Kernel-based ARchitecture for safetY-critical cONtrol

KARYON

FP7-288195

D2.5 - Definition of failure modes and failure semantics

Work Package	WP2		
Due Date	M24	Submission Date	2013-12-03
Main Author(s)	Jörg Kaiser (OVGU) Tino Brade (OVGU) Sebastian Zug (OVGU)		
Contributors	António Casimiro (FCUL) Rolf Johansson (SP)		
Version	1.2	Status	Final
Dissemination Level	Public	Nature	Report
Keywords	Failure modes, sensor-based systems, data validity		

Part of the Seventh

Framework Programme

Funded by the EC - DG INFSO

Version history

Rev	Date	Author	Comments
V0.1	2013-08-07	Tino Brade (OVGU)	Basic structure and first working copy
V0.2	2013-09-04	Tino Brade (OVGU)	Add section “Validity representation”
V0.3	2013-09-06	Tino Brade (OVGU)	Add section “Relation between validity and ASIL”
	2013-09-26	Rolf Johansson	Adding review comments
V0.5	2013-11-22	Tino Brade (OVGU)	Second review
V0.8	2013-11-29	Tino Brade (OVGU)	Adding failure algebra
V1.0	2013-12-02	Tino Brade (OVGU)	Releasing final version
V1.1	2013-12-03	Tino Brade (OVGU)	Fixing references
V1.2	2013-12-03	António Casimiro (FFCUL)	Final review and submission

Glossary of Acronyms

Dx.y	Deliverable belonging to work package x, with serial number y
FMEA	Fault Modelling an Effect Analysis
IV	Impact vector
KARYON	Kernel-based ARchitecture for safetY-critical cONtrol
OV	Occurrence vector
RD	Detection result
RF	Filter result
RPN	Risk Priority Number
S	Selection vector
TD	Detection transition matrix
TF	Filter transition matrix
WPx	Work Package with serial number x

Executive Summary

The KARYON project (Kernel-based ARchitecture for safetY-critical cONtrol) focuses on the predictable and safe coordination of smart vehicles that autonomously cooperate and interact in an open and inherently uncertain environment. The main objectives of WP2 are the definition the KARYON safety architecture, providing the guiding principles on how to structure a safe system in relation to assumed system and fault models. The primary motivation for expending particular effort on the analysis and abstraction of failure modes of system components originates from the distribution, mobility and complexity of the control systems envisaged in KARYON. We argue that this firstly requires fault models that abstract from the subtle and diverse behaviours of faulty components and provide a well-defined failure semantics at the component's interface. This allows defining a control algorithm without the detailed knowledge of individual component failures. The basic idea is to derive a validity of the information by analysing and classifying the failure modes of components.

In the previous report D2.2 the focus was on the development of a failure model and a failure semantics for sensor components. We introduced sensor related failure modes and explored basic approaches to estimate the validity of sensor data. The data output of a component was complemented by an output providing a validity estimate. Thus, the multiple sophisticated ways a sensor may fail was mapped to a validity, keeping the knowledge of failures and how to handle them close to the sensor component. Because of the uniform representation of sensor data it was possible to exploit remote sources of perception without a detailed, intimate knowledge of their internal properties.

This report carries further the idea of an assessment of data and deals with the problem how validity is affected by typical computational elements of a control system like failure detectors and filters. We strive for answering the question how we can derive the validity of data when it passes through such computational elements. In this report we refine the notion of validity and present what we call "failure algebra" for a sequential structure of the computational chain. The final goal of the investigations is the validity assessment of complex systems that include redundancy and fusion. A second objective is to highlight how validities are related to safety requirements of a function. See further D4.1 for elaboration of safety requirement formalism and a preliminary relation to validity attributes. Validities can help in many ways. The proposed failure algebra allows an analytic derivation of the bounds for the validity. These bounds can be checked against the requirements of an application function. The requirements are considered to be the result of a hazard analysis and associated with an ASIL. Therefore, the designer may detect at an early stage of the design that the system will not fulfil its safety requirements. So far, the compliance with a certain ASIL has to be proved according the combination of process arguments and product arguments provided in ISO26262. If a test fails, it is difficult to estimate which component was responsible for the malfunction because of the inherent reachability and observability reasons. The way in which we calculate the validity will uncover the influence of failure types through the computational components and provides hints on the weak parts of the system. It eases the problem, which parameters have to be improved to finally meet the needs of the application and thus complements testing.

At run-time, dynamic validities are generated. These values are the results of the various detection and filter components along the processing chain and map them to values. These values allow the application to assess the actual data that is provided.

Because the notion of validity is one of the central concepts, we will discuss this in more detail. A validity is the outcome of a multidimensional mapping that takes a failure model, assumptions about its coverage and the capabilities of detection mechanisms as an input and maps it to a value. This value is an indication of how much confidence can be put in the associated sensor value. Validity has a static, design-time and a run-time property. The design-time property defines the bounds in which a generated validity may be expected and is

dependent e.g. on the quality of a sensor, the detection and the filter mechanisms. The run-time validity is the actual outcome of the check and filter components. In the previous report, these aspects have been treated as two separate values. The application was able to assess the dynamic validity value in the light of what we called system validity. In this report, we try to present a way of condensing the system validity and the run-time results into a single value that reflects the actual value (indicating the confidence in the sensor data) *and* the confidence that we can have in this value.

Table of Contents

List of Figures

List of Tables

1. Introduction

One of the basic concepts of KARYON is the departure from a fully static design time safety assessment of the overall complex control system. KARYON aims to balance the performance-safety trade-off by providing multiple levels of service each of which satisfies the full safety needs but at different performance levels. Moving safety assessment from design time to run time allows for relaxing overly restrictive assumptions about the required integrity status of the overall complex control system. However, it will require the continuous monitoring and evaluation of the system integrity during run time as a basis to decide which functional level is possible without sacrificing safety demands.

In KARYON, we investigate an approach that keeps the integrity assessment close to clearly separated and identifiable functional components. For every component we propose to complement the nominal value generated by a component or by an additional output that provides an integrity attribute. This attribute represents a measure for the integrity of the respective signal produced by a component. We call this the validity of a signal. The overall structure has been briefly described in D2.2 and D4.1 previously. Figure 1 depicts such a component.

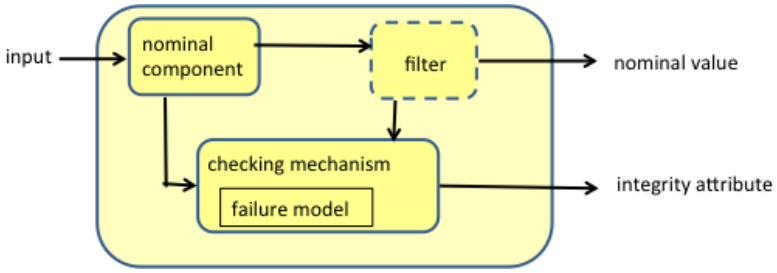


Figure 1: Estimating the validity by a checking mechanism

We assume that a component comprises the acquisition and computational components from a sensor that captures a real-world entity to the component that outputs an application relevant nominal value together with a validity attribute that enables the safety assessment at run time.

To generate the integrity attribute, the component needs a self-assessment mechanism. The checking box in Figure 1 represents this capability. It is based on the failure model that was introduced in [D2.2], [1]. It will be further discussed in Section 2.1 in this report. The checking mechanism detects a failure without affecting the respective signal. It modifies the integrity attribute only. For mitigating or masking the effect of a failure a value may pass a filter. The filter is a general abstraction of a component that handles the effect of a failure. Typically a detector and a filter are integrated in a fault-tolerance mechanism. We separate the aspects of awareness and treatment because these are different concerns and the separation allows for more freedom of design, e.g. omitting a filter completely in the component and handling the failure in a subsequent stage.

The run time assessment mechanisms are based on the specification and quantification of design time assumptions. During design time an engineer has to answer questions like "which failure types are affecting the components and what is their impact?", "How are these failures detected and how good the detection mechanisms need to be?" and "How is the data conditioned and filtered to compensate the effects of failures?". Taking reasonable assumptions is a very important and difficult task of an engineer because they depend on a good estimation of system properties and include substantial experience and empirical knowledge. Based on these assumptions the engineer decides on the mechanisms for adjusting the quality of the component's outgoing data at run-time to the integrity requirements. In our approach, these engineering assumptions are quantified in a failure model, a quantification of the detection

capabilities and the filter characteristics to allow a comprehensible assessment. This is particularly beneficial when such a components is used in a larger setting or will be reused in another design. As an example we examine a typical component where the input data is provided by a sensor.

We distinguish two flows of information in Figure 2. The lower part generates the nominal data output while the upper part is devoted to the calculation of the validity attribute. In this report we will focus on the definition and transformation of the parameters defining the validity.

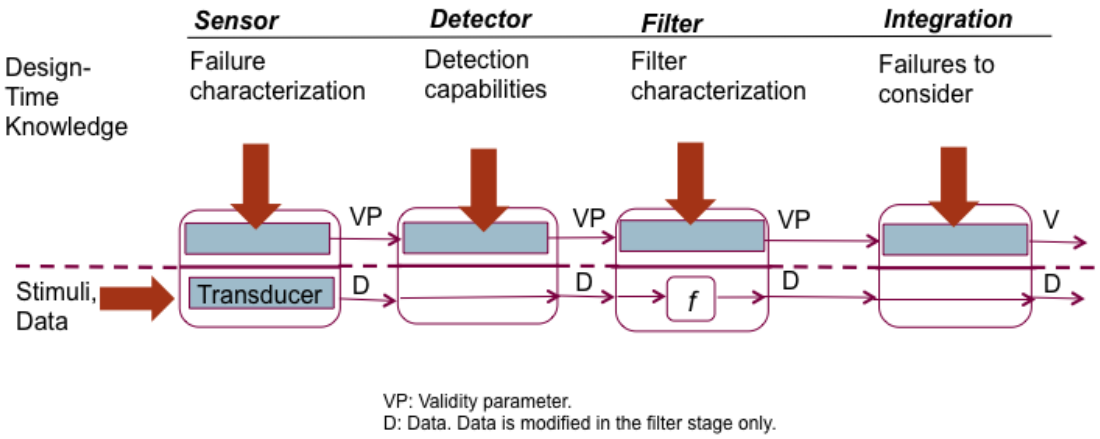


Figure 2: Computational chain for a sensor processing component

We are considering a data centric failure model [D2.2], [2] which specifies failures in terms of how they affect the data e.g. according to an anticipated signal behaviour. The starting point is the identification of relevant sensor failures. Sensors deliver continuous values that may be affected by many subtle and sensor specific failures. A detailed discussion about sensor failure modes is given in [1]. An assessment vector quantifies for each failure type a number that characterizes the anticipated severity and the occurrence probability of this failure. This is comparable to a risk priority number in the FMEA scheme [3]. Different from FMEA, we provide a scheme that allows combining multiple failure types and using this for run time assessment, while FMEA maps multiple failure types to the static worst case risk priority number. An extensive discussion of related work is provided in [4].

For describing the detector and filter characteristics we provide transformation matrices that modify the assessment vector for the respective stage. In case of a detector, the matrix specifies the ratio of correctly detected failures versus the wrongly and not detected failures. This statically sets the upper and lower bounds for the validity and modifies the assessment vector accordingly. The filter matrix defines the impact that the filter may have on the signal, i.e. the degree of suppressing a faulty signal. A filter that operates as a failure masking mechanism will raise the lower bound for the validity because it eliminates faulty values and thus, may improve the validity of the nominal value. Applied to the assessment vector it modifies the respective elements related to the affected failure types. The assessment vector finally holds for each failure type an element that describes which effect this specific failure will have on the final validity attribute. It should be noted that this number includes the capabilities of the detector and filter with respect to the particular failure. The integration stage collapses the vector representation to a single scalar integrity attribute. This stage uses a selection vector that holds weights for each failure type and therefore allows a further restriction to relevant failure types. E.g. for long term navigation, single outliers of a localization sensor may not be as relevant as a constant offset failure. However, outliers may have a high impact on the validity because of their amplitude. Thus, an outlier would decrease the integrity attribute to an inacceptable low level although it would not be relevant. Another application may need a high validity of differential positions. Here, constant offsets would not play a major role. The selection vector

can adjust these different application needs. In the end, we obtain what we call the system validity, a measure of how good the detection and filter mechanisms will deal with failures.

So far, all the information that is captured in the assessment vector, the transformation matrices and the selection vector is available at design time and quantifies of the engineering assumptions about relevant failure types, their impact, the quality of the detection mechanism and the power of the filter in suppressing the effect of failures.

In the conventional approach, the outcome of this analysis would be compared to a required integrity and in the case of a match, the design would be accepted. This implies that the assumptions about the failures, the detection and filter capabilities are worst-case assumptions and require a substantial amount of resources to keep the bounds. If the design cannot be proved to fulfil the static requirements, improvements have to be applied to the sensors, detectors or filters. In our architectural pattern, the numerical representation of the validity is exploited to detect at run time whether the component will meet the integrity requirements. If the dynamically derived validity falls below a certain threshold, we are able provide the countermeasures on a higher level, i.e. on the level of provided services that may need to be degraded.

At run time, the detector will provide a result for each failure that it is able to detect. These outcomes are stored in a vector with the same dimension as the assessment vector. The elements of the assessment vector are applied as weights to this vector to form the validity vector. The validity vector represents the actual estimated validity as a result of the detector and filter stage. It is transferred to the filter stage where a similar calculation is performed. The final validity vector holds a dynamic estimate about the validity of the generated nominal data item with respect to each failure type. Finally, this vector is converted into a single scalar number which represents the integrity attribute. The details of the assignment of values and the operations defied by the failure algebra are now introduced as follows:

Chapter 2 deals with the assessment of data and system validity and introduce the various data structures that represent and transform the validity attributes. It provides a glossary of representation and transformation data structures. They are distinguished in design time and run time representations.

> Section 2.1 describes the data structures that transform the validity through the stages sensor, detector, filter and integration that were introduced above. Transformations describe the knowledge that is available, how this knowledge is transformed in a validity representation and how detector and filter modify this validity representation respectively.
>
> Section 2.2 provides a detailed description and discussion of the representation forms of validity. It is followed by a numerical example for an entire computational chain in section 2.3.

