functional superiority for a mid criticality this would probably not hold.

Note that we generally can not allocate risk here because it is not easy to say what probability of failure a indication/mitigation has in general We could try to build LOPA type qualitative dependencies - but this is not the topic of the derived item problem.

The goal of the derived item consolidation is to find a ideally minimal set of items that have **maximum cohesion and minimal coupling**. If SACs can be allocated to unique derived items that is an indication of low coupling (good) if many SACs pop up multiple times it means that the allocateion has potentially high coupling (bad) - in that case a further iteration is advisable. Thus there is a strong relation between consolidating SACs by review/modification and deriving new items as depicted in figure 5 [32].

Note that the level of coupling/cohesion really depends significantly on how you grouped SACs - with other words what grouping you considered.

[31] Notably rarely called functions may reduce criticality due to limited exposure.

[32] Note that the roles of TODOs and Selection recording is ommited from this figure

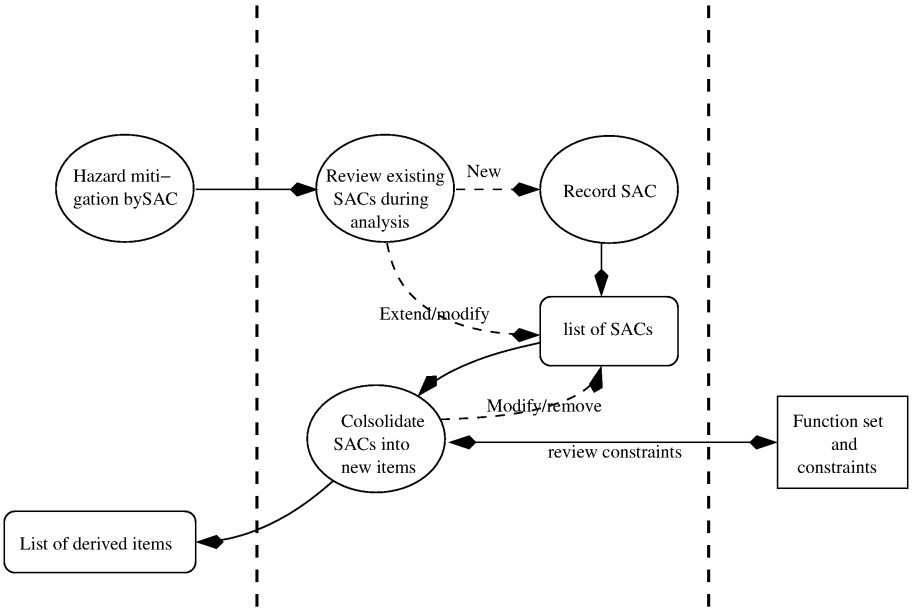


Figure 5: SAC consolidation subprocess (see also figure 6)

The example case is allocation by functionality of the SAC and then (if warranted) split by criticality.

12.2.3 Grouping by functionality

Grouping by functionality focusses on functional cohesion and is close to traditional functional design. Basically the cumulative SAC list from the analysis is simply treated as requirements set and allocated to a set of functions. This set is driven by coupling and cohesion considerations and is not a formal process but more a engineering judgement issue.

Criteria for functional grouping would be common helper-functions or common resources (.e.g timers or storage devices, etc.) and strive to retain a hierarchical decomposition of the systems elements. This is not only quite common approach for software function grouping but specifically in the safety related context crucial to retain maintainability of the system. If the underlying safety releated dependent functions turn up as CCFs across a larger set of functionality the impact of field-findings/analysis-findings may be problematic to handle (or at least larger than necessary effort). Such sharing also would imply opportunities for fault propagation so ideally they should be limited.

12.2.4 Grouping by criticality

Criticality has a few aspects, questions like is it protecting against false positives or false negatives ? how often is the same mitigation called for and thus haw many items could be affected by systematic faults ? Are the elements at the same level or at different levels (e.g. functional vs safety-response)

In some cases it may be possible to also identify the severity during analysis in ordinal categories e.g. high, mid, low directly. What seems doable in any case though is a simple classification over

counts.

- Layer at which the SAC was introduced
- Nr of items utilizing the SAC
- Nr of items in different subitems (at the lower layers of decomposition)

These ranking attributes extend the above set. With these course-grain meta-data for SACs recorded one can provide a first severity ranking of SACs.

- higher layers > lower layers
- higher reference count > lower reference count
- CCF: Nr. of items coupled by SAC

How now to relate the individual factors against each other is an open issue at this point. At present the only answer we have is engineering judgement and provision of a reviewable rational allowing to obtain an independent second opinion. The intent of the ranking is to allow efficient resource allocation and impact analysis on reporting of a finding.

12.2.5 SACs in lower levels

Some of these functions might not actually be treated as high-level derived functions but could be "delegated" to later levels simply because a technology agnostic interpretation does not make too much sense in some cases. It will though not hurt to do an analysis on an item that is only a proxy for a lower-level item (it probably would not emit any new SACs either). The solution is that SACs have the layers indicated in their names so rather than consolidating a SAC into a function that is unnatural it is an option to simply leave it as SAC. Currently we do not intent to differentiate SACs that are "treated" - that is consolidated into some function - and those that are "untreated". This is an open issue if this is needed or not (for our first experiments it did not seem to be necessary.

While consolidation is important it should be noted that overdoing it can backfire as well - at the highest level we do not yet know the criticality and thus consolidation of multiple SACs into derived items (functions) may result in mixed criticality functions which may increase the complexity unnecessarily - the decision to consolidate SACs or leave them as "untreated" is an engineering decision (put differently we have not yet found hard criteria pointing to the one or the other solution).

The next level of analysis is already emitting SACs that are, in case of derived functions, there to prevent non-detection of false-positives/negatives and prevention of false-positives. Keeping this in mind is important as it may be legitimately decided that increasing the design complexity further to prevent a non-detection of a false-positive is not justified. Essentially hazard driven design may drop possible mitigations if they are responsible only for low-criticality events and the design complexity increase is viewed as more harming than the benefit of the added diagnostics. There is not rule for this - this is engineering judgement.

If we can manage to trace the criticality all the way down to the lowest level of design - just before we transit to implementation or selection of a particular candidate:configuration - then the

high-level safety impact of failed isolation can be judged quite accurately. This is a common problem in complex designs that at the lowest level nobody actually knows what the hazard contribution of a failure is - not even at a qualitative level !

