Formalisation and algorithmic approach to the automated driving validation problem

Jan Erik Stellet, Tino Brade, Alexander Poddey, Stefan Jesenski, Wolfgang Branz

***Abstract*— Automated driving road vehicles are to operate in an unstructured, public real-world environment. The openness of the operational design domain, the serious safety risk, the complexity of the system itself, as well as the regulatory situation pose a large challenge to the automotive industry. Thus, a strategy is necessary to ascertain the validity of such systems. An extensive formalisation of the problem and its root cause, the *deductive gap*, is provided in the authors' work [1] and described in a compact version in this paper.**

Thereto, the interdependent aspects *purpose*, *context* and *realisation* are detailed. This allows us to establish why deductive gaps between the *required*, the *specified* and the eventually *implemented behaviour* can occur. These gaps are caused by violations of underlying assumptions. Identifying such violated assumptions is the main goal of a novel algorithmic approach. Furthermore, the contributions and aspects left uncovered by normative regulations, i.e. ISO 26262 and ISO PAS 21448, are established.

I. Introduction

In the recent years, there has been a rush towards highly autonomous systems operating in public environments, such as automated driving of road vehicles, passenger shuttle systems and mobile robots. These systems, operating in unstructured, public real-world environments – the ***operational design domain*** can be characterized as ***open context*** – per se bear a serious safety risk.

Due diligence is necessary in development, release and even post release operation, which are all related to validation aspects. Otherwise, a *'winter of autonomous systems'* [2] might come down and the large investments taken, e.g. in the automotive industry, will not pay-off.

The **contribution** of this paper is a formal analysis of the fundamental challenges related to the valid design and operation of open context systems (Sec. II). Thereby, a simplified[1] representation of the authors' work [1] is provided. This analysis is followed by an investigation of normative regulations given by the ISO 26262 [3] as well as ISO PAS 21448 [4] in Sec. III. In order to address the crucial but yet unspecified aspect of aligning assumptions with validation activities, an algorithmic approach is proposed in Sec. IV.

The authors are with Robert Bosch GmbH, Corporate Research, Robert-Bosch-Campus 1, 71272 Renningen, Germany
firstname.lastname@de.bosch.com
www.bosch.com/research

[1]Sec. 2 in [1] contains a more detailed account on the notions of purpose, context and realisation as well as deductive gaps than provided in Sec. II-A to Sec. II-B of this paper. Furthermore, Sec. 3.1 of [1] presents an extended version of the 3-circles model, including a fourth circle related to assumption monitors, compared to the one given in Sec. II-C.

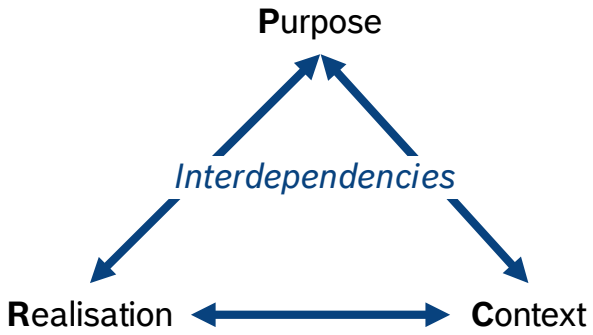


Fig. 1: The interdependence among the purpose, context and realisation is referred to as validation triangle [1].

II. Challenges for the Validation of Open Context Systems

Autonomous systems operating in public environments are usually designed to take over typical human tasks, e.g. driving road vehicles. The technical systems thereby do not only need to fulfil the tasks from a functional point of view. They also need to take over responsibility for safe operation and mitigation of hazardous situations, traditionally incurred by the human operator. This requires a significantly more extensive and reliable understanding of the context than e.g. in driver assistance systems.

A. The notion of validity

The complexity of the system and the context require several topics to be addressed in a systematic and holistic approach in order to achieve a ***valid system***. This means that a product bears no ***unreasonable risk***[2] to users and the society (e.g. safety and security risks), and, albeit subordinated, no unreasonable risk to the manufacturer (e.g. liability and costs). Note that the definition of unreasonable risk is closely related to the (societal) ***tolerated risk*** which is a complex topic and even a moving target.

B. The relation between purpose, context and realisation

In general, providing a valid solution to a given task is related to developing, designing or implementing a ***realisation R*** (e.g. a technical system) for a ***purpose P*** (e.g. automated driving on a highway), which needs to be fulfilled in a specific ***context C*** (e.g. on German highways at daytime).