Chapter 3 deal with the formal definition of the elements of a failure algebra. We define data structures and operations to calculate the validity when passed through a chain of computational components with detector of filter characteristics.

> Section 3.1 provides the definition of the sensor, detector and filter elements.
>
> Section 3.2 defines the operations and section 3.3specifies rules about the combination of elements.

Chapter 4 contains a brief discussion about the notion of uncertainty and validity. This is ongoing work. The problem is that uncertainty and validity are related. Validity always is specified with respect to a given uncertainty. The goal is to accommodate the different uncertainty requirements of different applications.

Finally, Chapter 5 contains a summary of achievements and defines next steps. There are two main problems that we intend to tackle during the last phase of KARYON. The first is to handle

redundancy and fusion by the proposed validity calculus. The second will be the consistent treatment of uncertainty-validity relation.

2. Representation of validity

The validity is an estimate that quantifies the confidence in continuous-valued sensor data. It is a measure represented by a numerical value ranging from zero to one. A validity of zero indicates the least and a validity of one indicates the highest confidence in the outcome of a processing chain. As part of a processing chain we consider a combination of sensor, detector and filter components. By applying the validity concept, such a processing chain outputs sensor data together with a validity value. Consequently, there is no longer in depth knowledge necessary to interpret the provided confidence of a processing chain. Independent of the used components to form a processing chain, we state the confidence one can have in the provided output via an explicit confidence measure called validity. This is in line with the term failure semantics we introduced in Deliverable 2.2 (Chapter 2.2).

In order to achieve this, we introduce different representation forms to describe the validity of a processing chain at design-time as well as at run-time. Table 1 presents a glossary of representation forms, used in the subsequent Figures and the discussion. The variable "n" in the dimension column, is related to the failure modes. It represents the number of considered failures out of a set of relevant failures.

Table 1: Glossary of representation forms

Representation	Dimension	Assigned to	Available at	Description
System bounds	n x 2	processing chain	design-time	bounds within the validity can vary
Assessment vector	n x 1	processing chain	design-time	statistic estimate of the validity
System validity value	1 x 1	application	design-time	statistic estimate of the application-specific validity
Validity vector	n x 1	processing chain	run-time	validity of current sensor data
Validity value	1 x 1	application	run-time	application-specific validity of current sensor data

By using these representation forms, we are able to state the confidence of a certain processing chain by hiding its peculiarities. As a brief introduction, the system bounds define the limits of the remaining representation forms used both at design-time and at run-time. Based on the result of static failure analysis, the processing chain is quantified by an assessment vector at design-time. When we focus on run-time, the validity vector informs about the confidence of current sensor data. Instead of using static failure analysis, this calculation is driven by actual detector and filter results. The system validity value (design-time) and the validity value (run-time) represent an application-specific adjustment. A detailed presentation and discussion of these representation forms are given in Section 2.2.

In order to calculate and to update these representation forms, we define transformations as listed in Table 2. Transformations are used to modifying the validity attribute according to a sensor, detector, filter and an application. Some of these transformations can only be performed at run-time, because they are related to actual sensor data. The other transformations quantify design time knowledge of the sensor, detector, filter and application. For the sensor, we analyse

the set of relevant failures. As a result, we know which failures can occur, what impact these failure have on the sensor data and how often these failure will happen. The first question, which failures are relevant, determines the dimension of the vector holding the characteristics. The second question provides the necessary knowledge to form the impact vector. The third question about the probability is captured in the occurrence vector. The characteristics of a sensor define the inherent properties of a sensor and are relevant at design-time only. Considering a detector or filter mechanism, both design-time and run-time transformation will have to be defined. Design-time transformations express how good a mechanism will work and quantify the coverage of mechanisms. Run-time transformations map the respective detection or filter result on the validity representation. This quantifies the confidence in the respective data. The selection transformation is applied at design-time and at run-time. At design time, it is used to calculate the system validity. This is a measure for the quality of the mechanisms with respect to a defined set of failures. At run time, the selection vector is applied to determine the actual data validity with respect to these relevant failures.

Table 2: Glossary of transformations

Representation	Dimension	Assigned to	Available at	Description
Impact vector (IV)	n x 1	Sensor	design-time	impact of each failure on the validity
Occurrence vector (OV)	n x 1	Sensor	design-time	probability that a certain failure occur
Detection trans. matrix (TD)	n x n	Detector	design-time	false positives and false negative on each failure
Detection result (RD)	n x n	Detector	run-time	failure detected, not detected and unable to detect
Filter trans. Matrix (TF)	n x n	Filter	design-time	static, proportional and elimination parameter on each failure
Filter result (RF)	n x n	Filter	run-time	feedback of filter on the validity
Selection vector (S)	1 x n	Application	design/run-time	selecting relevant failures according to an application need

The following paragraph presents an overview of how to express the information that finally is used to determine the validity. We firstly define properties that are particularly relevant during design-time. Our starting point is the failure model which has been introduced in D2.2. Figure 3 depicts the data structures that comprise the aspects of validity. It should be kept in mind that the engineer who designs a control system takes certain assumptions about the components of a computational chain. In our approach, we try to quantify this engineering knowledge and exploit the numerical representation for calculating the validity of data that we can expect.

The impact vector is strongly related to the assumed failure model. It has an entry for every failure type that holds a weight which defines how much this failure will affect the data of the

component. A weight may e.g. be related to the amplitude of a failure. Actually, the impact vector is a quantification of assumption coverage and specifies which failure types have which impact on the overall validity. Next, we have to determine the bounds within which the actual validity will vary. The range vector has one entry defining this window related to the type of failure. The upper bound is given by the value in the impact vector. The lower bound for a sensor is initially assumed to be "0". For computational components it may be >0. These cases are discussed below. The assumptions about the occurrence probability is maintained in the occurrence vector. It should be noted that the occurrence of a certain failure type may not only depend on the malfunction of a component but also on the environment in which this component is used. This is particularly true for sensor failures (e.g. a radar sensor will produce wrong results in heavy rain). Combining the information about the window of possible validities and the occurrence probabilities allows assessing the overall validity of the component. It is captured in the assessment vector on a per failure type basis.

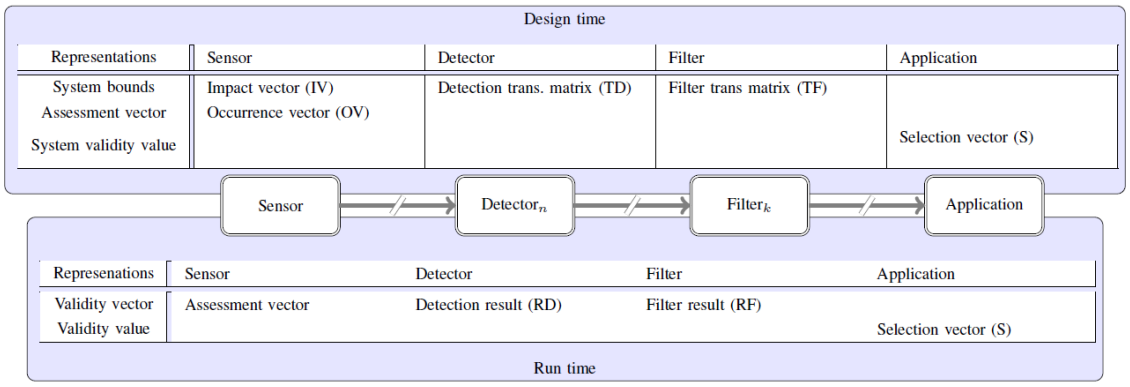


Figure 3: Schematic representation of validity

So far, the occurrence vector constitutes a general representation of the system validity derived during design time, i.e. what confidence we can put in the mechanisms. For an application, however, it may not be necessary to consider all failure types from the model. E.g. for a parking assistant, a failure type only occurring for long-range measurements of the distance sensor may be not relevant. This becomes evident when having done the break-down of safety requirement into the actual architecture where the component under consideration is instantiated. Therefore we apply a selection vector that masks out irrelevant failures. It should be noted that this is crucial because here the requirements of the application meet the capabilities of the components in the processing chain. If an application has a high requirement on short distance measurements, a low confidence in long-distance measurements of this sensor will lead to a false low confidence that the application can have in this data. The selection that contains an application specific weight for each failure type is able to correct this for every application individually. From the combination of the selection vector with the assessment vector we derive a single value that represents the confidence that we can put into the output of the processing chain. We call this value the static system validity value.

During run-time, the actual sensor value is passed through the respective components. At the interface to the rest of an application we generate a dynamic data validity value together with the actual data. This was described in D2.2. The data validity is the outcome of the assessment in the checking and filtering components. For each component, the assessment vector that has been derived during design time is combined with the actual validity that is estimated by the respective component. For each of the failure modes that are affected by the component (checked in a detector or filtered) there is an entry in the outcome vector. The dynamic validity vector holds these outcomes weighted by the assessment vector. In the last step, the selection vector is applied that masks the failure modes that are not relevant for the application. As a result, we obtain a single scalar data validity value.

2.1 Transformations

Transformations generate a validity representation and transform one validity representation into another. We separate transformations from validity representations in order to reach a uniform calculus. Transformations are tightly related to a certain component, e.g. the impact vector is related to a sensor component, which is used to initialize the validity calculation. As another example, the selection vector selects failures from a vectorized representation according to an application need, which is, of course, an application-dependent representation. Transformations are essential to generate, to update and to scale the validity representation. These operations will be further explained in Section 3. In the remainder of this section, we introduce transformations for sensor, detector and filter components and for selecting application requirements.

2.1.1 Sensor-dependent transformation

The failure behavior of a sensor is described by an impact vector and an occurrence vector. The impact vector characterizes the severity of failures. The probability of failures is captured by the occurrence vector.

2.1.1.1 Impact vector (IV)

The impact vector informs about to which extent failures affect the intended behavior of a component. In general, failure types are defined by the failure model. The failure model specifies a set of failures where each failure is further characterized by a severity and a probability attribute. The completeness of the failure model is expressed by the assumption coverage. A fault-free system will be represented by an assumption coverage of zero. Assuming arbitrary failures this will result in an assumption coverage of one. The assumption coverage is a means to quantify the completeness of the model. We use it to limit the validity as an indication of an improperly defined failure model. The problem is what we define as a failure in a sensor-based system and where we have to deal with inherent measurement uncertainties like thermal noise and conversion errors. To overcome this difficulty, a failure model is linked to an uncertainty margin. An uncertainty margin defines a measurement range within the true measurement is supposed to be with a certain confidence. Measurements that deviate within this bound from the ideal value are not considered as a failure. Only values that cross the uncertainty margin are considered to be relevant and treated as a failure which will lower the validity.

The remainder of this subsection is to introduce the impact vector representation and the relation to the failure model, the uncertainty margin and the assumption coverage.

The starting point to build an impact vector is the definition of a failure model. This definition implies a profound analysis of the failure types as well as their respective significances and probabilities. The significance, also referred as deviation or as severity, determines which impact a certain failure type will have on the ideal value. Additionally the probability to fail is recorded for the listed failure types. The listed failures are modeled using a Markov model, which is well known as a memory-less statistically modeling technique. The memory-less property fits well to the way in that failures usually occur. To build a failure model, a set of n+1 states is formed according to the Markov model. The n states are representing the different failure types and the additional state is assigned to the failure-free behavior. By the properties of this modeling technique, the probabilities to fail are reflected by the transitions of the Markov model. The steady state of the Markov model shows the overall percentage that a certain state will be active. This knowledge is used to build the occurrence vector, which will be focused in the next subsection. Attached to each failure state, a distribution function of the respective significance is stored to rate the further on build validity term. Frequent failures with a large

deviation from the correct value get the highest significance. Vice versa, sporadic failures with slight deviations from the correct value have a smaller impact.

In practice, we consider permanent, sporadic and statistical failures as introduced in D2.2. In this document we consider failures in which the value of the perceived data deviates from the ideal value of a real world entity. There are also other properties conceivable, for instance the slope, the frequency and spectrum of a signal or the time-entity of a real world entity. Therefore our considerations rely on an amplitude-based failure model. This does not imply that the validity concept is limited to this kind of failure model.

The impact vector is defined as an n-dimensional vector representing the failure types (states) of the failure model. This consistent representation is a key issue to achieve interoperability when communicating the validity across system borders. In particular, the assessment of remote sensor data will only be possible if this consistent representation is available. Figure 4 illustrates an example of an impact vector together with a failure model. It should be noted that the additional state assigned to the failure-free behavior is not part of the impact vector, because the failure-free behavior is inherently expressed by the validity exactly in the case when no failure occurred and the maximum possible validity (one) is reached.

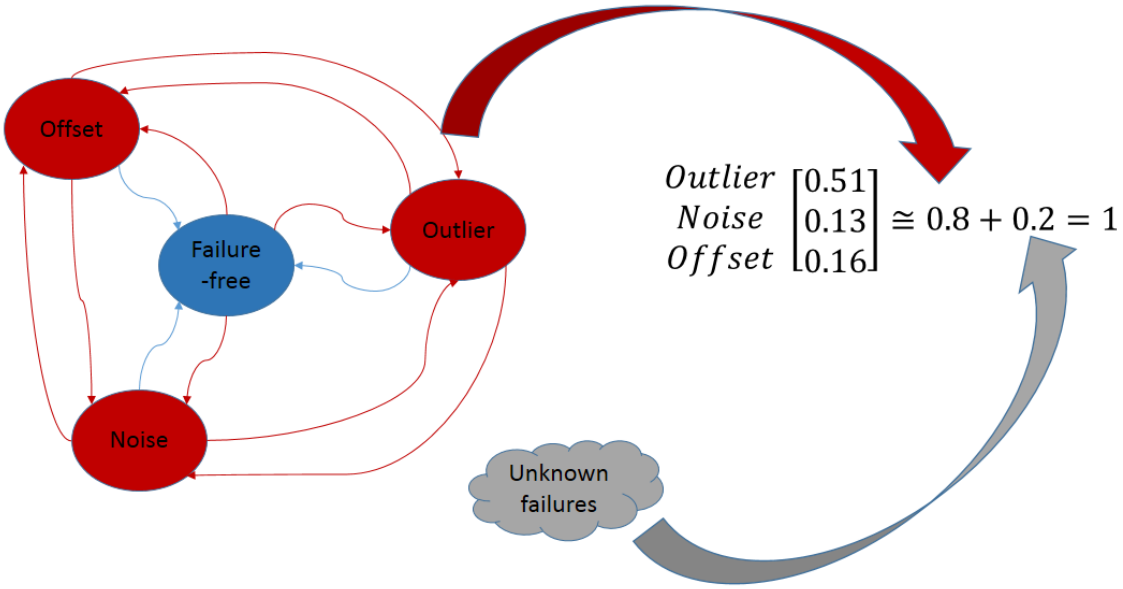


Figure 4: Impact vector based on failure model

This argument is only valid if the failure model is precisely defined. Otherwise the assumed failure-free behavior would lead to an uncertain significance on the components output because the set of non-modeled failures is inherently mapped to the failure-free state. This would obviously limit the usability of the failure model. In fact, it needs to be noticed that each model shows a certain degree of insufficiency. In order to inform about the degree of incompleteness, the term assumption coverage was introduced by David Powell in the early 90's [5]. The assumption coverage reflects a conditioned probability that the defined failure model covers the abnormal behavior when failures occur. In other words, the assumption coverage reflects the ratio between the considered failures and the entire set of possible failures. The complete set of failures is difficult to define. Because of the many degrees of uncertainty and the lack of complete knowledge of the physical characteristics of a sensor, we base the failure model and the estimation of the assumption coverage on an empirical approach. Because of an incomplete failure model, the assumption coverage will limit the validity. For instance, if the assumption coverage is estimated to be 0.8 as shown in Figure 4, the validity can never go beyond 0.8, even if none of the considered failures occurred.