12.2.6 Grouping by location in the system software architecture

Note that the "UP" SACs have been removed. The remaining SACs are now listed per top level item that emitted them item. The first step is to add in the criticality. Those protecting against false-negatives are high those against false-positives are mid, SACs that would only be contributing to maintenance or diagnostic of system failures might be ranked low. At lower layers SACs that are secondary SACs (that is a SAC that is emitted while analyzing a SAC of a higher level) would be of lower criticality in general on the other hand they may be called upon more often which then again would raise their criticality. We do not have guidance on allocation of severity yet simply due to lack of experience.

Grouping by location tries to keep the SACs within the analysis scope of a particular function. Some SACs e.g. monitoring needs, may be emitted from almost any part of the system - if consolidated into a single functional unit as a derived item this may create not only an obvious CCF in the system but also may invite fault propagation via the interfaces for monitoring.

Grouping by location currently seems to be a relatively weak criteria - but again: experience is very limited to date - and if it seems that SAC consolidation would induce "unnatural" coupling it may be advisable to leave these SACs "untreated" and resolve them later in the decomposition so that they can take into account software architectural potentials (e.g. partitioning) to break undesired coupling in the system.

12.3 Conclusions from this first mapping analysis

One could now refine the criteria and see if that emits further hints for grouping or one simply accepts that as sufficient result and uses engineering judgement. One - important - side-effect of the exercise is that one gets a better understanding of the system under fault conditions, and that is what then actually allows to use engineering judgement - the tallying and simple guidance rules are just a helper four our meager brain-power!

As before we try a grouping of the SACs - it well may be that SACs can not be merged into sensible groups and a SAC ends up as a singleton in a derived item - thats OK as long as its not too often - dont try to force merge SACs just to get a short item list. If SACs are joined into derived items in a very "unnatural" way it does not simplify the problem as the complexity then re-emerges during analysis where one then has to split up the different call-sites of the SAC - again engineering judgement.

The SAC consolidation process gives us a first set of derived items that we then can analyze again. As a precursor to technology selection as well as deriving the preliminary architecture we now cluster those groups into technological categories - this does **not** change their criticality but it seems clear that they will be allocated to different technologies and or logical units in the overall system. This logical units might be System Processes, Safety Processes, Application Processes, Hardware, Other.

12.3.1 Is this SIL-decomposition in disguise

No - well - not yet - there is no intent to de-compose the SIL of the item, see IEC 61508-3 Ed 2 7.4.2 (notably 7-11), we are not partitioning or claiming independence. If that were desired then the grouping of SACs would need to emit independence assumptions and/or constraints that then can be consider in the SW-architectural decomposition - see grouping by locality above. Grouping by criticality or having multiple indications/mitigations available may permit a LOPA type protection arguments but SACs are in general not IPLs. If such a de-composition (e.g. 2-channel diverse detection of some fault) is possible it should be noted but not immediately mandate until we have a first estimate of the criticality of the SAC/item to justify the increased complexity.

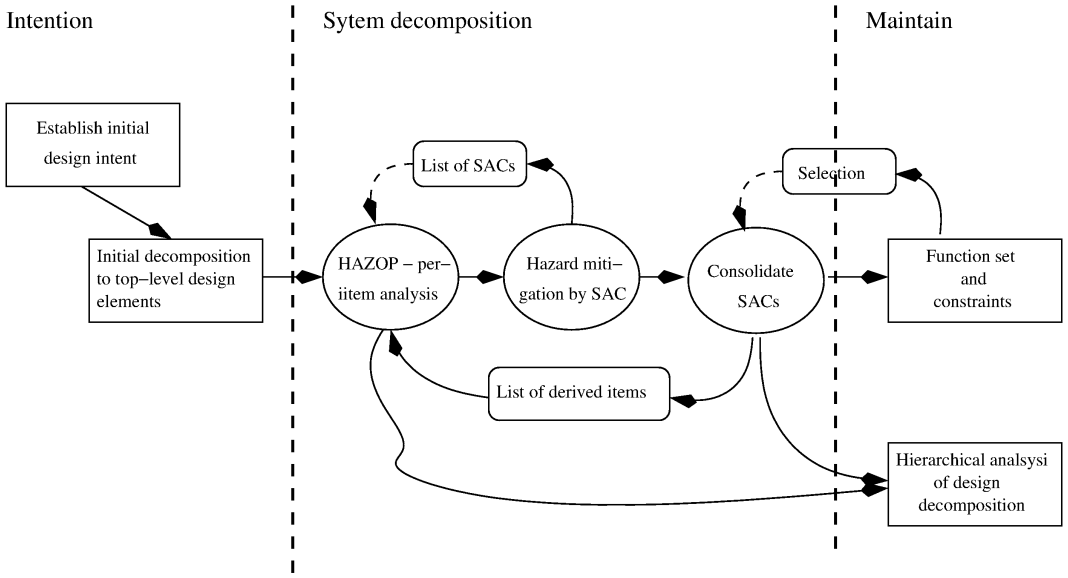
Finally - every SAC is increasing the complexity of the system it is a continuous need to tradeoff this growth of complexity against the increased diagnostics or reactive capabilities of the system. It well may be possible to drop some SACs at a lower level if the hazard is deemed tolerable or there are other indirect ways to detect/mitigate a failure with only slightly lower assurance while gaining a reduced complexity. Read this as a general warning to not plaster over a broken design with SACs - that will not result in a safe system - if SAC allocation seems to be going out of control then a overall design review might be needed to mitigate the situation.

13 Application of HD^3

This section outlines how to put together the different steps described in previous sections in an actual task set and integrate it into the overall work-flow. With respect to the overall safety life-cycle HD^3 is located in the top of the V-model with the technology agnostic being entirely in part 1 of IEC 61508, technology aware unspecific focussing on part 2 of IEC 61508 with a not-so-sharp transition to part 3 of IEC 61508 which is mostly addressed in the technology aware specific layers and below.

13.1 Study Work-flow

The simplified HD^3 work-flow can be split into a preparatory phase that emits relatively few documented result but never the less is crucial to the overall process of understanding the hazard potential of a system, the core iterative analysis process, and the result handling. The somewhat simplified figure 6 leaves out a few of the iterations and dependency relations.