[2]According to ISO 26262 [3]: *'Risk judged to be unacceptable in a certain context according to valid societal moral concepts.'*

The interdependent elements P, R, C are referred to as ***validation triangle***, see Fig. 1.

The fundamental challenge of validating complex open context systems is that none of the three vertices of the validation triangle can be expressed in a formally complete manner and the expression of one aspect even depends on the remaining two. This will be discussed per aspect in the following.

1) Purpose: The purpose is based on implicit expectations. As an example, consider the ambiguous expectations that on the one hand, an automated vehicle on a highway shall drive defensively, e.g. leave sufficient gaps for lane changes, but on the other hand show an agile driving style to not slow down the traffic flow. Such implicit expectations are referred to as the ***aimed purpose***.

The aimed purpose has to be transformed to explicitly expressed expectations which results in the ***intended purpose***. However, such explicit statements about a certain purpose depend on the related context of application and therefore most often are based on assumptions. More precisely, the customer or even the society might be able to express an expectation e.g. about appropriate behaviour of the system for a given, specific ***setting*** (related to a specific realisation in a specific context). However, it is non-trivial or rather impossible to identify all relevant settings. Therefore, explicit expectations are usually expressed quite abstract which leaves open room for quite different (setting specific) interpretations.

2) Context: The unstructured real-world operational design domain can be characterized as an open context. It bears infinitely many characteristics, possible interactions and effects (∞***-complexity***). Moreover, the context develops in time (it is evolving) with so far unseen characteristics and interactions appearing suddenly.

In addition, the necessary mapping of the ∞–complex context onto a reduced subset of an expected to be relevant context strongly depends on the related specific purpose and realisation. For example, if the realisation contains a video sensor, mainly the visual appearance but not the materials of the context elements are relevant whereas for a radar sensor, the opposite is true.

3) Realisation: The properties of complex systems are ***emergent***. I.e. the effective properties and behaviour of a realisation are the result of a complex interplay of the components ***(emergent characteristics*** and ***emergent behaviour)***. Therefore, analysing components is in general not sufficient to argue about the safety of the entire system [5].

C. Open context system behaviours

The previous section makes clear that developing complex systems for open contexts necessarily deals with simplified representations of the complex reality for the interdependent aspects of purpose, context and realisation. Given these inherently limited representations, one can differentiate between the ***required***, ***specified*** and ***implemented behaviour*** of an open context system. Here, *behaviour* is understood as interaction of the system with the environment, i.e. what is observable from the outside in contrast to internal state variables.

First, the required behaviour describes, from an omniscient perspective, what is the required behaviour in order to serve the purpose of the realisation in its context. In other words, this is the behaviour that we aim for in order to provide a valid and safe system.[3] Any deviation from the required behaviour is a failure.

Second, the term specified behaviour refers to what is explicitly understood and formalised, e.g. in the form of requirements, as a valid system. The specification is derived from an understanding of the purpose and context, i.e. the intended purpose and expected to be relevant context. It is the foundation for the design and implementation work.

Third, the implemented behaviour is what is actually achieved with the realisation. This might differ from the specified behaviour, e.g. due to emergent characteristics of the realisation as well as differences between the specified and the effective context.

The three kinds of behaviour and the possible relations between them are visualised in Fig. 2a. Except for the central set 3, all enumerated areas refer to distinctive deviations of the realisation fulfilling its purpose within its context. These areas will be later referred to for the question of how to achieve a valid system. Beforehand, further details on the ***deductive gap***, which is the main reason for imperfections to occur, will be given.

D. Deductive gap

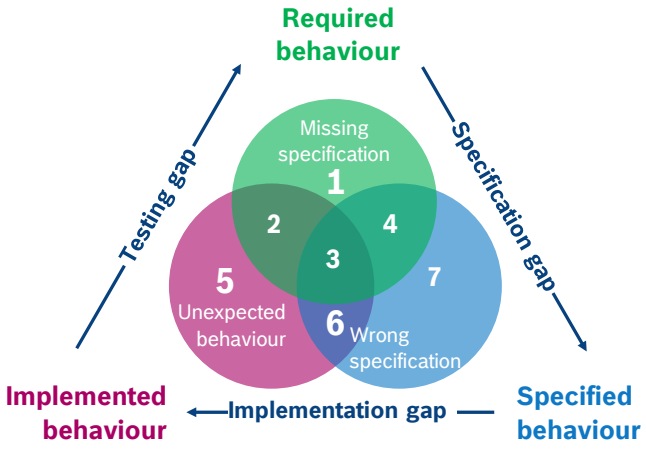
Building a model is a form of deduction. Different types of deductions appear all along the process from the initial formulation of a product idea to the implementation of a concrete product. An example from the top-most level would be specifying the intended purpose, the relevant aspects of the context and the characteristics of the realisation (e.g. as set of requirements and constraints).