The final aspect to form the impact vector deals with the uncertainty margin of a measurement. A high uncertainty margin leads to a simplified determination of the assumption coverage

because only failures are considered to be relevant when they exceed the defined uncertainty margin. Fewer failures have to be considered if the uncertainty is set to a large margin. Vice versa, a small uncertainty requires a more profound analysis to consider the set of possible failures. We conclude that there is a relationship between the assumption coverage and the uncertainty margin. The higher the uncertainty margin, the higher is the assumption coverage. In practice, the uncertainty margin is set according to an uncertainty margin given by the data sheet of a sensor. For a further explanation of the relation between uncertainty and validity see Section 4.

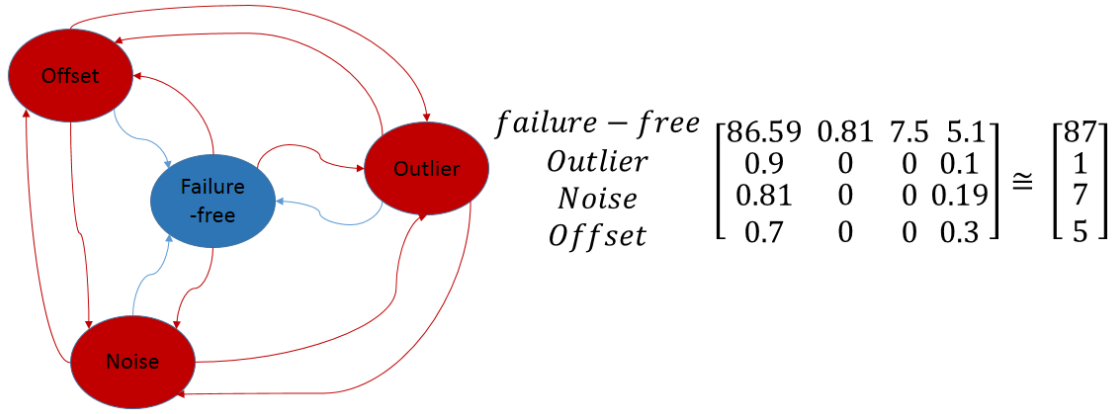


Figure 5: Occurrence vector based on failure model

2.1.1.2 Occurrence vector (OV)

The occurrence vector represents the probability to fail. For each failure listed in the failure model, the occurrence vector holds a respective probability. Hence, the occurrence vector is of the same dimensionality as the impact vector. A probability of a certain failure ranges from "0" representing the case that this particular failure will never occur to "100%" in case of a known permanent failure. The exact values are deduced from the steady state of the Markov model, which is used to model the failure behavior. As mentioned above, the steady state reflects the probability that a certain state, associated to a failure, will be active in total.

In our data-centric failure model, we assume that a sensor is the only source of failure in the processing chain. We do not consider at the moment failures that may be introduced in the detector or filter stage. A detector does not change the nominal data value and therefore does not modify the probability of a failure. A filter that mitigates or masks a failure changes the nominal data value and therefore may affect the probability of a failure. This is reflected by the respective filter transformation that includes the knowledge about filter characteristics.

In Figure 5, we depict an example of an occurrence vector together with its failure model. In this example the failure-free state and three failure states are considered: outlier, noise and offset. The probability to leave a state or to go to another state is described by the transition matrix shown beside the failure model. In this matrix the first row stands for the probability to stay failure-free (86.59%), to go to an outlier failure (0.81%), to go to a noise failure (7.5%) or go to an offset failure (5.1%). The remaining rows can be interpreted respectively. We derive the steady state from this knowledge (shown right of the transition matrix). The steady state indicates how much time the system spends in the failure-free (87), outlier (1), noise (7) or in the offset state (5).

2.1.2 Detector-dependent transformation

The detection characteristics identify the capabilities of a detection mechanism. A detector which is not perfect may wrongly detect a failure, even though no failure is present. Vice versa,

a detector does not detect a failure even though it affected the nominal value. These situations can best be expressed by the notion of true positive, false positive, true negative and false negative [6]. A true positive represents a real failure, which was accurately detected. In the case of a false positive, the detector has erroneously identified a failure. A true negative describes the correct outcome, when no failure occurred. Finally, the false negative reflects the case that a failure occurred but was not detected.

2.1.2.1 Detection transformation matrix (TD))

This matrix quantifies design time knowledge. We use the notion of true positive, true negative, false positive and false negative to update the validity representation. In an ideal system, where no false positives and false negatives appear, we receive a validity of zero in case of a detected failure and otherwise a validity of one. Figure 6 visualizes the mapping of detection results on the validity representation. The weight of the failure ($w_{failure}$) describes the impact on the total validity and is taken from the impact vector (see 2.1.1.1 above). As shown in Figure 6, the full weight enters into the calculation of the validity, when no failure occurred. This case is known as true negative (TN). An accurate detected failure described as true positive (TP) leads to a weight of zero to the calculation of the validity.

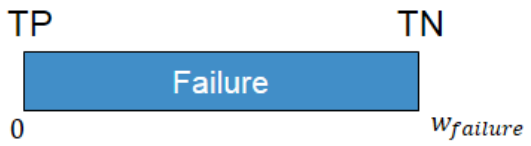


Figure 6: Validity representation of an ideal detector

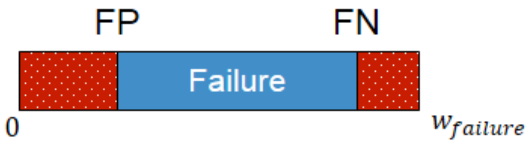


Figure 7: Validity representation of an imperfect detector

But in the case of false positives (FP) or false negatives (FP), the weight given to the validity calculation must be adapted, as shown in Figure 7. Instead of setting the weight either to zero or to the full weight ($w_{failure}$), we argue for increasing the lower validity bound and for reducing the upper validity bound in accordance to the false negatives and false positives. As a result we only consider the remaining window of confidence in the data. Even when a failure was detected, the confidence does not go to "0" because of the non-zero possibility that this is a true positive. The percentage of the false positive (FP) represents the lower bound of the validity. The similar argument applies to the upper bound that reflects the lack of confidence in the data due to failures that were not detected. The percentage of false negative (FN) represents the upper bound of the validity.

2.1.2.2 Run-time Detection result (RD)

The run-time transformation maps the detection result to the validity representation. The validity is defined as a confidence measure ranging from zero to one, where a validity of zero indicates the least and a validity of one indicates the highest confidence. A detector may have an arbitrary representation of its detection results. Therefore, a detection result must be individually mapped to a corresponding validity. Such a mapping must clearly distinguish between a detected failure, a failure not detected and a failure that cannot be detected by the detector. Particularly, the differentiation between "not able to detect" and "not detected" failures plays an essential role for the validity calculation. Suppose a set of detectors is unable to detect a failure, this should be mapped neither to a validly of zero meaning that a failure is detected nor to a validity of one meaning that a failure is not detected. We have to prevent an arbitrary interpretation in such a case. To solve this problem, we defined a neutral element to deal with the inability to detect failures in a consistent way. When discussing the failure algebra in

Section 3, we introduce the mapping of a detection result on the validity representation and the neutral element.

2.1.3 Filter-dependent transformation

A filter suppresses the effect of failures on the nominal data value. Therefore, it may actually improve the validity of data. On the other hand, a filter can be designed too aggressive or too weak. In the first case actual sensor data may be filtered out. In the second case, sensor data is still affected by failures. Consequently, it is essential to find the balance between these two extremes. Anyway, it needs to be noticed that a filter usually comes along with considerable side effects. For instance, a filter technique to suppress outlier failures may use a moving average method. This filter will mitigate the effect of outlier failures but, at the same time, this filter will lead to an additional offset and delay. The following subsection introduces a set of parameters to clearly describe both the positive and the negative effect which a certain filter mechanism may have on the validity.

2.1.3.1 Filter transformation matrix (TF))

This matrix is formed at design time. Expressing the characteristics of a filter technique, we make use of static, proportional and elimination characteristics. Additive effects can be expressed by static parameters. Independently whether the filter mitigates failures or is just idle, additive effects are present. Taking the moving average filter example, the filter output is always delayed. The wider the moving average window of this filter was specified, the larger the delay. The described additive effects are static and mainly dependent on the filter design. The dynamics of a failure do not have any impact on additive effects. Proportional parameters are used to express either positive or negative effects of the filter. Mitigating failures are clearly a positive effect. Again considering the moving average filter example, the mitigation of outlier failures is proportional to the window of the moving average filter. Let us first consider a window of one. This leads to a proportional parameter of one, which means there is a one to one relation between input and output. Obviously this makes sense because a moving average filter with a window of one will not mitigate failures. This is represented by a proportional parameter of one. When we enlarge the window of the moving average filter, the capability of the filter to mitigate failures improves. This relation can be described by proportional parameters less than one. In this case the relation of failures on the output to the input is lowered. In detail, either the impact (failure amplitude) or the occurrence rate (probability) of a failure might be reduced. In both cases those results represent positive effects. A negative case is expressed by a proportional parameter greater than one which means that the amplitude or the occurrence of a failure is larger on the output compared to the input. Taking once more the moving average example, the capability to mitigate outlier failures results in an additional offset, which is a side effect of the applied filter mechanism. When an outlier failure occurs, the discussed filter introduces an offset failure or increased noise because of the attenuation effect. This failure was not present before. By setting the proportional parameter adequately we are able to express this kind of relation. Finally, to express complete elimination of a failure we provide an eliminating parameter. However, complete elimination requires substantial effort of redundancy and calibration and therefore is achieved rarely.

During design-time, the introduced parameters are used to define the lower and the upper validity bound according to the filter capabilities. The minimum validity is marked by the lower bound. A filter can always guarantee that validity cannot fall below this bound. The filter however is not supposed to suppress, convert or mitigate any extent of a failure, e.g. any amplitude, any slope etc. The best filter result is expressed by the upper validity bound, which can be only achieved if the assumptions on the extent of failure hold. Thus, a filter cannot guarantee the upper bound under all operational conditions. The quality of a filter is therefore mainly determined by the lower bound for the validity.

2.1.3.2 Filter result (RF)

During run-time the filter feedback selects a validity value out of the pre-defined validity bounds, which were assigned during design-time. The filter feedback is an active filter status informing about the current mitigation capabilities of the filter. This information can be extracted either by the filter only or else by a detection-filter combination in case the filter operates as a black box. The lower validity bound will be taken, if the filter shows its worst mitigation performance. On the other hand, the upper validity reflects the best achievable result that can be delivered by the filter.

2.1.4 Application-dependent transformation

2.1.4.1 Selection vector (S)

In a final step, the vectorized validity representation is converted into a single scalar validity value. We provide an assessment vector that holds the estimates about the quality of the detector and filter with respect to the failure model. The assessment vector has been defined at design time. The complementary validity vector has passed through the various stages of the processing chain and has been modified as described above. It finally holds a validity estimate with respect to every failure type of the failure model that has been generated dynamically at run time. For an application it may be more appropriate to have an application specific validity value that reflects the estimates for all the failures which are relevant for the application. Several applications may require a validity estimate even though they have different requirements and failure sensitivities. For instance, a navigation application may be sensitive against all failures listed in the failure model. Another application is to avoid collisions, which only requires dealing with major deviations like outlier failures.

To express these different requirements, we provide a selection vector that is applied to the assessment vector and the validity vector. To compress the vectorized information to a single value, the selection vector is of the same dimension than the system and the validity vector. We obtain a system validity that is a measure for the quality of the mechanisms and a data validity that characterizes the confidence in the nominal data. The system validity is derived during design time. It characterizes a system property and is used to check whether the mechanisms are appropriate to fulfill the demands of an application. The selection vector masks out the failure types that are not relevant for the application and otherwise would inappropriately lower the overall system validity. In the example, the selection vector of the navigation application considers the entire set of failures while the selection vector provided by the obstacle avoidance application marks out all failures except outlier failures.

The data validity value is derived during run time by applying the detector and filter mechanisms. It represents a measure for the actual individual data item. The selection vector is the same as for calculating the system validity. At run time it allows only to consider the relevant outcomes of the computational components. The different representation forms are detailed in the next chapter.

2.2 Representation forms

The section introduces the validity representation forms:

- System bounds,
- Assessment vector,
- System value,

- Validity vector and
- Validity value.