Preparation:

- Role assignment
- Study planing
- Study schedule

Study session:

1. iterate over items
2. record item, SACs, TODOs
3. followup work

Long-term management:

- Analysis trace
- Session records
- data repository

Figure 6: HD^3 process flow overview (simplified)

13.1.1 Study preparation

A Key intent of HD^3 is to provide a systematic analysis methodology for new systems and new technology. For traditional, largely evolutionary, development one typically can build on a wealth of knowledge accumulated by staff — some of the knowledge is formal and fact-driven, some is quite anecdotal never the less of value during development of the context and scope of a new system. For new system utilizing novel technologies both inputs are generally weak and there is a tendency to focus on the known problems and overlook the often systematic issues with the novel parts of the system design. Notably relying on FMEA type analysis as a prime tool is critical when new concepts and novel technology plays a major role in determining the systems design.

The proposed mitigation of the limited understanding of the new system as well as limited understanding of impact and dependencies of novel technology elements is to drive the entire development by an explorative process focussed on the early safety life-cycle phases - specifically IEC 61508-1 Ed2 clauses 7.2-7.6 (and in case of predominant usage of Route 3_S the proposed extension 7.X see also 8.1). The preparation of the study thus is quite limited - effectively the starting point is to get a team agreement on a short design intent statement - this might be a single sentence (see 10.0.1). And starting at this design intent statement deduce the first **functional** decomposition to the top-level items. The top-level items should cover the entire functional scope of the design intent and may be extended by "obvious" items - though we pledge for a minimalistic approach and prefer leaving out additional items. They will appear - if needed - during the analysis (see 10.1).

With the initial design intent developed in the study team and the initial functional design decomposition established, roles for the study can be distributed (see 2). It is important to note that a major goal of explorative studies like HD^3 is to grow the domain-expert base. It should not happen that during the entire study only the domain-expert actually understands what is going on - notably with novel technologies this would indicate a critical inability to communicate concepts and actually invalidate the study results. This dissemination of know-how is something that must be actively sought by all study participants - this does **not** happen auto-magically.

13.1.2 Planing and schedule issues

Planing in HD^3 is at the level of individual session while one probably can make rough estimates on the number of items that will need to be studied once a few systems have been analyzed currently we only have a single moderately complex system to dram from thus not enough experience to offer a credible estimate of efforts. So the planing is limited to the preparation phase and to the resources needed up-front.

Preparation needs to set the basic roles and those should hold for the study at least mostly - if one role were to be replaced this hardly will be an issue but frequent replacement of key roles would be problematic. Selection of domain experts on the other hand will most likely need to be dynamic to some extent - depending on the items being analyzed. It is important to note that for the consolidation phases we would see it as mandatory that **all** involved staff be part of at least the review cycle. Consolidation phase has a profound impact on design and notably incorrect design (a high-level fault) has daunting impact on safety properties [33]. Preparation must ensure the availability

[33] IEC 61508 Ed2 always refers to "test and analysis" and we note here that for complex systems testing is not sufficient to uncover design faults thus the onus lies on the analysis.

of the necessary roles and domain expertise as well as the technical resources.

As a minimum access to a CMS with unique user s per study participant is mandatory. These unique IDs should be real-life names and no aliases. These names/IDs should be the same that are used for the review cycle for signoffs reviews and acknowledgement of change/fix requests. The CMS should allow for multiple branches to be coordinated effectively as subtopics commonly may be treated asynchronously (availability of domain experts) and thus the different groups of items should be separated in an administrative way so that reviews and acceptance (signoffs by study participants) can be harvested independently without blocking progress. The relevance of this is that temporal proximity of session discussions and reviews should be retained to ensure that the review cycle actually is effective. Other communication needs include dedicated (archived) mailing lists to allow backtracking discussions later (possibly many years later) thus we would recommend keeping the content format/encoding to an absolute minimum (e.g. 2822 - Internet Message Format - April 2001) using plain ASCII preferably.

To allow effective management of asynchronous review cycles, with changes to items being submitted as patches, [34] tool-support will be mandatory. Tools like Patchwork [**?**] or similar allows to manage relatively large numbers of patches that are in different states of review. The HD^3 work-flow allows a significant amount of asynchronous work - which is almost mandatory for high-complexity systems - but carries the risk of loosing oversight even in relatively small teams. Traceability (see below) is a key property that must be observed to allow justifying any conclusions drawn from item analysis, thus these seemingly trivial demands should not be underestimated. Note though that this section is based on quite limited experience and it can be expected to change in near future releases of this document.

Finally for the sessions them selves with multiple participants off-line tools are crucial - it is not realistic to have a large set of specialized engineer to convene frequently over longer periods of time - thus using off-line tools like IRC (our preference due to traceability and searchability) or phone in combination with web-based tools seems almost mandatory [35]. Our preference for IRC is due to the fact that IRC logs can be searched effectively (possibly also automatically post-processes) and allow for a good understanding of the analysis flow later with very little effort compared to face-to-face or phone meetings.

With respect to scheduling our current recommendation is to not analyze more than two items when using functional or security guidewords, or more than eight items when using concurrency guidewords, in a single session an item should take 60 to 90 minutes - functional/security and about 20-30 minutes for concurrency (average) - if adequately prepared). Time supervision seems to be effective (no consolidated scheme for that yet) No more than two sessions per day can be recommended. If one tries to force it this will favor mental cut&paste and not lead to effective analysis. Currently we think that a limit of 20 items per weak is a reasonable upper limit for a study team well into the process for a team with new members the limit should be significantly lower. The longest study done with HD^3 to date was 698 items, 111 SACs and 44 TODOs (Colominder Use-Case) thus extrapolating any scheduling data is too early.

[34] e.g. conforming to Open Group base-specification and/or GNU-diff/patch

[35] This may also raise some security issues with respect to communicating in ASCII test (both mail and IRC) - our mitigation currently is to protect such communication by appropriate higher-level technologies e.g. - but this is out of scope for this manual

13.1.3 Conducting study sessions

Each study starts asynchronously by one team-member preparing an item documentation (often times references to specific document sections or manuals) and sending it out to the study mailing lists as a recorded (traceable) input to the item analysis. All team members are to review the preparation information - a meticulous study lead will at least ask if all study members have prepared for session and if found that this is not adequate during review (typically this will be detectable during the attempt to agree on an interpretation) the session might need to be abandoned - an unprepared session is not only inefficient but it is ineffective with respect to safety.

If the prerequisites are satisfied

- Study preparation information was distributed;
- Preparation material was digested by study participants; and
- Critical roles are served -

then the study can commence with iterating over the set of guidewords for the item under study.