The unavoidable deductions however are at the heart of the validation problem [6]. Every model building is necessarily based on explicit and most often even implicit assumptions. If these assumptions are (temporarily) invalid, the deduced representation will become invalid. This is referred to as ***deductive gap*** and ***insufficiency*** of the deduced representation.

One specific instance of a deductive gap is that the specified behaviour is different from the required behaviour (***specification gap***, see Fig. 2a). Violated assumptions about the purpose, i.e. when interpreting the aimed purpose, or about the context, e.g. the perception of objects by environment sensors, are typical causes for a specification gap.

Likewise, it is possible that the realisation shows an implemented behaviour that is different from what is specified (***implementation gap***). This might occur, e.g., because

[3]Note that the term *required* (another possibility would be *desired*) *behaviour* is intentionally chosen over *intended behaviour*. On the one hand, ISO 26262 defines a failure as the *'termination of the ability of an element to perform a function as required'*. On the other hand, the term *'intended behaviour'* already has a different meaning in ISO PAS 21448 which equals our definition of the specified behaviour, cf. Tab. I.



(a) Visualisation of the relations between required, specified and implemented behaviour.

Area	Description
1	Missing knowledge (specification) of required behaviour
2	Robust unspecified behaviour
3	Implementation behaves as specified as well as required (ideal)
4	Missing implementation of correct specification
5	Unexpected wrong behaviour
6	Wrong specification or technical limitation
7	Missing implementation of wrong specification

(b) Description of the areas in a.

Fig. 2: 3-circles model: The enumerated areas relate to qualitatively different parts of the validation challenge.

requirements have been implemented incorrectly, because the system is operated in a context different from the specified one or due to emergent characteristics of the realisation.

The ***validation challenge*** is closely related to the many necessary deductions applied during development:

- all underlying assumptions need to be made explicit and
- for each assumption, evidence needs to be provided for its validity.

In order to generate evidence, hypotheses need to be formulated and checked, e.g. by formal approaches, simulation and real world observation. Note that hypothesis checking based on (implicit) models needs to be done carefully, as applying the same approximations in the derivation and interpretation of hypothesis checks might lead to a self-fulfilling prophecy and hence to a wrongly passed test, providing an unfounded argument [9].

In the following, two normative regulations applying to safety of driver assistance systems will be analysed with regards to specification and implementation gaps.

III. Normative Regulations

There are currently two norms with regard to safety-related E/E systems in series production vehicles. ISO 26262 [3] addresses *'possible hazards caused by malfunctioning behaviour'* but excludes the *'nominal performance'* of said systems. In addition to this, the scope of ISO PAS 21448 [4] includes *'hazards resulting from functional insufficiencies of the intended functionality or by reasonably foreseeable misuse by persons'*.

A. ISO 26262

ISO 26262 provides a comprehensive definition of a *'failure [being] the termination of the ability of an element, to perform a function as required'*. Since *'function as required'* can be mapped to our definition of required behaviour (see Tab. I), this comprises all gaps in Fig. 2a.

However, the scope of the standard excludes the *'nominal performance of E/E systems'*. Instead, it focusses on *'malfunctioning behaviour'*, i.e. *'a failure or unintended behaviour of an item with respect to its design intent'*. To ensure the integrity of the implementation, ISO 26262 imposes requirements for the development (cf. ASIL) and for the control of random hardware faults.

As already mentioned above, ISO 26262 does not address the question of the necessary nominal performance in order to be considered as safe. Using the model from Fig. 2a this question can be formulated as

- Under which circumstances (due to nominal performance issues) does the implemented behaviour not correspond with the required behaviour (areas 5 and 6)?
- Is this discrepancy acceptable?