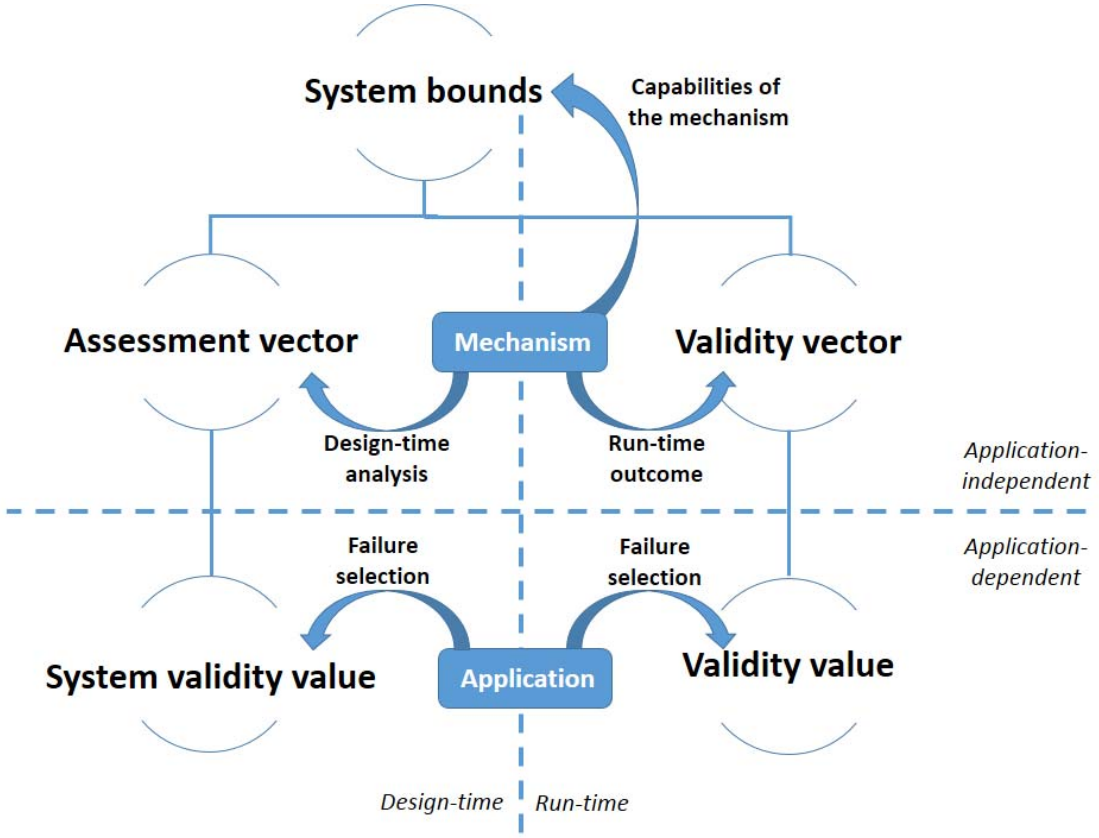


Figure 8: Forms of validity representation

The representation forms describe the validity of a system, which consists of a sensor and any combination of detection or filter mechanism. At design-time, we start to define the bounds of the validity with respect to the capabilities of the system (sensor, detector or filter). In case of an ideal system, equipped with perfect detection and filter mechanisms, the validity ranges from zero to one. Any kind of insufficiency leads to a limitation of these optimal bounds. Consequently, the system bounds reflect the quality of the system and at the same time they determine the anticipated validity that can be expected from this system. The system bounds are the starting point for both the design-time assessment and the run-time validity calculation. At design time, we derive assumptions about these system bounds by a careful analysis of the failure types their impact and occurrence (impact vector, occurrence vector) as described in Section 2.1. From this information, we derive the assessment vector that maps this information to the system bounds. At run-time, the outcome of detection or filter mechanisms allows calculating the exact data validity of an individual actual sensor reading. To consider application needs, we perform a failure selection on both design-time and on run-time as presented above. Out of the assessment vector a single validity, called system validity value, can be computed. During run-time, the application dependent validity is called validity value. Figure 8 summarizes the validity representations and shows the classification of design/run-time and application independent/dependent validity forms. Subsequently, we provide a detailed description of the representation forms.

2.2.1 System bounds

The system bounds specify an anticipated validity range within the output of a system will vary during run-time. The lower bound of this range defines the validity in case of a failure. In the ideal case this will be set to zero, which means that the failure is reliably detected. On the other side, the upper bound of the system bounds marks the best case. An upper bound of one would represent the guaranteed fault-free case. Unfortunately, because of the deficiencies of the sensor, detector or filter components, these bounds will not be reached. The system bounds are defined as a matrix of the dimensionality (2 x n), where one column describes the lower bound and another column specifies the upper bound for each of the n failure types. The system bounds are successively updated according to the properties of the sensor and several detection and filter mechanisms. This will be described in Section 2.1.

The system bounds indicate how good a system can distinguish between a failure-free and a failure state. To specify the bounds more precisely, we need to quantify assumptions about the system. We use the well-known concepts of *assumption coverage* and *coverage of the mechanisms* to define these bounds. Assumption coverage quantifies the completeness of the failure model. This has an impact on the upper bound of the system bounds because every failure that is not considered but may occur will decrease the validity of the data item. The representation of system bounds is defined as a "2 x n" matrix where the columns hold the lower and upper bound of a failure and each the row in associated with a failure type. Assuming a failure model, which contains three failure types, this will lead to a dimensionality (2 x 3) of the system bound. The matrix is initialized with system bounds derived from the information of the impact and the occurrence vector described in 2.1.

The coverage of mechanism describes the quality of detection and filter mechanisms. Therefore, the coverage of mechanism affects both the upper system bound. The quality of a detector is characterized by false positives and false negatives (see Sec. 2.1.2). In the case of a false positive, the detector has erroneously identified a failure. The false negative reflects the case that a failure occurred but was not detected. Thus, even when a failure is detected, there is a non-zero confidence in sensor data. This confidence is given by the probability of false positives. Therefore, the validity of a failure type will not drop to zero even if a failure is detected. The percentage of the false positive (FP) represents the lower bound of the system bound. The false negative (FN) represents the case that a failure was not detected. Taking into account these measures, the upper bound is expressed by the percentage of false negatives.

Filter mechanisms have an impact on the lower system bounds because a filter mitigates the impact of a failure or even masks the occurrence of a failure. In case of mitigation, the lower bound will be increased. The validity cannot fall below this increased bound because the failure is to some extent less crucial. E.g. if a filter substantially decreases the amplitude of a certain failure the overall error is lower and the confidence can increase in this case.

2.2.2 Assessment vector

The assessment vector is a design-time estimate of the validity. During run-time, the outcome of a mechanism is exploited to calculate the dynamic validity. When generating the validity, we make use of the knowledge defined by the occurrence vector in Sec. 0. We argue that a validity of one implies no failures at all. On the other hand, a validity of zero is a 100 percent probability of a failure. Applying the occurrence vector that holds the probabilities of failure to the system bounds we derive the assessment vector.

The components of an assessment vector are not binary (failure or failure-free). Instead they represent a set of continuous values according to the analyzed failures of the sensor. This reflects the quality of the used sensor versus the respective failures. For example, a sensor of low quality will lead to a lower validity when compared to a high quality sensor, which of course will exhibit a lower probability to fail.

Detection mechanisms do not affect the assessment vector because a detector does not change the data and thus has no effect on set of failures. The probability of a failure stays the same. In contrast to a detector, a filter mechanism may lower the probability of a faulty data item at its output. In Sec. 2.1.3, we argued that a filter either mitigates the impact of a failure or even reduces the probability of a failure. It should be noted that mitigation has effects on the system bounds (see Sec. 2.2.1).

A filter may transform a sever failure into a less severe failure. In this case, the probability of the severe failure drops down to zero and the probability of the less severe failure goes up to the probability that the severe failure had before. The filtering process might lead to an unwanted effect that a filter lowers the probability of one failure and raises the probability of another one. These are effects that have to be treated thoroughly to keep the main objective of a filter mechanism: increasing the validity.

2.2.3 System value

The system value generated out of the assessment vector described above. This system value is similar to the RPN (risk priority numbers) of the FMEA (failure mode and effect analysis) without the specific disadvantages. A detailed discussion of the advantages of our approach can be found in [4]. Here we briefly summarize the results. A RPN ranges from 1 to 1000, where 1 belongs to the best and 1000 to the worst case respectively. The multiplication of the amplitude of a failure, the probability of a failure and the detectability of a failure results in the RPN of a system. Each of these three factors, vary from 1 to 10. RPNs are informally defined, which may cause a problem when used beyond system borders. Because there is no single failure model as a root of RPN calculation, it is not guaranteed that there is an agreement about failures and impacts. Further, the FMEA does not distinguish between false positives or false negatives for a detector. Because of the lack of clear separation between false positive and false negative, every misbehavior of a detector is mapped to a single case. Moreover, only considering detectability is misleading. Because a detected failure does not necessarily mean that the failure can be handled. Therefore, the controllability of a failure is as essential as the detectability. In addition, the RPN is limited to the system assessment during design-time. There is no possibility to take detection or filter results into account for a run-time assessment. Despite to the fine-grained failure analysis, the RPN merely reflects the most serious failure case. Because of all that, our approach to validity calculations shows many important improvements. To generate the system value from the assessment vector we introduce the selection vector. Two cases must be distinguished.

Firstly, the set of relevant failures considered by the failure model is larger than the set of relevant failures of the application. In this case, the selection vector allows us to mask out all failures that are represented in the assessment vector but are not relevant for the application. To generate the system value, we select and rebalance the relevant failures in a way that the significances of the selected failures are bounded by the assumption coverage.

Secondly, the set of relevant failures considered by the failure model is lower than or equal to the set of relevant failure of the application. If the assumption coverage of the failure model is close to one, we generate the system value by calculating the sum of the assessment vector. But if the assumption coverage is expected to be significantly lower than one, we have to deal with a serious issue, given that there may exist an unconsidered failure in the failure model, which might be considered as a relevant failure by the application. In this situation, the generation of an application specific system value is prohibited.

2.2.4 Validity vector

The validity vector informs about the current validity of the sensor data during run-time. Based on the system bounds, we determine the validity vector in consideration to the outcome of the used detection and filter mechanisms. In fact, the system bounds define a range of validities

taking into account the knowledge of the failure model and the characteristics of the applied detection and filter mechanisms. This knowledge is used to determine the precise validity according to each failure type while run-time.

First of all, the validity vector needs to be initialized using the outcome of a component. Concerning a detector the detection result will be taken, but the validity vector is also assigned to a sensor, which does not have a comparable outcome. The initialization of the validity vector is rather problematic, because there is no knowledge or mechanism available in a pure sensor device informing about the validity on run-time. A plausible argumentation is to initialize the validity vector with the mean value of the validity bound defined by the impact vector. Such an argumentation is neutral and will be neither interpreted as a failure nor considered to be failure-free.

By focusing on the detection, the outcome of a detection mechanism will be either: failure is detected or failure is not detected. In an ideal system, we expect to receive a high validity when no failures occur and in case of failures a decrease of the validity. This idea can be implemented by taking the upper bound of the assessment vector if the detector does not detect any failure. On the other side, a detected failure leads to a validity, which is given by the lower bound of the assessment vector.

A similar approach is applied for filter components. According to the filter feedback, the validity vector is set. The filter feedback expresses the ratio to mitigate a certain failure. A high filter feedback relates to the highest possible filter performance, which is described by the upper bound of the assessment vector. Otherwise the lowest bound of the assessment vector is taken to assign the validity vector.

2.2.5 Validity value

The validity value is an application specific generated value out of the validity vector. In general, an application is incapable to exploit the details of the validity vector. The application is mainly interested in a single value informing about the confidence of the sensor data. Exactly, this need is fulfilled by the validity value. The generation of the validity value does not differ from the system validity value introduced in Section 2.2.3. In general, two cases have to be distinguished.

First, the set of relevant failures considered by the failure model is larger than the set of relevant failures of the application. In this case, we have to sort out the failures in the validity vector which are not relevant for the application. To generate the validity value, we select and rebalance the relevant failures in a way that the significances of the selected failures are bounded by the assumption coverage. Subsequently, to receive the validity value, we sum up the validity vector.

Second, the set of relevant failures considered by the failure model is lower than or equal to the set of relevant failure of the application. If the assumption coverage of the failure model is close to one, we generate the validity value by calculating the sum of the validity vector. But if the assumption coverage is expected to be significantly lower than one, we have to deal with a serious issue, given that there may exist an unconsidered failure in the failure model, which is considered as a relevant failure by the application. In this situation, the generation of an application specific validity value is prohibited.

2.3 Example

In this subsection we present an example to illustrate the concepts introduced in Section 2. In this example, we consider a sensor, a detector and a filter as a processing chain. This processing chain will be then linked to different applications. The example explains the dynamics of the validity concept and highlights the respective representation forms.

Figure 9 depicts the overall structure showing the design-time representation forms above and the run-time representation forms below the processing chain. We use bar charts to present these representations graphically. For each failure type, we provide a respective bar. The first row lists outlier failures, the second row noise failures and the third row offset failures. Gray bars represent the theoretical bound of these failures. These bounds are modified by the components of the processing chain and therefore, the actual bounds within a validity can vary are shown in blue. Beside these charts, we provide the corresponding numerical representation by using matrices and vectors. We will now proceed through the processing chain starting from the sensor.

Sensor:

The sensor is characterized by the impact and the occurrence vector from static failures analysis. These vectors are used to initialize the system bounds and the assessment vector for the static validity calculation and the validity vector that is updated at run-time. System bounds, assessment and validity vectors characterize the sensor and express the expected quality of the sensor. The following equation shows the validity calculation for the sensor failures. Multiplying the impact vector with the inverse of the occurrence vector (component-wise) results in a vector that initializes the system bounds matrix, the assessment vector and the validity vector.

$$System_{bounds} = impact_{vector} \circledast (1 - occurence_{vector}) = \begin{bmatrix} 0.51 \\ 0.13 \\ 0.16 \end{bmatrix} \circledast \begin{bmatrix} 1 - 0.07 \\ 1 - 0.06 \\ 1 - 0.11 \end{bmatrix} = \begin{bmatrix} 0.47 \\ 0.12 \\ 0.14 \end{bmatrix}$$

Detector:

Detector capabilities are expressed in terms of false positives and false negatives. The system bounds are modified respectively. We set the lower bound according to false positives and the upper bound according to false negatives. As described earlier, the assessment vector is not further modified by the detector.

During run-time the detection result will be used to calculate the validity vector. In case of a failure the lower system bound will be taken. The upper system bound is used when no failure has been detected. Figure 7, we illustrate both cases.