If SACs are found to be necessary the study team should review existing SACs for re-use (and if found that there are close-proximity SACs already defined a TODO for merging these SACs might be assigned). If technical decisions do not make progress or do not converge then they should be deferred to an assigned TODO. The item should then not be re-opened until the TODO is resolved.

Note that concurrency guidewords (see A.2) should **not** be done together with the functional guidewords (see A.1), probably separating functional and security would also make a lot of sense (notably as it may require different study team setup).

13.1.4 Managing databases and communication

.

HD^3 generates a number of data-bases which must be maintained in a traceable manner. The records play a key role in the re-use of pre-existing elements as they address the issue of intrinsic design faults which can not be addressed globally for a complex element but can be addressed point-wise once the full contextual detail has been developed. Thus the hierarchical data from the decomposition process serves the purpose of extracting the - system specific - context of the pre-existing elements so as to then allow a context aware analysis at the behavioral level. The consequence is that any modification to the data-base can have a profound impact on these assumption (context/scope/hazard-classification) and thus **must** be managed with respect to integrity (CM process) as well as with respect to correctness (or at least plausibility) thus reviews need to be managed. Management of reviews means that no change should be allowed in the data-base that was not the result of either:

- A analysis session with appropriate quorum
- A patch (patch-set) with adequate Acked-by/Reviewed-by from an adequate number of study participants whereby its not just a count issue - the role of the reviewing study participant should be taken into account depending on the nature of the change proposal.

A data-base without traceability and without tagging to allow counting of references and or severity and or impact breadth (e.g. how man items are affected by a SAC) would not allow establishing severity and more importantly would not allow maintaining safety attributes for even mildly dynamic elements. For complex pre-existing software elements with an expected high change/fix rate this is thus crucial.

13.2 System work-flow integration

Integration into system development, maintenance and retrofitting work-flow

13.2.1 Design recording

in HD^3 the design is driven by the hazard elimination and hazard mitigation not by the functionality - thus the design documentation can not be developed prior to the analysis but rather is developed during the analysis. This implies that the first design documents might be very high-level and not as extensive as typically used for a HAZOP style analysis.

The methods used for co-development of the design documentation is certainly not generic but specific to our know-how and personal preference thus there is no need to use the same methods - rather the methods should be mapped to the respective intent (see IEC 61508-7 Ed 2 as well as Annex C of IEC 61508-3 Ed 2). The design methods outlined here are thus to be read as examples - based on the Colominder Use-Case that is the basis for SIL2LinuxMP.

The analysis is layered from technology agnostic to vers specific technology aware levels. This roughly mapps to different design information being emitted - though this is **not** a strict mapping. The layers - introduced in section 5.2 - focus on different aspects of what we call HAZOP Allocation - a somewhat odd term as what actually is being allocated is hazard mitigations. This step allows seperating as well as splitting safety functions (e.g. checking exit conditions from one function aswell as entry conditions of the next function). The Allocation is currently being recorded in the PSpec or mini-specs of the identfied elements. Allocation takes place in all phases of the de-composition. Recordin allocation is also important for later impact analysis as the analysis item recordings are mapped into the PSpecs of the elements and are a first entry point for modification/impact-analysis.

HL: Technology agnostic analysis -> requirements specification (external)

UT: Unspecific technology aware analysis -> operationsl specification (relation of elements)

ST: Specific technology aware analysis -> design assumptions/constraints (elements -> API)

The intent is to build-up a sufficiently strict hierarchy so that reviews and impact analysis can focus on the appropriate level of abstration without getting into details on the upper layers or being "bothered" by high-level concepts at the lowest level. As noted this is not a strict requirement in any way - this really describes the change of focus from very generic to very specific. Accordingly the design representation will shift with progessing analysis and with that the individual methods - or rather the rank of methods.

- FSM example (see also semi-formal methods B.2.3 Semi-formal methods and finite state machines/state transition diagrams: see IEC 61508 B.2.3.2; For some details on how FSM are used in HD^3 see D.1

- intro and simple example,
- Mini-Spec or PSpec (Process Specification) are a further key element that is derived from the layered analysis flow with the focus on hazard elimination and mitigation. For a more detailed example of derived PSpecs in HD^3 see D.2
- Data-Dictionary example

[9]

13.2.2 Maintenance considerations

TODO:

- review needs,
- record-consistency checking automation (at least basic properties,
- impact analysis -> hierarchy

A Guidewords

A.1 Functional Guidewords

1. No - None of the design intent is achieved but also nothing else
2. Less - for quantifiable intent less indicates quantitative underachievement - e.g. intent "fill 10ml" but only 3ml are actually filled. The activity is the correct activity but not at the intended level.
3. More - the opposite of Less - a quantitative increase.
4. Part-of - for activities that may have multiple diserneable phases part-of incicates that some of the intent was achieved as specified and some was not. e.g. "Cloase Valve" might require three steps like set valve neutral state -> set valve new state, part-of might be that the neutral state was set but the ensuiing new state never reached. All action taken are intended but no all intended actions are taken.
5. Other-than - something else than intended occures - e.g. rather than setting the valve the motor is turned on.
6. As-well-as - the design intent is achieved & something else that was not intended also happens - e.g. heating of an element also heats a unrelated element.
7. Early - the action is executed too early while in the correct sequencial order - e.g. wait 10ms before signaling only waits 3ms.
8. Late - the opposite of early.
9. Before - sequence violation - e.g. the motor is turned on before the valve was opened.
10. After - sequence violation opposite of before.

A.2 Concurreny Guidewords

1. Interrupted - The correct action being executed is interrupted for some time with no ability to continue the controll of the action - e.g. a heater controller is interrupted and thus no longer able to change the heater setpoint due to an **unrelated concurrent** event - e.g. a priority inversion by one task blocks another task.
2. Terminated - The current action is terminated due to an **unrelated concurrent** action - e.g. a error in the processor forces the system to enter into a safe state asynchronously to the measurement process and thus the peripheral stays in a undefined intermediate state (it is not clealy shut down - which then may manifest it self as unexpeced behavior on the next restart).

A.3 Security Guidewords

These guidewords are 1:1 from Andreas Platscheks "A Harmonized Threat/Hazard Modeling Method for Safety Critical Industrial Systems" [**?**].