Therefore, *'for some systems, which rely on sensing the external or internal environment, there can be potentially hazardous behaviour'* [4] although the system is not affected by faults in the scope of ISO 26262.

The challenges in applying ISO 26262 to driver assistance systems, e.g. automatic emergency braking, have been pointed out in [6], [7]. One central aspect concerns that in a layered development approach (e.g. V-Model), each layer of abstraction refines the specification of the previous one (i.e. from top level safety goals down to the implementation). Thereby, it is implicitly assumed that the deductive step from one layer to the next is correct. However, if this deduction depends on assumptions that can be violated, a gap in the refinement chain occurs. Although the implemented modules and systems can be verified against their respective specification, it depends on the correctness of the deductive steps, whether the safety goals are achieved.

Additionally, ISO 26262 assumes a given *'item definition'* which makes the *'requirements of the item as well as the dependencies between the item and its environment'* explicit. From the discussion in Sec. II-B it becomes clear that

TABLE I: Mapping of terminology in ISO 26262 and ISO PAS 21448 (SOTIF) to the model from Sec. II.

	3-circles model (Fig. 2a)	**ISO 26262**	**ISO PAS 21448**
Terminology:	Required behaviour	*'function as required'*	–
	Specified behaviour	*'intended functionality'*	*'intended functionality'*, *'intended behaviour'*
	Implemented behaviour	–	*'implementation of the intended functionality '*

achieving this with sufficient completeness is challenging in open contexts.

B. ISO PAS 21448 (SOTIF)

ISO PAS 21448 – commonly referred to as SOTIF (*'Safety of the Intended Functionality'*) – *'addresses intended functionality where proper situational awareness is critical to safety, and where that situational awareness is derived from complex sensors and processing algorithms'*, e.g. in automatic emergency brake systems[4].

As such it supplements the scope of ISO 26262 to *'functional insufficiencies of the intended functionality or from reasonably foreseeable misuse by persons'* mainly for automated driving functions which are affected by the open world issue. These functional insufficiencies can be either caused by performance limitations (i.e. functional insufficiencies of the implementation itself) or by an incorrect specified behaviour, corresponding to the areas of 1, 5, 6 and 7 in Fig. 2a, the specification and the implementation gap. Thus, ISO PAS 21448 in combination with ISO 26262 can address all critical areas in Fig. 2a for these systems.

ISO PAS 21448 stipulates to exploit understanding about the system and the context in order to *'identify the system weaknesses (including those of its sensors, algorithms, actuators) and the related scenarios'*. To this end, the notion of *'triggering events'* is introduced, which describe conditions of a driving scenario that may trigger a system response potentially causing a hazardous event.

Triggering events constitute instantiations of the ∞–complex context. Thus, identifying them without proper use of system understanding is a hopeless endeavour. ISO PAS 21448 points out that *'the analysis can be supported by inductive and/or deductive methods'*, which is precisely what shall be substantiated with the algorithmic approach in the following section.

IV. Algorithmic approach to align assumptions with validation activities

Although both ISO 26262 and ISO PAS 21448 call for the consideration of assumptions, they do not regulate how to exploit this knowledge for the detection of deductive gaps. For that reason, a novel algorithmic approach (algorithm 1) is provided in order to align the knowledge available about assumptions with validation activities.[5]

[4]Even though the scope of ISO PAS 21448 is limited to L1 and L2 driver assistance systems [12] it needs to be taken into consideration for true automated driving systems (SAE L3-L5).

[5]The relation of algorithm 1 to the algorithms given in Sec. 3.2 of our previous work [1] is as follows: The first two steps given in Sec. IV-A and Sec. IV-B can be found with more details in [1]. The subsequent steps from Sec. IV-C and Sec. IV-D are novel extensions of the scope of [1].

The algorithm operates upon a high-level description of the purpose and the context, which are considered to be given. To obtain a high-level description for the context, e.g. the layered model described in [11] can be used as a starting point. Obviously, these descriptions are rather abstract but this is seen as a key for a description that holds in case of an open context. Furthermore, a too restrictive high-level description limits our capabilities to identify deductive gaps.

Once the input is provided, the algorithm investigates within four steps the chances for a deductive gap due to assumptions, which are made during the refinement of the purpose as well as context description and its realisation. Note that although the first two steps merely refer to external activities and therefore provide no active contribution, they are necessary since a failing in the third step requires us to go back and revise them accordingly.