Filter:

The filter component in our example was designed to mitigate outlier failures. As a side effect the filter mechanism produces additional offset failures. Noise failures are not affected due to the filter. Exactly this is expressed by the system bound. The validity entry (first row) regarding outlier failures is increased and the validity entry (third row) regarding offset failures is decreased. It should be noted that the only validity bounds are modified, which are affected by the filter. This allows us to exploit detection results also after a filter has been applied. Therefore, the second row (noise failure) of the system bound is not changed by the filter. The system bounds are applied to modify the assessment vector. Outlier failures and offset failures which are modified by the filter also modify the assessment vector. The noise entry of the assessment vector is left unchanged.

Table 3: Sensor characteristics

	Impact vector	Occurrence vector
Outlier	0.51	0.07
Noise	0.13	0.06
Offset	0.16	0.11

Table 4: Detector characteristics

	False positive	False negative
Outlier	0.17	0.01
Noise	0.07	0.02
Offset	0.01	0.01

Considering the validity calculation at run-time, the filter removes or mitigates the effect of outliers and offset failures as it has been specified by the impact of filter characteristics on the bounds at design time and hence, there will not be a dynamic change of the validity with respect to outlier and offset failures during run time. Only a detected noise failure will change the validity.

Integration stage (application):

In this stage, the validity representation as a vector that reflects validities according to the respective failures is converted into a single validity value. As we explained in Section 2.1.4.1, this mapping will be applied at design-time to form the system value and at run-time to generate the validity value. A selection vector expresses which failures will be considered for the mapping. This may change according to application requirements. In our example, we assume two applications which have a different sensitivity against failures. Application A, shown in Figure 9, gets affected by any failure listed in the failure model. In this case all failures will be considered. The following equation performs the conversion of the assessment vector into the system value. The selection vector simply controls, which failure type will be included in the single validity value. The impact vector is necessary to calculate the validity weights and the assumption coverage is used to limit the upper bound of the resulting calculation. Note that operations followed by a dot have to be performed element-wise.

$$systemVal = (AV./.IV)' * (acov *.(IV *.S)/.(IV * S))$$

In case all failures are selected, this equation performs like a sum, because the devisor of the first quotient is then equal to the result of the second quotient. When all failures are selected, the resulting weight of the second quotient corresponds exactly the weight which was used to calculate the input (assessment vector). Therefore, the system validity for Application A is just a sum of all validity entries of the assessment vector, as shown in the following equation.

$$systemVal_A = 0.5 + 0.12 + 0.09 = 0.71$$

Considering the calculation of the validity value at run-time, the validity vector will be taken instead of the assessment vector. The respective equation is shown below.

$$validityVal = (VV/.IV)' * (acov *. (IV *. S)/. (IV * S))$$

While Application A is considered, the selection vector stays the same. Therefore, the sum of the validity vector results in the validity value. The following equation shows the calculation of the validity value for Application A in case of no failure.

$$validity_A = 0.5 + 0.12 + 0.09 = 0.71\ (in\ case\ of\ no\ failure)$$

The calculation of the validity value in case of a failure can be done in the same way, as illustrated below.

$$validity_A = 0.5 + 0.01 + 0.09 = 0.6\ (in\ case\ of\ a\ failure)$$

The application-specific validity value ($validity_A$) changes very little when a failure is present. This is because the outlier failure has been filtered and only detected noise failures affect the validity value. In case of no failure the validity value does not reach its maximum (0.8) because of the false negatives of the detected noise and the additional offset due to the filter operation which finally results in a lower validity.

Application B, presented in Figure 9, may be sensitive against outlier failures only. In this case the power of the selection will come into play. By analyzing the following equation, the first quotient provides the ratio of the actual validity within the defined limit given by the impact vector. This, of course, has to be performed element-wise. The second quotient is to calculate the new validity weight for the selected failure types. By multiplying the assumption coverage (acov), the new weight will be limited. The new weight is the result of the division of the impacts of selected failure types to the entire set of selected failure types. This is nothing else than a reallocation of selected failure types on the validity bound reaching from zero to the assumption coverage.

$$systemVal = (AV/.IV)' * (acov *. (IV *. S)/. (IV * S))$$

The following equation shows the calculation of the system validity for Application B.

$$systemVal_B = \left(\frac{0.5}{0.51}\right) * \left(0.8 * \frac{0.51 * 1}{(0.51 * 1)}\right) = 0.78$$

By using this calculation, the validity value for Application B can be performed accordingly.

In summary, Application B profits from the applied filter, whereby Application A suffers under the negative effect of the filter. To draw a conclusion of these results, Application B will be provided by sensor data of high quality (0.78 out of 0.8). These achievements are due to the mitigated failure amplitude of outliers and the fact that outliers appear rarely. Application A also profits from an increased validity due to mitigated outlier failures but at the same time the validity regarding offset failures is lowered as a side effect of the applied filter mechanism. This leads to a slightly decreased validity compared to Application B.

The necessary operations to perform these calculations are part of the failure algebra introduced in the following section.

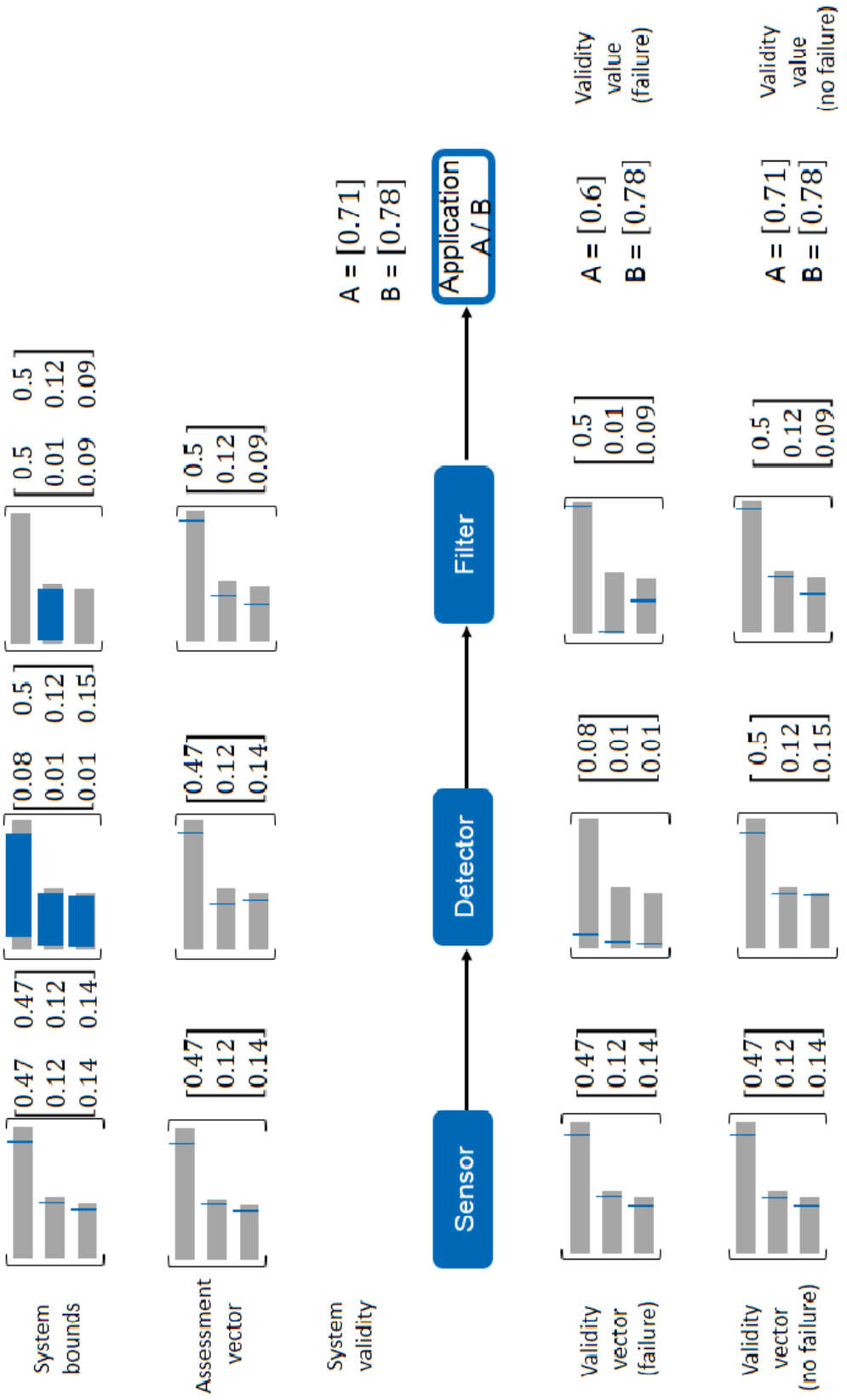


Figure 9: Schematic representation of an example

3. Elements of the failure algebra

The failure algebra forms a mathematical basis to calculate the validity of a sequence of components that we call a processing chain. Such a processing chain may result in a highly complex system, which consists of several detector and filter components. In a conventional system, where all these parts are known at design time, the assessment of the entire system might be feasible. In KARYON, we have to address systems in which essential parts are not known beforehand. This is because remote sensor information is used that is further processed by an application. As outlined before this is the major motivation to introduce the validity measure. In the previous section, we introduced representation forms for different aspects of an overall validity that can be evaluated by the application. These representation forms define the numerical basis towards a failure algebra. We now need to define operations and rules to transfer one representation form into another. Moreover the operations are key to deal with non-pre-determined knowledge of processing chains. By taking the advantage of a failure algebra, the effect of each component of a processing chain becomes effective, even if the sequence of components would be unknown. An inductive update of the validity results in the confidence measure of the entire processing chain.

In this section, we mathematically define the failure algebra and explain in detail the operations as well as the rules.

3.1 Definition

The failure algebra is defined by a set of validity representations (first equation below), a set of transformations (second equation below) and a set of operations as shown in the third equation below. The validity representation (rep_val) ranges from zero to the assumption coverage, which limits the validity representation due to the imperfection of the underlying failure model. In first equation below, the validity representation is defined as a set of the system bounds (SB), assessment vector (AV), system validity value (SV), validity vector (VV) and validity value (V), which are described in Section 2.2. The transformation ($trans$) holds the necessary information in order to update and to transform the validity representation specified in the second equation below.

$$rep\_val = \{SB, AV, SV, VV, V\}, rep\_val = [0, acov] \epsilon \mathbb{R}$$

$$trans = \{IV, OV, T_d, T_f, R_d, S\}, trans\ \epsilon \mathbb{R}$$

$$failure\_algeba = (\{rep_{val}, trans\}, \otimes, \circledast, \oplus, \odot)$$

3.1.1 Sensor

The occurrence vector and the impact vector are defined to characterize a sensor component. In Figure 10, we show the generation of the impact vector. Out of the static analysis we receive the steady state. By omitting the failure-free state of the steady state, the occurrence vector can be inferred.

$$steady\ state = \begin{bmatrix} 0.8104 \\ 0.071 \\ 0.067 \\ 0.1155 \end{bmatrix} \cong \begin{matrix} failure-free \\ Outlier \\ Noise \\ Offset \end{matrix} \Rightarrow \begin{matrix} Outlier \\ Noise \\ Offset \end{matrix} = \begin{bmatrix} 0.071 \\ 0.067 \\ 0.1155 \end{bmatrix} \cong \text{occurrence vector}$$

Figure 10: Numerical representation of the occurrence vector

The impact vector is defined in a way that the sum of all failures plus the assumption coverage (acov) produce a validity of one. An example how the impact vector might be built is given in Figure 11.

$$deviation = \begin{bmatrix} 16cm \\ 1.6cm \\ 1.9cm \\ 2.4cm \end{bmatrix} \cong \begin{matrix} Outlier \\ Noise \\ Offset \\ acov \end{matrix} \Rightarrow \begin{bmatrix} 0.51 \\ 0.13 \\ 0.16 \\ 0.2 \end{bmatrix} \cong \text{validities} \Rightarrow \begin{bmatrix} 0.51 \\ 0.13 \\ 0.16 \end{bmatrix} \cong \text{impact vector}$$

Figure 11: Numerical representation of the impact vector

The system bounds, the assessment vector and the validity vector are calculated by an element-wise multiplication ($\circledast$) of the impact vector ($impact_{vector}$) with the inverse result of the occurrence vector ($1 - occurence_{vector}$), as shown below. It needs to be noted that the lower and the upper validity of the system bounds are set to the same value. This indicates the inability of a pure sensor to detect a failure, for any further details see Section 2.1.1.

$$System_{boundsLower} = System_{boundUpper} = Assessment_{vector} = Validity_{vector} = impact_{vector} \circledast (1 - occurence_{vector})$$

3.1.2 Detection

A detection mechanism modifies the system bounds and the validity vector. The characteristics of the detector have an impact on the system bounds (design-time). During run-time, the detection will be used to calculate the validity vector. The assessment vector is not affected by a detector, compare Section 2.1.2.