1. Spoof - Is it possible for an attacker to pose as a legitimate user, or let a device under his/her control act as if it were a legitimate device?
2. Elevate Privileges - Is it possible for a user with insufficient permissions to perform actions he/she does not have the necessary permissions for?
3. Listen - Can an attacker read/listen to sensitive in- formation?
4. Corrupt - Can an attacker manipulate the data in a non-systematic way (e.g. to random values the attacker does not have an insuffiuence on)? NOTE: This can be seen as a form of Denial of Service attack.
5. Tamper - Can an attacker manipulate data in a systematic way (to speciffic values the attacker does have an influence on)?
6. Denial of Service (DoS) - Can the attacker dis- turb communication or crash a hardware node? (Denial of Service Attack)
7. Malware - Is it possible for an attacker to intro- duce malware via this data-ow?
8. Wrong Connection - The attacker reconnects ca- bles/nodes in a wrong way, e.g. so that there is a cable from an untrusted network to a trusted port, used by the application. Could this wrong connection lead to an insecure situ- ation?

B (informative) Summary of procedures for HD^3

- HAZOP
- Design derivation
- Design decomposition
- Emitting of
  - Derived items
  - Safety Application Conditions
  - Selection Criteria
  - Traced TODOs
- Re-opening of items "from below"
- Re-opening of items "from above"
- Re-runing of concurrency analysis after sequencial analysis

B.1 HD^3 subprocess specifications

The semi-formal subprocess specifications outlined here (and still not completed !) are the process flows for those subprocesses briefly described in section 9.5.

B.1.1 HD^3 Study precondition checks

The Study preconditions that need to be satisfied before actually starting a HD^3 analysis section as outlined in 1.

1. System intent definition (see 10
2. Needed team competence available
3. Long-term commitment to organizations responsibility (initial intent definition to de-commissioning)
4. Sufficient time available (2 items per session is realistic)
5. Team members allocated for full analysis sessions planed
6. Configuration Management System (CMS) in place
7. Communication channels established and archiving initiated

B.1.2 HD^3 Study planing checks

The study planing includes all resources (see outline in) as well as tools. The checklist should be used to ensure that all resources are available and team-members know how to use them.

1. CMS tools allowing per-item file-based management (atleast diffable) preferably searchable (e.g. git [**?**]) available
2. Asynchronous communication channels agreed uppon in the team and all team members know how to use them in a consisten manner (e.g. mail format issues, file-format issues, log-message formats)
3. Artefact structure agreed and understood by all participants (location and tools usage)
4. Design methodology undertadnable/readable by all participants (symbols format and syntax - where applicable tools usage)
5. Study team members know how and where to raise concerns regarding tools, artefacts and formats. Special care needs to be given to language issues not only in multinational teams but also with respect to terminology if study participants are from different technical backgrounds.

Questions that arise during study regarding tools-usage, formats or terminology, etc. should be recorded in session logs - preferably in session logs and for technicaliteis in the item notes where the issue poped up first.

B.1.3 HD^3 Study session specification

Study sessions are team efforts and only can be conducted if a minimum set of study members is available. This is the process specificatoin for the step outlined in Section 9.5 Item 4. The individual specifications are not necessarily completly non-overlapping (which they ideally should be).

```
item
  origin: higher level item or initial design intent
  inputs: origin item, item specific documentation and expert opinion
  precondition: TODOs in origin item closed
  for guideword in GuideWords record
      for each guideword in context of Item do
          interpretation
          Causes
          for each cause in Causes do
               consequence
               for each indication/mitigation do
                   findings
                   SACs
                   selection criteria
                   check consistency with origin
                       cond: item reopening
                   TODOs
               done
          done
          Notes
      done
  done
  output:
     update tables
     update of design documentation
```

B.1.4 HD^3 Item review specification

Item reviews can be triggert by minor findings that do not mandate reopening an item (e.g. typos, or ambiguous wording) Item reviews can be off-line/asynchronous and are not a team effort. This step is outlined in Section 9.5 Item 5

```
item
  origin: completed study session on item
  trigger: findings (in session or during preparation)
  for finding in findings
       propose fix patch
       submit fix patch to list
  for fix patch do
       if Acked-by: CM apply
       if Reviewed-by: review -> CM apply
       if Nacked-by: discuss in team
           if accepted -> CM apply
           else OOPS: no record !
  output: update of design documentation
          mark dependent items for review
```

B.1.5 HD^3 Study Consolidation specification

This specification refers to the Study Consolidation step outlined in Section 9.5 Item 9

Once a study is considered completed with respect to a particular design intent it will need some consolidation since complex systems descriptions by humans result in some level of inconsistency or at least ambiguity as well as trivial defects overlooked (e.g. dead links). Resolution of any such issues in close temporal proximity of the session is necessary to ensure that issues can be resolved (6 month later nobody remembers enough detail for reliable closure)

```
Table
  origin: completed study session
  precondition: TODOs in all item closed
  trigger: unconditional
  for items in Table
    for SAC in SAC_tables
         review consistency
    for link in Tables
         check links
  if findings
    reopen item -> item review
  output: update of all Table
```

B.1.6 HD^3 Study Item Modification

This specification refers to the Study Modification step outlined in Section 9.5 Item 6 A modification can be necessary due to feature change (both adding and removing needs to be treated as modification ! [36])

[36] A dormant functionality may be actually originally contributing to diangostics and that would be silently lost of the feature were simply no longer used without re-opening the analysis

```
Table
  origin: completed study session
  precondition: modification request understood
  record: rational for modification
  for modified items
    for SAC in SAC_tables
         review consistency
    for link in Tables
         check link validity
  if findings
    reopen item -> item review
    record finding, changes and closing
  output: update of all Table
          update of design documentation
          mark dependent items for review
```

Note that there well may be a point where item-wise modification is no longer reasonable if the change request really is a new system in its own rights then this should not be attempted by modification. Determining this border is engineering judgement and can not be defined in a formal manner.

C (informative) Summary of HD^3 record formats

C.0.7 Session record

session ID : unique Id

date : of the session

duration : in minutes

item : the items name + HAZOP IDs

participants : in comma separated list

prepared by : Name of the person who prepared the session and sent out preparation material.

preparation sent : DATE (so everyone can find it in their mail-box or in the mailing list archive. If applicable a link to the respective message should be included.)

notes : you want to add – something did not work out in the session? did someone have to leave after half of the time? just note it here!

C.0.8 Item record

HAZOP ID Unique ID identifying the item + guideword combination.