A. First Step – Refinement

The first step aims for a refinement of the purpose and context description where the strong dependency between them makes a crosswise concretisation necessary, cf. [1] for details. To attain this, a top-down analysis of data-bases [8], use case catalogues, observed failings of traffic users as well as the systematic generation of obstacles [10] seems beneficial.

While refining descriptions, care must be taken to note the underlying assumptions. On the one hand, assumptions narrow the context under which the description holds but, on the other hand, they provide an explanation of why the refined descriptions can dominate the variety of the required behaviour.

B. Second Step – Realisation

After the descriptions of purpose and context have been refined, the second step focuses on the realisation. The realisation entails a composition of functionalities whose behaviour must comply with the refined purpose and context. A compliant realisation is found if verification activities have shown that the specified behaviour is part of the implemented behaviour. Here, verification is understood as an activity that states the compliance ($|=$) of something given with something specified. Different granularities exist for this compliance check, which range from review and consistency checks up to a formal proof.

It must be clear though that the verification is only conclusive up to the degree to which the required is made explicit. Hence, verifying that the specified behaviour is part of the implemented behaviour does not imply that the two are equal. Therefore, unwanted implemented yet unspecified

Algorithm 1: Algorithm that aligns assumptions with validation activities.

Data: $P_1 \& C_1$ as a description of the purpose and context, respectively
Result: R as a realisation w.r.t the refined set of $P \& C$

1 `Refine` $P = P_1$ OR $C = C_1$ and note the assumptions (A) under which the refinement holds

2 `Provide` a realisation R for $P \& C$ and note the assumptions (A) being made so that $R\,(C) \mid\!= P$
if *no realisation can be found* **then**
  if *no refinement was possible* **then**
    discard the refinements being made
  end
  continue with 1 to seek for a realizable refinement of $P \& C$
end

3 `Investigate` the assumptions (A) by
(a) Checking their consistency
  if *not consistent* **then** discard the inconsistent part of R and continue with 2;
(b) Exploring conditions that can contradict them, as follows:
  for $i \leftarrow 1$ *to* $\|A\|$ **do**
    (i) Explore conditions (Cd) so that $Cd \rightarrow \neg A_i$
    (ii) Associate Cd with C
      if *not possible* **then** continue with 1 to refine the context;
    (iii) Check whether another assumption A_j can compensate for $\neg A_i$,
      if *not possible* **then** discard the realisation where $\neg A_i$ is involved and continue with 2;
      else Note assumptions under which another assumption A_j prevails against $\neg A_i$;
  end
(c) Showing that the instances of A are
  (i) Repeatable under various C so that $(R\,(C) \mid A) \mid\!= P$
    if *not possible* **then** continue with 2 and discover hidden assumptions;
  (ii) Unambiguous under various C so that $(R\,(C) \mid\!\neg A) \mid\!= \neg P$
    if *not possible* **then** continue with 2 and discover hidden assumptions;

4 `Provide` a validation strategy to explore possibilities for $R\,(C \vee C_1) \rightarrow \neg P_1$
(a) Define reject criteria
  (i) Formulate hypotheses on $A \& P$ and define respective monitors
  (ii) Show effectiveness of monitors on hypotheses, as defined in 3(c)
    if *not possible* **then** redo 4(a)(i);
(b) Define conditions for the strategy in order to
  (i) Confirm the known - especially the results from 2 and 3(c)(i)
  (ii) Explore the unknown
  (iii) Formulate stopping criteria w.r.t 4(b)(i)&(ii)
(c) Conduct the strategy until stopping criteria is satisfied

(area 5) behaviour or required behaviour that is missing in the specification (area 1) are not covered.

In order to deal with these additional sets, the assumptions need to be noted under which the realisation can serve the purpose while being exposed to the specified context. Recalling that the assumptions noted in the first step describe conditions about the requirements, the assumptions noted in the second step indicate necessary conditions for the operationalisation of these requirements, which finally allows us to investigate assumptions, as described next.

C. Third Step – Investigation of Assumptions

The third step investigates the assumptions resulting from the first two steps in order to locate where deductions are made on the basis of invalid assumptions, which can cause a deductive gap.