By using the matrix multiplication ($\otimes$), the element-wise addition ($\oplus$) and the element-wise multiplication ($\circledast$) the system bounds are updated according to the detection characteristics. The first term of the following equation selects the failure types, which will not be detected by this mechanism. So the vector ($detection_{cap}$) holds the failure types a detector is able to detect, where one stands for unable to detect and zero stands for able to detect. This allows us to update the system bounds by using a conventional summation ($\oplus$) because a value of zero will be overwritten by the new validity bounds, otherwise the value will be taken over without any further modification. The second term sets the lower and the upper limit of the system bounds in accordance to the false positive and false negative of the detector, which will be done by a matrix multiplication ($\otimes$). The described operation has to be performed for both the lower and the upper entry of the system bounds representation.

$$system_{bound} = (system_{bound} \circledast detection_{cap}) \oplus (detection_{transion})$$

Below we show an example of the detection capability vector ($detection_{cap}$). The first row is used for failure types, which can be detected. The second row represents the case when a detector does not detect a particular failure type.

$$detection_{cap} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \cong \begin{pmatrix} able\ to\ detect \\ unable\ to\ detect \end{pmatrix}$$

In addition, the transformation matrix of a detector is depicted below. Each row affects the system bounds of a particular failure type, in this example ranging from one to ‘n’. The first row shows how the lower bound can be set to the false positive rate and the upper bound can be set to the false negative rate. The second row will be used when a detector is unable to detect this particular failure. This row serves as a neutral element in the validity calculation. The last row just highlights the dimensionality of this transformation.

$$detection_{transion} = \begin{bmatrix} FP_1 & FN_1 \\ 0_2 & 0_2 \\ \vdots & \vdots \\ FP_n & FN_n \end{bmatrix}$$

Next, we illustrate the usage of the introduced system bounds update. By using an element-wise multiplication, the first term selects system bounds, which are not affected by the detector. The remaining failure types are set to zero. Then, we make use of an element-wise addition to calculate the resulting system bounds. The transformation matrix holds zeros for failure types, which cannot be detected (second row) and otherwise the respective false positives and false negatives. Due to the neutral elements (zero rows), we are able to provide the update of the system bounds by using a conventional addition instead of a tailored operation.

$$system_{bound} = \left(\begin{bmatrix} 0.47 & 0.47 \\ 0.12 & 0.12 \\ 0.14 & 0.14 \end{bmatrix} \circledast \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 0 & 0 \\ 0.12 & 0.12 \\ 0 & 0 \end{bmatrix} \oplus \begin{bmatrix} 0.08 & 0.5 \\ 0 & 0 \\ 0.01 & 0.15 \end{bmatrix} = \begin{bmatrix} 0.08 & 0.5 \\ 0.12 & 0.12 \\ 0.01 & 0.15 \end{bmatrix}$$

So far, we explained how the system bounds are updated. Next, we present the generation of the validity vector, which states the confidence of actual sensor data during run-time. The first term of the following equation provides the same functionality as explained above. The second term is used to map a detection result on the system bounds, which finally results in the validity vector.

$$Validity_{vector} = \left(Validity_{vector} \circledast\ detection_{cap}\right) \oplus \left(\ system_{bounds} \otimes detection_{result}\right)$$

In order to give an understanding how this mapping works, we present an example of the detection transformation. The first column represents the case when no failure has been detected. Third column is used when a failure is detected and the second column serves as a neutral element when no detection result is provided.

$$detection_{result} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

An example of a validity vector is presented below. The assumed detector is only able to detect outlier and offset failures, this setting has been assumed to highlight the dynamics of the concept. The first term selects failure types the detector does not detected because those values will be taken over without any further modification. The second term maps the detection result on the system bounds. Remember the lower bound is related to the case when a failure has been detected and the upper bound is related to the case when no failure has been detected. Now we see in the interim step of the calculation that the first term and the second term are mutually exclusive. This allows us to calculate the resulting validity vector by using a simple summation ($\oplus$). Undetected failures are listed in the result of the first term. The detection result determines the result of the second term. The following calculation shows a detected offset (third row), a not detected outlier (first row) and how the validity for a respective failure type can be taken over (second row) in case a detector does not detect such failures.

$$Validity_{vector} = \left(\begin{bmatrix} 0.47 \\ 0.12 \\ 0.15 \end{bmatrix} \circledast \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}\right) \oplus \left(\begin{bmatrix} 0.08 & 0.5 \\ 0.12 & 0.12 \\ 0.01 & 0.15 \end{bmatrix} \otimes \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0.12 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} 0.5 \\ 0 \\ 0.01 \end{bmatrix}$$
$$= \begin{bmatrix} 0.5 \\ 0.12 \\ 0.01 \end{bmatrix}$$

It needs to be noticed that the presented calculation, only work for single detection stages. For instance, the validity for a sequence of sensor detector, filter and detector cannot be calculated by using the above calculation. The multiple usage of detectors can be clearly distinguished by comparing the upper system bound with the lower system bound. In case the upper system bound and the lower system bound are set to the same value, no detector has been applied so

far. If the upper system bound and the lower system bound differ from each other, another detector has been applied, which is the subject of future work.

3.1.3 Filter

The characteristics of a filter mechanism are described by the filter transformation matrix (TF), which affects the system bounds. In Figure 12, the effects of a filter on the system bounds are visualized. Mitigated failures lift the lower system bound because filtered failures occur with a smaller impact. Side effects of a filter lower the upper system bound as an indication that this failure will occur.

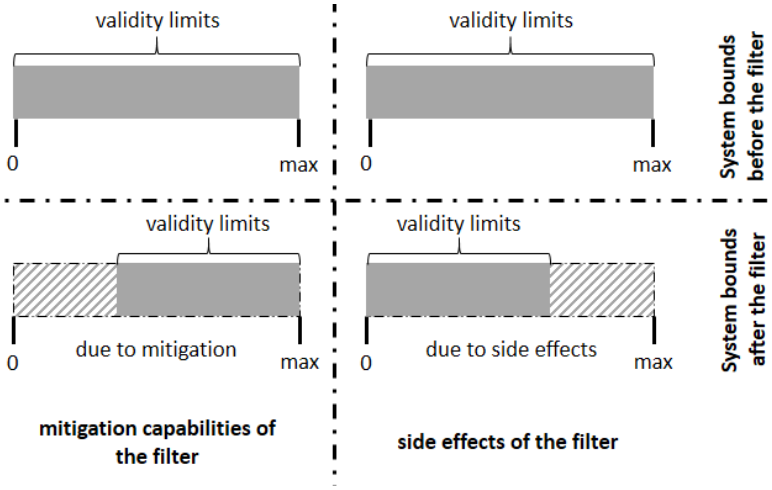


Figure 12: Filter characteristics affecting system bounds

Mitigation:

The following equation shows how the lower bound can be updated. It needs to be noticed that the resulting lower system bound ($lower_{new}$) is transposed. First of all, we extend the system bounds by an additional column holding the number one. This technique is known as homogenous coordinates and is used for expressing the effect of static parameters. How to make use of homogenous coordinates is shown in the first row of the transfer matrix (multiplicand).

$$\begin{bmatrix} lower_1 & upper\_1 & 1 \\ lower_2 & upper_2 & 1 \\ lower_3 & upper_3 & 1 \\ lower_4 & upper_4 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1-P & 1 & 0 \\ 0 & P & 0 & 1 \\ S & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} lower_{new1} \\ lower_{new2} \\ lower_{new3} \\ lower_{new4} \end{bmatrix}$$

For calculating the updated lower system bound, we select the lower system bound by multiplying with '1'. The upper system bound will be omitted when we calculate the update, therefore we multiply the upper system bound with the number zero. Now comes the advantage of homogenous coordinates, we multiply the number one with the static parameter (last entry in the first row of the transfer matrix). This results in the summation of the lower system bound with the static parameter, as illustrated below.

$$lower_{new1} = (lower_1 * 1) + (upper_1 * 0) + (1 * S) = lower_1 + S$$

The second row of the filter transformation matrix expresses the effect of a proportional parameter. In this case the lower and the upper system bounds are affected. In addition, the updated lower system bound have to range from the lower to the upper system bound as depicted in Figure 12, which requires adding the lower system bound to the resulting value once. Below we show the detailed calculation of a proportional updated lower system bound.

$$\begin{aligned} lower_{new2} &= \big(lower_2 * (1-P)\big) + (upper_2 * P) + (1 * 0) \\ &= lower_2 - lower_2 * P + upper_2 * P = lower_2 + P(upper_2 - lower_2) \end{aligned}$$

The third row of the filter transformation matrix represents the neutral element, which is used when a filter mechanism does not affect this particular failure type. In this case, we take over the lower system bound without any modification. The fourth row of the filter transformation matrix shows the elimination of a particular failure type. By setting the lower system bound to the upper system bound, the elimination is expressed. As a consequence, the validity is always set to its maximum.

Side-effects:

The side effects are expressed in a similar manner as the mitigation introduced above. Instead of selecting the lower system bound in the filter transformation matrix, we select the upper system bound to express the side effects of a filter. The following equation shows the update of the upper system bound by multiplying the system bound with the filter transfer matrix. In accordance to the mitigation example, the first row of the filter transformation matrix shows the usage of a static parameter, the second row expresses proportional effects, the third row represents the neutral element and the fourth row demonstrates the case when a failure will be present in any case. In contrast to the mitigation, the fourth row is not called elimination because side effect express negative effects and does not mitigate or suppress a failure. The depicted filter transformation matrix should be considered as an example to address the variants of the validity calculation.

$$\begin{bmatrix} lower_1 & upper_1 & 1 \\ lower_2 & upper_2 & 1 \\ lower_3 & upper_3 & 1 \\ lower_4 & upper_4 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & -P & 0 & 1 \\ 1 & P & 1 & 0 \\ S & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} upper_{new1}` \\ upper_{new2} \\ upper_{new3} \\ upper_{new4} \end{bmatrix}$$

Updating the assessment vector:

After the new system bounds have been calculated, the assessment vector is updated. Therefore, the occurrence of a failure is mapped on to the updated system bounds. First of all, the actual occurrence rate must be determined, because the actual occurrence rate can differ from the occurrence vector (OV) due to already applied filter mechanism as part of the processing chain. The following equation show how the occurrence rate can be obtained.

$$occ_{rate} = \frac{Assessment_{vec}}{impact_{vec}}$$

In the second step, the assessment vector can be calculated by multiplying the updated system vector with a transformation matrix holding the actual occurrence ratios. The first row of this transformation matrix is used when a filter mitigates a failures and the second row is used to deal with side-effects. Remember the validity of mitigated failures ranges from the lower system bound to the upper system bound, because of this the additional number one (first entry in first row) is necessary. This effect was also explained above and is depicted in Figure 12.

$$Assessment_{vecNEW} = System_{Vec} \otimes \begin{bmatrix} 1 - occ_{rate} & -occ_{rate} \\ occ_{rate} & occ_{rate} \end{bmatrix}$$

3.1.4 Application

The sensitivity of a specific application to the defined failure types is expressed by the selection vector. For each failure type a respective entry is assigned in the selection vector. In the case a failure affects the application, the respective entry in the selection vector will be then set to one and otherwise the entries in the selection vector will be set to zero. The following example shows a selection vector holding in every entry the number one. The respective application to this selection vector exhibits a sensitivity to all failure types. Such a selection vector has been used to calculate the application specific validity for Application A, compare the example in Section 2.3.

$$S_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \approx \begin{pmatrix} sensitive\ to\ outlier \\ sensitive\ to\ noise \\ senstive\ to\ offset \end{pmatrix}$$

Another example demonstrates the selection vector representative for an application, which is sensitive to outlier failures only. Therefore, the entries in the selection vector regarding noise and offset failures are set to zero. The following selection vector has been used to determine the validity for Application B, see the example in Section 2.3.

$$S_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \approx \begin{pmatrix} sensitive\ to\ outlier \\ not\ sensitive\ to\ noise \\ not\ senstive\ to\ offset \end{pmatrix}$$

3.2 Operations

In order to update and to calculate the validity representations (rep_val), we define the update operations ($\otimes$,$\circledast$,$\oplus$) and the selection operator ($\odot$). The first update operator ($\otimes$) represents a conventional matrix operation and is used to update validity representation forms like the system bounds, the assessment vector or the validity vector. Therefore, the actual validity representation form will be multiplied with the respective transformation matrix. In Section 2.1, we introduced transformations for a sensor, detector, filter and an application. In accordance to the present component, the update will be performed by using the respective transformation matrix.

The remaining update operations ($\circledast$,$\oplus$) are needed to calculate interim representations and to deal with the situation that a mechanism does not operate on all failure types. Both operations are performed element-wise, where the operation ($\circledast$) conforms a multiplication and the operation ($\oplus$) is equivalent to a summation.

Finally, the selection operator ($\odot$) is defined to calculate a validity value (V) out of a validity vector (VV). This calculation is controlled by a selection vector (S), which defines the relevant failures of an application. If all failures are considered to be relevant, the selection operator is similar to a sum of the current validities. Otherwise, the validity vector needs to be rebalanced. Because, an unbalanced validity vector would lead to a significant limited validity value even though no failure has occurred. In fact, this situation leads to an erroneously interpreted validity by the application. In the following equation, we define the rescale process using element-wise operations indicated by the dot. First of all, we have to normalize the current validities ($(VV/.FV)$), which are originally weighted according to the impact vector. Then, we have to calculate the new weights due to the selection (S). In detail, we select ($IV *. S$) the failures in the impact vector which are relevant for the application. To calculate the new weight ($(IV *. S)/. (IV * S)$), we have to divide the selected failures by the sum of selected failures. Subsequently, these weights are bound in accordance to the assumption coverage ($acov$). Otherwise, the rebalanced validities would range from zero to one, which represents the ideal case only.

$$VV \odot S = (VV/.FV)' * (acov *. (IV *. S)/. (IV * S))$$

3.3 Rules

We define operations to be commutative, associative or nothing. In case only detection mechanism has been applied, the defined operations are stated to be commutative and associative as shown below. The reason this property can be fulfilled lies in the nature of a detection mechanism, because a detector only reads the sensor data and affect the validity by detecting failures. Consequently, the order of detection does not matter.

$$VV_{d1} \oplus VV_{d2} = VV_{d2} \oplus VV_{d1}$$

$$VV_{d1} \oplus (VV_{d2} \oplus VV_{d3}) = (VV_{d1} \oplus VV_{d2}) \oplus VV_{d3}$$

In contrast, the validity calculation for filter mechanism does not reach the commutative or associative property. This fact is expressed in the equation below. A filter mechanism modifies the sensor data, which can be associated to a write operation on sensor data. Therefore, the given sensor data varies from filter to filter.

$$VV_{f1} \oplus VV_{f2} \neq VV_{f2} \oplus VV_{f1}$$

$$VV_{f1} \oplus \left(VV_{f2} \oplus VV_{f3}\right) \neq (VV_{f1} \oplus VV_{f2}) \oplus VV_{f3}$$

4. Relation between validity and uncertainty

Relying on the reliable perception of the environment raises two questions:

1. What is the typical deviation of a physical entity or environment condition in the real world from its respective digital value used by an application?
2. What is the confidence that the individual, observed value does not exceed this typical deviation?