Item The item under investigation (descriptive name or function/call)

Guideword (GW) The guideword that is applied on the item under investigation in this row of the table (see section A.1, **??**, A.3).

Interpretation The interpretation that was used for this item/GW combination.

NOTE: In Tables **??** *and* **??** *do not contain this column, because we only found out during these analysis that this column would add a great value to the analysis. The reason is simply that we often found out in the middle of discussing a item/GW combination that our interpretation of this combination was quite different. Thus we introduced this new column in order to first come to a common understanding of the interpretation and only then start the actual hazard analysis. Basically the interpretation explicitly states the hazard.*

Cause The cause(s) that lead to the hazard of this item/GW combination.

Consequence The consequence(s) that arise from this hazard.

Indication The indication(s) and/or mitigation(s) that are already in place or could be introduced to counter-act the hazard, referencing existing SACs with short description (repliated for readability) or introducing a new SAC if necessary, for each of the perceived consequences. For each consequence the classification should be included. Indication/mitigation should be used to review options to eliminate any detected hazards.

Question/Notes This contains open questions (most of them formulated as TODOs), and other notes. We also note newly created SACs that were emitted from this item/GW combination in this column. If item selection criteria (often constraints) are emitted they are added to the Selection table.

C.0.9 SAC record

SAC ID Unique ID identifying the SAC.

Description A cleaned-up version of the SAC description from the HAZOP table.

HAZOP ID The HAZOP ID in which the SAC was identified (multiple possible). This allows backward-traceability from SAC to where it was identified.

Direction The direction indicates whether the SAC has to be handled in the design of the use-case, or if it is assumed to be handled by another entity and thus is de-scoped for the use-case. There are only two possible values:

Down The SAC has to be handled by the use-case and it thus shall be reflected in the design.

Up The SAC is de-scoped because it is handled outside of the use-case. E.g. SAC ID 2 "Choose appropriate point to get sample" has to be handled during system installation. This is out of scope for our use-case and thus de-scoped. But will show up in the safety manual once installation and commissioning is covered (61508-1 7.9)

Rationale/Note Rationale for direction/de-scope and notes on how the SAC could be fulfilled in the design/implementation.

C.0.10 TODO record

ID : A unique ID for each TODO.

TODO : Description of the TODO itself.

Origin : The HAZOP Id where this TODO was identified.

Creation Date : The day on which this TODO was created.

Responsible Person : The person responsible for the resolution of this TODO.

Date of Resolution : The day on which this TODO was resolved.

Usage : An explanation of how the output produced by this TODO is to be used. The output can be e.g. code examples or a detailed description of the problem at hand.

NOTE: The output of TODOs is collected in the SIL2LinuxMP git repository in SIL2LinuxMP/src/HAZOP/ and an reference to the specific directory (labeled with the TODOs ID) shall be added to the Usage.

Conclusion : The conclusion to be drawn from the output of the TODO – what does this mean for our HAZOP, are the assumptions we made correct, or do we have to re-open a HAZOP item because the assumption was not correct.

C.0.11 Selection record

ID : a unique ID for each selection criteria.

Selection Criteria : A short description of the identified selection criteria.

HAZOP ID : The unique HAZOP ID of the item/GW combination in which the selection criteria was identified.

Rationale : A rationale on why this is in fact an important selection criteria.

C.0.12 Study preperation record

This is the only record that does not have a fixed format and it is not reasonable to set a fixed content either - effectively this is an engineering judgement of the study participant preparing the items for the study session. Thus we give the intent of preperation here not the content:

- Short description of the item
- Relevant related items and SACs
- Relevant design and specification documentation
- Any other information seen necessary to assist preperation (e.g. technical references for context)

```
Hi !

 so for today its pthread_cond_wait

 HAZOP prep pthread_cond_wait
   from Standby (UD_HAZOP ID 10-19) + DFD0
     Transitions:
       Standby-> Do_recalibration-> Do_measure
     context:
       on-demand measurement:
       Handle -> Standby: COMMAND was M_ON_DEM
       Standby -> Do_recalibrate: Command M_ON
     So we are waiting for communication in St
       that is where pthread_cond_wait comes f
  see doc/review/Use-Case/HAZOP_Allocation/DFD

 ref1: http://pubs.opengroup.org/onlinepubs/00
       pthread_cond_wait.html
 ref2: man pthread_cond_wait/pthread_cond_time

 Note the dependency on cancelation type !

 Essentially we will most likely be using the
 anyway - but technically that is not a must a
 supervision is in place. The argument for the
 would be a LOPA argument - given that we next
 recalibration a time failure is though not re

 Pleas look through the references and also th

thx!
```

An example of session preperation email:

Note that preperation information should be traceable - in our example it is an email that was archived in a dedicated mailing-list archive.

D (informative) Brief intro of design methods for HD^3

The design methods outlined here in the specific context of their use in HD^3 are no in depth introduction to the methods but focus on the specific role they play in the way we used them in the SIL2LinuxMP Use-Case Colliminder. It is conceivable, actually expected, that other developments will utilize other design representations - the role of design representation is though different in HD^3 as it primarily is the output of the hierarchical analysis refinement and input to the lower layers of analysis. The design process is thus an integral process of analysis and not vice-versa.

D.1 FSM

Finite state machines are a common way or recording stateful processing and are still relevant for structuring safety related functions respectively relevant for the analysis of safety related functions. The quite obvious limit of FSM is the non-consideration of concurrency. Never the less from our quite limited experience, FSMs seem to be a suitable recording of the initial designs sequential/temporal relation and play an important role in resolving the Before/After guide-word as well as a certain role in resolving "Part of" and "Other than".

The way FSMs are used in HD^3 is to record the sequential relation between functional elements identified, inserting new elements along the way if they emerge during analysis (e.g. initialization or rechecking needs). The FSMs emitted this way are living documents and relatively simplistic ASCII-text recordings were sufficient to capture the relevant nature of analysis scope. It is important to note that the design record is more related to the analysis scope than to the traditional design context. The implication of this is that versioning of the FSMs emitted is important including retaining the earlier versions that were the analysis basis. If these intermediate versions are not retained then understanding some of the analysis results in HD^3 might be hard or impossible.