First, the consistency of assumptions must be checked (see 3(a) in algorithm 1) since inconsistent assumptions indicate that the realisation will certainly be unable to behave as described. The difficulty of this is not to find contradictions between assumptions but rather to ascertain semantic mismatches that mask included inconsistencies. It goes without saying that inconsistencies require us to discard the part of the realisation where a conflicting assumption is involved, which sets us back to the second step so that we can redesign or revise the conflicts.

Second, given that the assumptions are free from contradictions, step 3(b) seeks for conditions by which assumptions

become invalid. To this end, assumptions are systematically turned into something invalid in step 3(b)(i), e.g. by adding specific prefixes as known from HAZOP [10]. Next, possibilities must be explored so that these conditions appear within the presumed operational context. If no such condition can be found, it is recommended to start again with the first step in order to refine the context. Note that the refinement is not focused on functional aspects driven by the purpose but rather on conditions where the assumption may become invalid.

Even if this refinement is exhausted, conditions outside of the operational context should be considered in order to show that the realisation cannot encounter them. In principle, this is the objective of step 3(b)(iii) where additional assumptions must be found that can compensate for the invalid assumption. If no such assumption can be found, a deductive gap is found since conditions are known by which an assumption becomes invalid. This requires the realisation to be revised. In cases where the realisation can cover for the loss of an assumption by another, the conditions must be stated by which one assumption prevails against another. On the one hand, these conditions provide the basis for a rational of why the considered deductive gap cannot occur but, on the other hand, they involve further assumptions, which require us to go through the steps of 3(b) as long as further assumptions appear.

Third, after steps 3(a)&(b) show the absence of deductive gaps with respect to the available knowledge, step 3(c) deals with the question whether the alleged understanding about assumptions is correct. Therefore, it is proposed to probe the cause-effect relationship where the realisation in a specific context is considered the cause whereas its behaviour is comprehended as an effect that becomes visible under certain assumptions.

It stands to reason that a disagreement among a cause, an assumption and an effect indicates a deductive gap. To find such disagreements, it must be shown that the cause-effect relationship is repeatable but not ambiguous (compare step 3(c) of algorithm 1).

Repeatability (step 3(c)(i)) means to demonstrate that the purpose (P) is met under various contexts. Its peculiarity is that the context must be set so that a specific assumption becomes necessary, which is a substantial difference compared to step 2 where ($R(C) \models P$) is solely focused on the purpose.

The next step 3(c)(ii) checks whether the relationship is unambiguous, which presupposes that the purpose cannot be satisfied under invalid assumptions. Unlike step 3(c)(i), demonstrating the unambiguity requires some kind of failure injection in addition to the use of the right context so that the system collapses in consequence of invalid assumptions. If we are unable to show that our understanding about the known assumptions is repeatable and unambiguous, hidden assumptions regarding the realisation must be discovered, which requires to continue with step 2 of algorithm 1.

Further knowledge about assumptions inevitably enhances our understanding in that conditions affecting an assumption can better be associated with a specific context (compare step 3(b)(ii)). In addition, more knowledge about assumptions facilitates the identification of assumptions that stand in for others (see step 3(b)(iii)).

D. Fourth Step – Validation

First of all, we must define reject criteria (step 4(a)) by which we can take notice that the realisation is invalid. Since the challenge here is to bridge the gap between an abstract and a concrete representation, we therefore merely formulate a hypothesis about indications for a violation of the described purpose or its respective assumptions.

In addition, a monitor is required to operationalise the formulated hypothesis so that a violation of the specified purpose as well as known assumptions can be observed during the operation of the realisation (step 4(a)(i)). To make sure that both the hypothesis and its monitor are effective with respect to what is known, it is important to show that both provide repeatable and unambiguous results, as described in step 3(c).

With these monitors at hand, a validation strategy should be conducted in order to show that nothing contradicts our monitored expectations and that we are unable to provoke such contradictions. The former aspect (step 4(b)(i)) aims to confirm the known in that, even if assumptions are left unknown, it does not matter. This implies the definition of targets in order to expose the realisation to a defined set of context elements. The latter aspect (step 4(b)(ii)) is to explore the unknown by running into a context that is not yet tested. Since unknowns cannot actively be discovered, we must probe the realisation within the randomness of the context.