The first question is related to uncertainty, the second to validity. Although these parameters are related this relation is not straightforward. Uncertainties may occur in the time and in the value domain and are related to measurement errors i.e. to the precision of sensor and network latency properties like steadiness and tightness [7]. Because sensor data are time/value entities and are subject of aging, there exist a mutual dependency between the latency of a network and the uncertainty of a value. In the value domain, uncertainty simply denotes the error margin. To put it more formally, a value with an uncertainty is defined as:

$$V_{observed} = V_{real} \pm \varepsilon \text{ where } \varepsilon \text{ is the accumulated uncertainty.}$$

Related to network latency this can be expressed in terms of temporal validity. A value is temporally valid if:

$$|f(t_0) - f(t_0+\Delta t_{acq} + \Delta t_{tr})| < \varepsilon,$$

where f(t0) is the real world value at the time of observation, Δt_{acq} is the time it takes to obtain a digital representation and Δt_{tr} is the time to transmit the sensor data. This equation relates temporal properties to the uncertainty margin ε.

Uncertainty is a static assumption that is based on the environment, sensor and network properties. In the data-centric view, uncertainty and failure are the same. Uncertainty specifies static bounds on an interval of deviation, but does not help much to decide about the quality and correctness of an individual value at run time. The uncertainty assumption is typically used to provide a robust control algorithm that can tolerate any deviation within the uncertainty margins. A confidence interval indicates how sure we are that the true measurement really is within the defined uncertainty margin. Therefore, an uncertainty margin has to be defined together with a confidence interval. The main problem is that a confidence interval is based on static probabilities only. There is no concept expressing how much an individual value exceeds the uncertainty bounds. Some applications may be robust to tolerate slight deviations, whereas another application may also accept more substantial deviations. Such differences can only be addressed by a more precise knowledge concerning the confidence in an uncertainty margin. Finally, a confidence interval cannot express detection or filter characteristics.

For deciding whether a value is within the defined error bounds, we introduced the concept of validity as described above. Validity quantifies the confidence that can be put in an individual sensor value. As outlined in this report, the validity value is a multidimensional mapping. It takes a failure model, assumptions about the coverage and the capabilities of detection and filter mechanisms as an input and maps it to a single value. This value is an indication of how much confidence can be put in the actual sensor value to be within assumed error margins.

It should be clear that the concepts of uncertainty and validity are different. Nevertheless there is a relation. If the demands on uncertainty are very low, i.e. a large error margin can be accepted, the confidence that a value is within these wide margins may be high. Vice versa, if there are very stringent needs with respect to the uncertainty bounds, a high validity may not be easy to achieve. A similar problem has been described as coverage stability [8] in the real-time network domain. Relaxing the demands on timeliness will increase the probability that these demands can be met.

The further elaboration of the relation between validity and uncertainty will be a topic of further investigations.

5. Conclusions and next steps

This is the second deliverable on failure semantics for distributed sensor systems. The first deliverable introduced failures modes and failure semantics as a result of careful analysis and classification of sensor failures. In D2.2, we elaborated the concept of an abstract smart sensor that provides a pair <data, validity> as output. The validity represents a uniform description that provides an estimation about the quality of information that is delivered by a sensor. This quality estimation enables the spontaneous use of external, remote sensors in a safety critical control application. This was possible mainly because of two properties. Firstly, during design-time the system validity describes the effectiveness of mechanisms to handle a set of relevant failures. Secondly, the event validity holds the outcome of these mechanisms at run-time. Although it progresses the state-of-the-art, the proposed scheme still had deficiencies because linking two different representation forms (system validity and event validity) to a safety kernel rule turned out to be not easy to achieve. Therefore we proposed to extend those measures to a failure algebra. By comparing the system validity with safety rules at design-time, we require a similar representation form to check the health status of the system at run-time. Therefore, we announced working in the direction of joining the system validity and the event validity.

In D2.5, we introduced a uniform validity description, which is in line with the terms introduced in D2.2. We defined appropriate operations as part of the failure algebra which allow us to map the outcome of a mechanism on the same validity representation, which is used at design-time. We have to note that this estimation is based on a tight relation and interplay between design time analysis and run time checking. During design-time we derive the system validity which describes and quantifies the coverage of failure assumptions and the effectiveness of mechanisms. This is captured in the assessment vector and the respective transformation matrices. At run time, the event validity is calculated by applying the quantified design time knowledge on the actual data flowing through the computational chain. The resulting data validity holds the outcome of these calculations. This overcomes the gap between design-time and run-time. We now allow a flexible assessment of the computational chain even though sensors with different quality are used or an arbitrary combination of detection or filter mechanism will be applied. Further improvements have been made in D2.5 regarding the quantification of the failure model and the applied mechanisms. The questions “How well does the failures model cover relevant failures?” and “What is the effectiveness of an applied mechanism to handle defined failures?” are explicitly expressed by the validity term. The quantification and representation of transformation information, the validity representations and the definition of appropriate operations finally form the failure algebra.

Although we made an important step towards a powerful design time analysis and run time assessment of data validity, there are still open research questions. Further investigations have to be made regarding redundancy and fusion. Also, we will direct our investigation to the notion of uncertainty. The problem is in adjusting the uncertainty margins to the requirements of different applications. Finally, an important subject of future work is to specify clearly the relation between validities and ASIL’s. The validity quantifies the confidence in the system, whereas the ASIL expresses the required quantification to reach a safety goal. These questions will guide us during the last year of KARYON.

References

[1] S. Zug, A. Dietrich and J. Kaiser, "Fault-Handling in Networked Sensor Systems," in *Fault Diagnosis in Robotic and Industrial Systems*, G. Rigatos, Ed., St. Franklin, Australia, Concept Press Ltd., 2012.

[2] K. Ni, N. Ramanathan, M. Chehade, L. Balzano, S. Nair, S. Zahedi, E. Kohler, G. Pottie, M. Hansen and M. Srivastava, "Sensor network data fault types," *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, no. 3, pp. 1-29, 2009.

[3] D. H. Stamatis, Failure mode and effect analysis: FMEA from theory to execution, Asq Press, 2003.

[4] T. Brade, S. Zug and J. Kaiser, "Validity-based failure algebra for distributed sensor systems," in *IEEE 32st Symposium on Reliable Distributed Systems*, Braga, Portugal, 2013.

[5] D. Powell, "Failure mode assumptions and assumption coverage," in *Fault-Tolerant Computing, 1992. FTCS-22. Digest of Papers., Twenty-Second International Symposium on*, 1992.

[6] J. Davis and M. Goadrich, "The relationship between Precision-Recall and ROC curves," in *Proceedings of the 23rd international conference on Machine learning*, 2006.

[7] P. Verissimo and L. Rodrigues, Distributed systems for systems architects, vol. 1, Springer, 2001.

[8] A. Casimiro and P. Verissimo, "Using the timely computing base for dependable qos adaptation," in *Reliable Distributed Systems, 2001. Proceedings. 20th IEEE Symposium on*, 2001.

Validity-based failure algebra for distributed sensor systems

Tino Brade, Sebastian Zug, Jörg Kaiser
University of Magdeburg
Institute of Distributed Systems
Magdeburg, Germany
Email: {brade,zug,kaiser}@ivs.cs.uni-magdeburg.de

***Abstract*—Distributed applications dealing with data from networked sensors need some indication about the quality of remote information. This paper describes how to derive a dynamic validity value that represents a measure for the confidence in remote sensor data. In contrast to conventional systems which treats a typical processing chain as a whole, this paper describes how individual characteristic of sensing, detection and filter mechanisms can be assessed and how this assessment can be evaluated to a single validity value. In particular the paper defines respective operations combining the run-time validity estimates of every stage in a processing chain. The combination is evaluated by rules as part of a failure algebra. The paper presents the generation of an application specific validity value which is finally demonstrated in a robotic application.**

***Keywords*-failure algebra; validity estimation; sensor systems; distributed processing chain;**

I. Introduction

Distributed computer systems interacting with the physical world which are known as cyber-physical systems (emphasizing the control aspect), ubiquitous systems, or ambient intelligence (emphasizing the spontaneous and pervasive nature) rely on the reliable perception of their environment. Smart sensors provide the application related information and typically comprise some processing and networking capabilities. For instance, robots or cars may collaborate based on using data from smart sensors that are dynamically discovered and used in order to improve and to extend the knowledge on the environment. This dynamic use raises the question of the confidence in remote sensor data in the presence of sensor failures. This problem is already difficult to handle for local sensor data [1] because sensor failures may lead to arbitrary values and often can hardly be mapped to a binary outcome of a detection mechanism. Applications may have different requirements on the precision of sensor information and sensor data that slightly differ from the correct values may still be helpful. In a previous paper we developed a classification of sensor failures [2], their impact, and how we can derive a notion of sensor data validity, which allows the application assessing the information from a distributed system of smart sensors. However, our initial scheme is not sufficient when sensor data is propagated through a chain of computational units that filter, fuse and combine simple sensor data to derive (and distribute) more complex properties of the environment. In this paper, we will deal with this problem and propose a concept to estimate the validity of sensor data in a more general way. The paper describes a calculus of how this validity can be determined when sensor data passes through a chain of computational units. A validity value is a multi-dimensional mapping. It takes a failure model, assumptions about the coverage and capabilities of detection mechanisms as an input and maps it to a single value. This value is an indication of how much confidence can be put in the associated sensor value. We argue, that for the assessment of a priori unknown remote sensor data such a validity estimate is crucial and eases the design of cooperative applications.

The starting point for using external sensors is a description of the properties contained in a sensor data sheet. This allows interpreting the remote sensor data by specifying typical data like the type, measurement range, the calibration, the degree of non-linearity and the absolute measurement error of the sensor. However, especially the information concerning the measurement error is overly pessimistic because it constitutes a static upper bound. Additionally, it represents the worst-case deviation of a correct sensor and does not cover failures. Although an application has to be aware of these sensor specific error bounds they have to be extended by assumptions about relevant failures outside the scope of sensor specifications. Existing solutions consider local sensors and often rely on replication of identical resources based on time and space redundancy. When using remote sensor information, the set of available sensors will provide a certain redundancy but will frequently change and sensor data may have different quality or even be erroneous. Therefore we cannot make a priori assumptions about sensor characteristics when defining the distributed application and fusing sensor data with unknown validity will not lead to an improvement of perception. The validity estimation will enable the application to select the most reliable information from a set of sensors.

To overcome this situation, sensor data need to be enriched with explicit information about the validity. By using validities, we were closing the gap between the static knowledge provided by data sheets during design-time and the many ways to fail during run-time. Defining the validity needs to consider the various detection and filter techniques comprised in a smart sensor. Therefore calculating a validity must consider all parts of the computational chain like the raw sensor and the subsequent failure detection and filter operations. The

dynamics of these systems further require rules to make sure under which conditions the calculated validities are consistent. Finally, we have to focus on the transfer of a calculated validity to the application. Each application might consider a different set of relevant failures as described by the failures model. For instance, some sensor data are affected by outliers, noise and offset failures, which all are captured by the validity value. The application using these data, however, needs only to be protected against outlier failures. Therefore, we have to provide means to compare the application requirements against the failures considered in the validity resulting in the generation of an application specific validity value.

The validity representation combined with operations and rules is what we call a failure algebra. The failure algebra tackles tree major issues of smart sensor applications:

- Firstly, we close the gap between the static design-time knowledge and the many ways to fail during run-time.
- Secondly, we identify the impact of a distributed detector or filter mechanism.
- Thirdly, we map the confidence of sensor data provided by the validity to the requirements of an application.

The paper is structured as follows. We start to refer the state of the art concerning the motivated issues. Then, we introduce the validity representation and define the failure algebra. The evaluation is based on a robotic scenario dealing with obstacle avoidance and navigation requirement. Finally, we conclude the present approach and point out the future work.

II. State of the art

A state of the art concerning the failure algebra does not exist. But approaches are present dealing with failures produced by the sensor. Another approaches also consider the detection techniques or the filter mechanism. This leads to different granularities on the components of a processing chain. In Fig. 1, we list the referred approaches in accordance to their considered granularity. An arrow over a component implies that the failure handling for this component is considered. Furthermore, these granularities can be linked to different views. One perspective focus on the sensor and tries to adapt against the need of an application. In this case, we direct the arrow in Fig. 1 from a sensor to the application. In contrast, approaches are found analyzing the sensor system from the application, which are illustrated by directing arrows from the application to the sensor. Another point of difference becomes substantial, by considering the design phase and the run-time phase of smart sensor applications which are shown using different arrow shapes. In order to assess the feasibility of an approach for the failure algebra, we define two criteria.