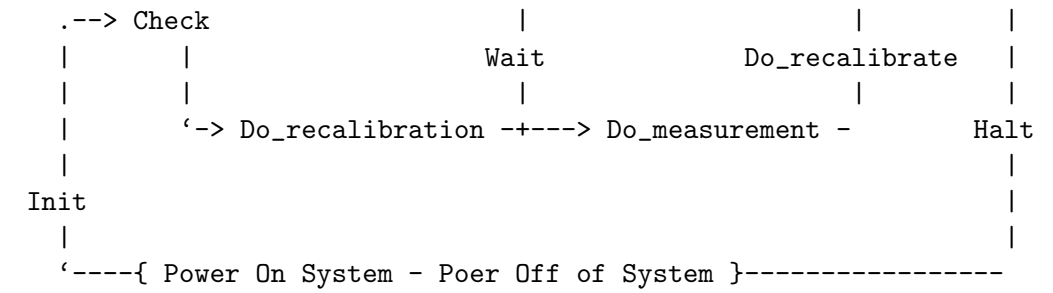
FSMs essentially record three things emitted from the design

- Elements - that is functionality that is being treated as a unit
- Transitions - elements to which a transition can take place
- Conditions - that must be satisfied to allow a transition to occur (or possibly that must be checked to verify the validity of the transition that took place)

While flow-control is a commonly used safety monitoring methodology to detect out-of-spec behavior, it seems that with adequate analysis a more course-grain FSM transition monitor might be sufficient in many cases provided the transitions are managed rigorously (source destination and conditions) - it is though too early to make a general claim of suitability - we simply do not yet have sufficient experience to make such a claim.

Example

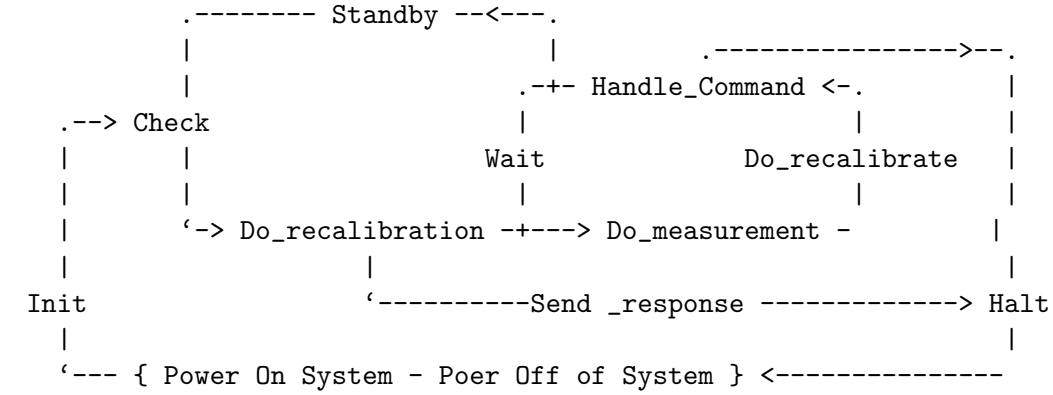
```
V2: Send response AFTER recalibration

          .-------- Standby --<---.
          |                       |         .--------------------.
          |                     .-+- Handle_Command <-.          |
```



2 modes of operation
 * periodic
 * on demand

During analysis the case of the system being idle too long was uncovered and a further mode of operation "single-shot re-calibration" was necessary to mitigate undetected drifting out of spec. This also emitted a high-level procedural SAC for the invocation of the re-calibration and resulted in adjustment of the FSM. Both versions of the FSM are retained in the design documentation for later reference.

V3: Send response AFTER recalibration and invoke "Send _response"
    alone if doing pure recalibration



3 modes of operation
 * single shot recalibration
 * periodic
 * on demand

State names used in the FSMs are also referenced in analysis reports and design related PSpecs. Arguments (in the above case only $_response$)$mustbemanagede.g.inadefinitionsectionsothatthemeaningisclearto all$

The structure of the FSM emitted is quite different than the FSM would be if a pure functional development would be undertaken. Even though this example is quite simple.

V4: Receiving would need to be earlier as currently

```
    the system would start measurement on power-on which is
    not resonable.
Rational:

    .-> Check -> Do_recalibration ---->-. .-> Do_recalibrate -> Send --.
    |                                   | |                  _response |
    |    .---------- Standby --<------. | |                            |
    |    |                            | | |.----------------->--.      |
    |    |                       .--- Handle <----------.       |      |
    |    |                       |                      |       |      |
    |    |                     Wait             Do_recalibrate  |      |
    |    |                       |                      |       |      |
    |    `---> Do_recalibration -+---> Do_measurement --       |      |
 Init_MC                                                       Halt <--
    |            |--- Safe-state ----.                          |
    |                                |                         |
    `-- { Power On System - Power Off of System } <-------------
```

The three (currently final) modes of operation are now specified as sequences

Single shot re-calibration: Init_MC-Check-Do_recalibration-Handle-Do_recalibrate-Send-Halt

Periodic measurement Init_MC-Check-Do_recalibrate-Handle-(Wait-Do_measurement-Do_recalibration)—Halt

On demand measurement Init_MC-Check-Do_recalibrate-Handle-(Standby-Do_recalibration-Do_measurement-Do_recalibration)—Halt

This FSMs transition is then elaborated on in a state transition table that carries further detail and serves as the common information base for the lower-layer analysis.

State transition table: $\mathrm{Init}_MC->Check_Status : All threads created and resources allocated Check-> Do_recalibrate : Env_OK, PV_self check, Sensor_self check Do_recalibrate-> Handle : unconditional Hanlde-> Standby : IDLE_MAX_TIMEOUT single shot recalibration : Handle-> Do_recalibrate : COMMAND was REC Send : unconditional(!) Send-> Halt : unconditional(!) periodic measurement : Handle-> Wait : COMMAND was_PERIODIC Wait-> Do_measurement : period expired Do_measurement-> Do_recalibration : unconditional(!) Do_recalibration-> Handle : unconditional(!) Handle-> Halt : COMMAND was_TERM on-demand measurement : Handle-> Standby : COMMAND was_ONE_DEMAN Do_recalibrate : Command_ONE_SHOT Do_recalibrate-> Do_measurement :! Do_recalibrate failed Do_measurem Do_recalibrate : unconditional(!) Do_recalibrate-> Handle : unconditional(!) Handle-> Halt : COMMAND was_TERM$

The terms used in these descriptions that are in capitals or with a leading _ are defined in a definition section of the design document e.g. COMMAND might not yet be fully specified at this point with respect to all possible commands the system might receive but for all commands that were found needed up to this analysis point the FSM is considere valid. If a new command were introduced in the system at a later point due to a further hazard uncovered then the FSM would be reviewed if the flow is still valid for the new command and if not a new FSM version would be emitted. Any

such change would mandate that assumptions previously made are reviewed and confirmed so as to preserve consistency of analysis and derived design documentation.