Regardless of the strategy used, stopping rules have to be defined (step 4(b)(iii)). However, can we consider a realisation as valid ($R\,(C \vee C_1) \rightarrow P_1$) once the stopping criteria are satisfied (step 4(c))? The satisfaction of stopping rules usually indicates the absence of risks beyond a certain threshold (residual risk) which shall be lower than the unreasonable risk. However, since we are lacking future knowledge about the validation triangle – due to unforeseeable changes to e.g. societal expectations and the context – the notion of *sufficiently valid* is preferred, which accepts the need for ongoing observation.

Ongoing observation refers to the monitoring of hypotheses, as described in step 4(a)(i), after the realisation has been deployed. A detailed account on this monitoring is provided in [1]. This yields an extended version of Fig. 2a with additional areas.

E. Aspects to consider when applying the algorithm

The implementation of algorithm 1 poses challenges on how to master the variety of traces between the refined purpose and context descriptions. With these traces at hand, it becomes clear which parts must be revised if a deductive gap is detected. Besides the handling of failings where the algorithm requires going back to a previous step, the handling of traces is key to identify deductive gaps when switching

from one purpose to another. To this end, one must be able to search for specific elements of context that implies a purpose so that the steps 3 (b & c) and 4 (b)(i) become possible. For managing these traces, the use of ontologies is advisable because the related description elements facilitate to reason about hidden assumptions. In addition, it is worth considering a structured process in order to note the assumptions when designing a realisation (cf. step 2).

V. Conclusion

This work provided an analysis of challenges that affect the validity of complex systems operating in an open context, such as self-driving vehicles. Since an open context features infinite characteristics, limitations in the model of reality being used to derive design decisions will thus be inevitable.

Due to this, the role of assumptions has been elaborated on with a particular focus on causes for invalidity and the effects. Invalid assumptions have been identified as causes for a specification or implementation gaps. Since these gaps are the underlying reason for an invalid system, normative regulations given by the ISO 26262 and ISO PAS 21448 (SOTIF) have been analysed. It turned out that the combination of both standards can deal with the specification and implementation gaps but leaves the role of assumptions unregulated.

To this end, an algorithm to align the knowledge about assumptions with validation activities has been proposed. Nonetheless, the proposed alignment is limited to known assumptions. The exploration of unknowns depends on the applied validation strategy. However, the infinite characteristics of an open context – especially considering potential future changes – prevent applying a formal criterion of completeness. For this reason, we argue for an ongoing observation in order to have at least a chance to detect unknowns.

References

[1] Alexander Poddey, Tino Brade, Jan Erik Stellet, and Wolfgang Branz: "On the validation of complex systems operating in open contexts." arXiv preprint arXiv:1902.10517, 2019.

[2] Shai Shalev-Shwartz, Shaked Shammah and Amnon Shashua: "On a formal model of safe and scalable self-driving cars." arXiv preprint arXiv:1708.06374, 2017.

[3] International Organization for Standardization, "Road vehicles – Functional safety,", ISO 26262, 2011.

[4] International Organization for Standardization, "Road vehicles – Safety of the intended functionality," ISO PAS 21448, 2019.

[5] Yrvann Emzivat: "Safety System Architecture for the Design of Dependable and Adaptable Autonomous Vehicles." PhD thesis, Centrale Nantes, 2018.

[6] Bernd Spanfelner, Detlev Richter, Susanne Ebel, Ulf Wilhelm, Wolfgang Branz and Carsten Patz: "Challenges in applying the ISO 26262 for driver assistance systems." Tagung Fahrerassistenz, München, 2012.

[7] Ulf Wilhelm, Susanne Ebel and Alexander Weitzel: "Functional Safety of Driver Assistance Systems and ISO 26262", in *Handbook of Driver Assistance Systems: Basic Information, Components and Systems for Active Safety and Comfort*, pp. 109–131, Springer International Publishing, 2016.

[8] PEGASUS project, https://www.pegasusprojekt.de/en/.

[9] Tim Kelly and Rob Weaver. "The goal structuring notation – a safety argument notation." Proceedings of the dependable systems and networks 2004 workshop on assurance cases, 2004.

[10] Brian Tyler and Frank Crawley. HAZOP: Guide to Best Practice. Elsevier, 2015

[11] Fabian Schuldt, *Ein Beitrag für den methodischen Test von automatisierten Fahrfunktionen mit Hilfe von virtuellen Umgebungen*. PhD thesis, Technische Universität Braunschweig, 2017.

[12] SAE International, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," SAE Standard J3016 JUN2018, 2018.