D.2

Process specification or mini-specs are related to the DFDs as well as to the FSMs - note that there is a certain redundancy in the design information which commonly is seen as undesired but we think that this is actually an important opportunity to uncover inconsistencies or incompleteness in team members perception. If the different design representations are in sync then we assume that there is a fair probability that the design in consistent as the individual attributes and aspects are defined from multiple viewpoints.

The PSpecs used in HD^3 are amended by the references to the SACs that were emitted pruning analysis as well as to analysis items that were conducted for the specific element. The PSpecs use a informal pseudocode notation noting functions and arguments that are defined in further PSpecs respectively in the definition sections and data dictionaries.

```
Init_MC:
========

 TODO once: Calibrate REACTION_SYSTEM - needs to be manually done !!

 Do_Init {
    Initialize RESOURCES

    Setup PERSISTENT_DATA

    Check STATUS

    while [Measure and (Record with ABSTIMESTAMP)](FLUORESCENCE_RESPONSE +
        TEMPERATURE + pH + TURBIDITY) at MEASURE_INTERVAL for
        INIT_VERIFY_TIME
    }

    Empty MEASUREMENT_CELL to WASTE

    Clean MEASUREMENT_CELL with [AQD and WL]

    Empty MEASUREMENT_CELL to WASTE

    Send _response
 }
```

This PSpec for INIT_MC is the final PSpec after the Technology aware unspecific phase of analysis. The individual functions are mapped to the respective analysis items and SACs. The INIT_MC

function is decomposed through the analysis of the individual sub-functions as emitted by SACs on the top functions or by a group of SACs (see 12).

The first example is a quite simple case where calling for a highly unspecific assurance on the initial calibration in the SAC.

```
Function Origin:
----------------
Origin: Calibrate REACTION_SYSTEM: Manual ??
  Highlevel HAZOP
        ID 80 (Re-)Calibrate Measurement System
             SAC ID 7: Initial Calibration done properly.
```

The second example is a more elaborate example of resource initialization that emerged during the analysis of initial unspecific technology aware (thus the prefix UT) items. Later technology aware specific (hence ST) items then extended the resource initialization design details and constraints.

```
Origin: Initialized RESOURCES
  Derived for allocation to SW
  UT_1042 Stir Probe - Less
          SAC ID UT_1015 SW shall conform with suitable systematic
                         capability
TODO: resource initialization details
      see ID ST 3026/29
      see SAC ID STS 3018
   -> Initialize IPLs
   -> IO Setup
   -> Create Files
   -> IPC Setup
   -> Finalize IPLs
   Note: Basic IO will be handled in Check (as the first
         action in init UNDER LOPA - we probably need
         to look at Early IO (read/write - BEFORE IPLs are
         active !)
   Note: IPC Setup must be partially done prior to Init_MC
         due to namespace seperation requirements !

TODO: shutdown needs to be defined !
```

The development of the hazard mitigations that called for a defined resource initialization - with defined properties - then is elaborated on while specific solutions are selected during decomposition leading to files IPC or IO needs. These specific resources then are further analyzed and derived in the form of SACs or selection criteria.

Note that the above examples are simply a snapshot from the SIL2LinuxMP Use-Case "Coliminder" and not yet completed (as indicated by the TODOs) but highlight the key point of HD^3 that

the design is driven from the hazard elimination and mitigation needs and not primarily by the functionality. This change of perspective leads to a design decomposition that may significantly differ from a pure functional decomposition.

Acronyms

ASCII American Standard Code for Information Interchange. 29

ATC Acceptance Test Criteria. 21

CCB Change Control Board. 30

CCF Common Cause Fault. 51–54

CMS Configuration Management System. 33

E/E/PE Electric/Electronic/Programmable Electronic. 9

FMEA Failure Mode and Effects Analysis. 32

FSM Finite State Machine. 60, 72

GNU GNU's Not UNIX!. 23

HAZOP Hazard and Operability Study. 7, 9, 12, 15, 29, 30, 42, 68

LOPA Layer of Protection Analysis. 51, 55

SAC Safety Application Condition. 17, 22, 30, 31, 47–50, 54, 68, 69

Glossary

bottom-half safety Functions in an OS/library (type B systems) that are in the sequence of a safety related function but where the safety related functions algorithm is not part of the OS/library function. These functions do not provide any safety functions on their own but only in conjunction with an invoking instance that is safety related.
Example: To provide a software watchdog function a software timer may be based on a POSIX timer. To initialize this timer a calls to library functions like timer_create()/timer_setttime() etc. might be used to handle OS level timer services. In such a case the timer service as well as the API to setup, configure, respond to and/or delete the timer are bottom-half safety functions. . 21

safety releated dependent function Any element safety function of SC/SIL ≥ 1 that depends on the generic functionality in question. The concept of safety related dependent funcions is only applied in the context of aggregating generic functions with element safety function but not with aggregation of element safety function - see IEC 61508-3 7.4.2.10/11 for the later case.
Example: A safety related control function that utilizes a OS or library function during startup or communication of results, but does not algorithmically depend on the OS or library function. . 52

References

IEC 61508:

ISO 26262: *Road vehicles – Functional safety – all parts*, Editon 1, IEC, 2011

IEC 62061: *Safety of machinery — Functional safety of safety-related electrical, electronic and programmable electronic control systems*, Edition 1, IEC, 2005

IEC 62882:

HSE, UK, *HSG 238 - Out of control - Why control systems go wrong and how to prevent failure*, HSE, 2003

Felix Redmill, Morris Chudleigh, James Catmur, *System Safety: HAZOP and Software HAZOP*, 1999

Andreas Platschek, *A Harmonized Threat/Hazard Modeling Method for Safety Critical Industrial Systems*, TU Wien, IEC, 2006

SIL2LinuxMP, *Coliminder HD^3 anylsis report* , HL-HAZOP-SIL2-OSADL, OSADL, 2017

Hassan Gomaa, *Software Design Methods for Concurrent and Real-Time Systems*, Addison Wesley, 1996