- First, the coverage criteria informs about the capability to consider all relevant failures in the concerned granularity range.
- Second, the completeness criteria provides detailed information about the covered failure types in order to assess them separately.

a) Category 1 - Sensor oriented on design-time: The definition of an uncertainty margin [4] is a general way to describe sensor failures by a static bound, encircled in Fig. 1 as number one (1). The deviation of a real sensor data to an ideal observation system determines the size of the uncertainties. The coverage criterion is only fulfilled as long as the reference system is fault-free. Because experimental designs typically suffer under random and especially systematic failures, which must be avoided for the reference system in order to satisfy the coverage criterion. But in any case, the uncertainty approach is not capable to meet the completeness criteria. Uncertainties only allow to separate between fault-free and faulty sensor data. Because there exist no possibility to separate failure types in case of faulty sensor data. An alternative approach uses signal models [5] (1), which define the allowed sensor data. Signal models consider every deviation from model-based knowledge as a failure. By the nature of this assumption, all relevant failures will be covered. But concerning the completeness, signal models provide no further information to separate the occurred failures. As a result, signal models are stated to be incomplete. A systematic approach to analysis the failures of a system is provided by Failure Mode and Effects Analysis (FMEA) [3] (2). FMEA uses three parameters to assess a system. The probability of occurred failures, the deviation of a failure compared to the intended operational state and the capabilities of the system to detect failures are expressed by parameters ranging from zero to ten. Whereas zero indicates no criticality and the number ten is used for the worst case. By multiplying these parameters, FMEA reflects the robustness of a system. The all-embracing approach of FMEA covers all failures which might happen. Therefore, FMEA satisfies the coverage criterion. But after the multiplication of the parameters, there is no chance to consider separate failure types. FMEA only reflects the most probable failure or the most deviating failure or the worst detectable failure, which leads to an incomplete representation. Zug et al. [15] (3) solves the incompleteness issue of FMEA by using a vectorized representation of each failures type. Therefore, this approach inherits the coverage criteria from the FMEA approach. Due to the vectorized representation, Zug et al. are able to fulfill the completeness criteria. The last approach sounds promising, but this approach is not applicable during run-time and does not consider the detection mechanisms of a smart sensor application. Further, it is still an open issue to fuse the vectorized representation into a single validity value.

b) Category 2 - Sensor oriented on run-time: Confidence intervals are used to inform about the reliability of sensor data. This approach fits well to distinguish between roughly disturbed sensor data and usable sensor data. For instance, Elmenreich el. al. [7] (4) uses 12 confidence classes whereas Piontek et. al. [16] (4) differentiate 16 confidence intervals. To satisfy the completeness criteria, the confidence intervals have to be detailed enough to hold the various failure types. On the other side, there is no instance making sure that all relevant failures are considered. Subsequently, both approaches cannot fulfill the coverage criteria. A similar approach is

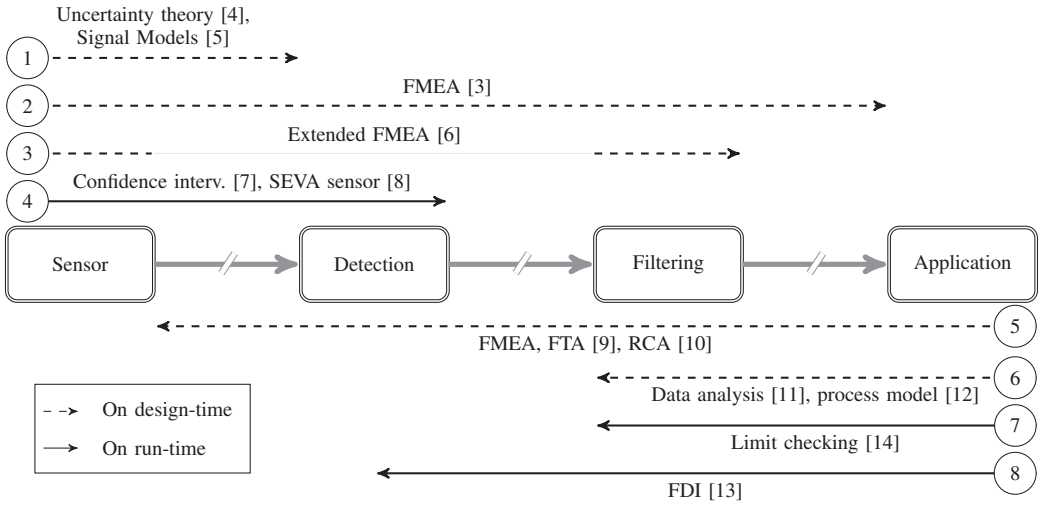


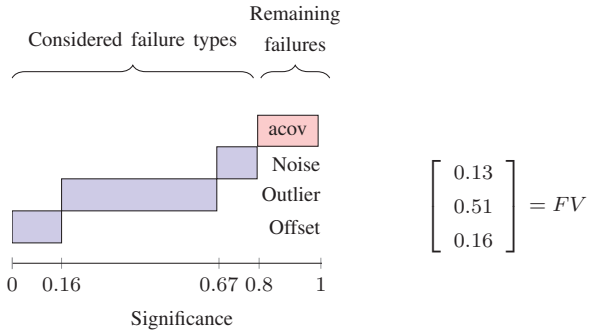
Fig. 1. Schematic representation of the state of the art

known as SEVA sensor [8] (4), which uses validities instead of the mentioned confidence intervals. Due to the representation range, the SEVA sensor fulfills the completeness criteria. But there exists also no linkage to a failure model to ensure that all relevant failures are covered. The SEVA approach does not satisfy the coverage criteria. The presented approaches are applicable during run-time but none of them present a fulfilled coverage criteria which is needed for a validity representation.

c) Category 3 - Application oriented on design-time: Techniques to analysis a system from an application oriented view are FMEA [3] (5), Fault tree analysis (FTA) [9] (5) and Root cause analysis (RCA) [10] (5). These approaches start the analysis with a malfunction on the side of the application and identify the source cause of this malfunction by analyzing each component. Therefore these techniques cover all components available in a smart sensor application. But the coverage criteria to identify all relevant failures is mainly depended on the set of malfunctions. Further these approaches do not separate failures, which is required for the completeness criteria. Another technique uses a process model [12] (6) to define the intended system behavior. Sensor data are considered to be faulty if sensor data deviate from an expected behavior which is defined by the process model. By the nature of a process model the coverage criteria is meet. Because every deviation from the intended system behavior is classified as a failure. But a clear separation between failure types is missing with respect to the completeness criteria. A critical situation occur if the process model is imperfectly defined. In such a case, reliable sensor data might be erroneously sorted out. For instance, the crash of the Airbus A-320 was due to a wrong interpretation of sensor data which finally blocks to switch from the flight mode to the landing mode [17]. An alternative technique to assess incoming sensor data is known as data analysis [11] (6). This approach uses orthogonal transformations to convert a correlated set of data into a linearly uncorrelated set of data. The transformation does not rely on signal model and is able to reduce the incoming dimension even if the sources are superimposed by failures. Regarding the coverage and completeness criteria, the data analysis is able to provide both but the usefulness of data analysis is limited by the set of incoming sensor data. The presented approaches are able to partly fulfill the coverage criteria and the completeness criteria without any further information regarding the sensor. This underlines our motivated aim. But these approaches are struggling to separate the failure types, which is a need for the validity representation. On the other side, the presented approaches are only applicable during design-time.

d) Category 4 - Application oriented on run-time: A failure classification without detailed knowledge on the sensing device is known as limit checking [14] (7). By employing the limit checking approach a threshold is used to separate reliable data from failures. To set this threshold, the knowledge about the uncertainty of sensor data can be exploit. But this technique only works for applications without dynamics. The coverage criteria is limited by the adjustment of the right threshold. Due to an unclear separation of single failure types the completeness criteria is not satisfied. Another technique to deal with failures is known as Fault detection and isolation (FDI) [13] (8). FDI uses the initial idea of a process model to describe the intended behavior of a control loop, which takes also the detection and the filtering into account. Due to the same basic approach, FDI satisfies the coverage criteria. But the degree of failure separation is application depended and not generally supported. It is application specific, whether FDI is adequately fine grained to be stated as complete.

In the end, none of the referred approaches are able to fulfill the coverage as well as the completeness criteria by considering all components of a smart sensor application. Further, the representation form we are aiming for needs to be applicable both on design-time and on run-time, which could not be found. Because of that, we introduce in the following section a representation scheme satisfying those criteria.



(a) Abstract representation of the failure significances (b) Failure vector (FV)

Fig. 3. Definition of the failure vector (FV)

III. Validity representation

The validity of sensor data describes the level of confidence an application can have in these data. A validity of zero indicates the least and a validity of one indicates the highest confidence in the sensor data, respectively. In order to calculate the validity of sensor data, we have to consider the components of a sensor system like sensor, detector, filter and the application need. In detail, the validity of sensor data is affected by the characteristics of a detector or a filter mechanism. Because of that, we have to describe these characteristics in order to calculate the validity of sensor data, which propagates through a processing chain of several detection and filter mechanisms. This leads us to static validity bounds within the validity will vary during run-time. Further, these representation forms needs to be linked to a failure model informing about the considered failure types and their respective properties. Each representation form is illustrated in Fig. 2, which highlights a general sensor application given as an example. In fact, our approach is not limited to a specific set or a certain order of detection and filter mechanisms.

A. Failure vector

The failure vector lists the relevant failures of the smart sensor application. The key to build a failure vector are the definition of a failure model and the significance of each failure. It needs to be noticed that each model shows a certain degree of imperfection. The degree of imperfection is reflected by the assumption coverage.

Assumption coverage reflects the ratio between the considered failures and the entire set of possible failures. The complete set of failures is rather difficult to define. It could be specified either by an analytical breakdown or by an empirical approach. If the details of the system are known the analytical approach could be utilized as it is the case for mechanical engineering purposes [18]. Because of the many degrees of uncertainty and the lack of complete knowledge of the physical characteristics of a smart sensor, we choose to the empirical approach.

In [19], we elaborated a failure model to cover failures of a smart sensor application. To figure out the significance of the defined failure types, we use a modified FMEA technique [2]. Frequent failures with a large deviation from the correct value get the highest significance. Respectively, sporadic failures with slight deviations from the correct value have a smaller impact.

Using the knowledge about the failure model and the significances, we define the failure vector as an m-dimensional vector filled with the significances of the m failure types defined by the failure model. The sum of the failure vector is inherently limited by the assumption coverage, which specifies the imperfection of the used failure model. In Fig. 3, we show an example dealing with offset, outlier and noise failures. The corresponding significances are illustrated by the width of bars, in Fig. 3a. Whereas the acov bar identifies the assumption coverage. The corresponding failure vector is specified in Fig. 3b.

B. System vector

The system vector (SV) specifies an anticipated validity range in which the output of a processing chain will vary during run-time. Without loss of generality, we assume that a processing chain consists of exactly one sensor and several detection as well as filter mechanisms. Because due to the nature of a detector or filter mechanism, they are used to operate on exactly one set of sensor data.

First of all, we initialize the system vector with the failure types and significances described by the failure vector. By using the characteristics of each component in the processing chain, the system vector is updated while propagating through the processing chain. In detail, the system vector assigned to the input informs about the anticipated validity of the sensor data, which is fed to a component. Whereas the system vector is updated in order to express the effect on the validity due to employing a component. Consequently, an application is enabled by using the system vector to sort out inadequate sensor data in advance. The following paragraphs address the impact of the detector and filter characteristics on the system vector.

a) Detector characteristics: The detection characteristics identify the capabilities of a detection mechanism to detect a failure. Considering a detector there may be the situation that a detector detects a failure, even though no failures are present. Vice versa, failures are present and a detector does not detect a failure. These situations can best be expressed by the notion of true positive, false positive, true negative and false negative [20]. A true positive represents a real failure, which was accurately detected. In the case of a false positive, the detector has erroneously identified a failure. A true negative describes the correct outcome, when no failure occurred. Finally, the false negative reflects the case that a failure occurred but was not detected.

We use these notions to update the system vector. In an ideal system where no false positives and false negatives appear, we would expect to receive a validity of zero in case of a detected failure and otherwise a validity of one. But in the case of false positives or false negatives, the validity range needs to be

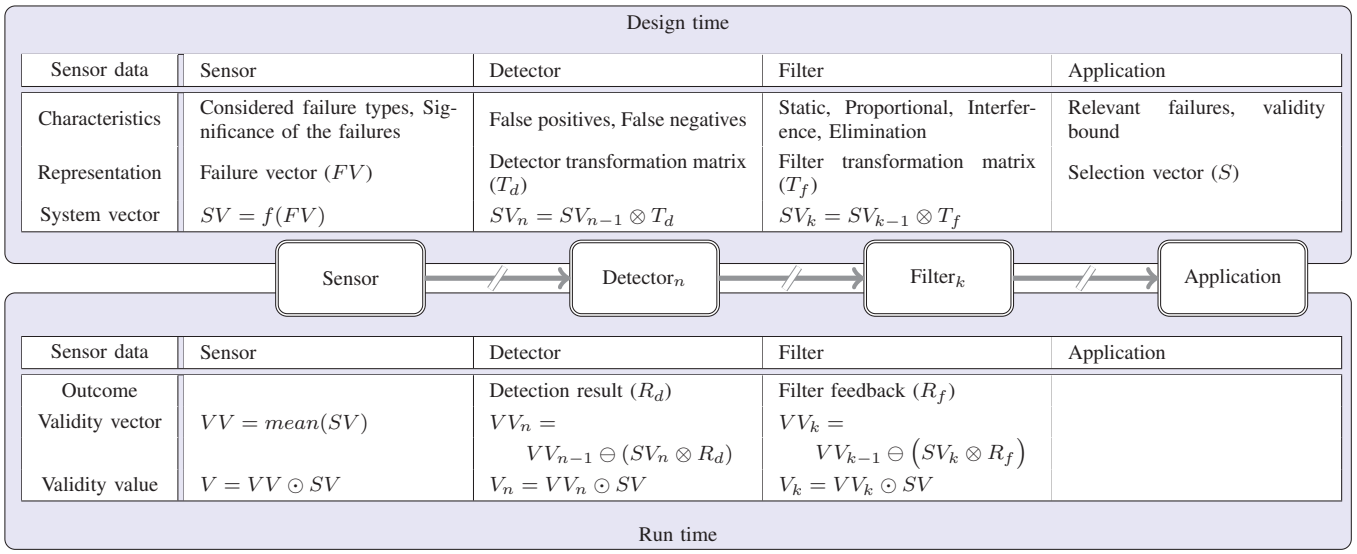


Fig. 2. Schematic representation of the validity representation

restricted. This allows us to consider the remaining confidence in sensor data although it was detected as a failure by the detector. The remaining confidence is given by the probability of false positives. Therefore, the validity of a failure type will not drop to zero even if a failure is detected. The percentage of the false positive (FP) represents the lower bound of the validity range. The false negative (FN) represents the uncertainty that a failure is present but not detected by the detector. Taking into account these uncertainties, the upper bound of the validity range is given by the percentage of false negatives. This allows to describe the uncertainty of a failure that remained undetected.

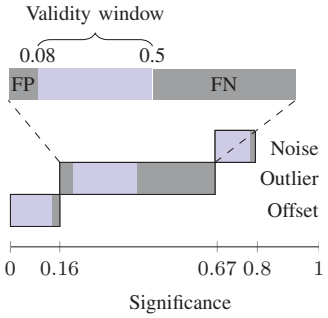
$$T_d = [T_{dmin}; T_{dmax}] \tag{1}$$

$$T_{dmin} = \begin{bmatrix} 1 - FP_1, & \dots, & 1 - FP_m \\ FP_1 & , \dots, & FP_m \end{bmatrix} \tag{2}$$

$$T_{dmax} = \begin{bmatrix} -FN_1, & \dots, & -FN_m \\ FN_1 - 1, & \dots, & FN_m - 1 \end{bmatrix} \tag{3}$$

In order to update the system vector, we define a transfer matrix T_d holding the detection characteristics, which is shown in Eq. 1. First, we modify the lower bound of the system vector by using the false positives (FP) defined in Eq. 2. Second, the upper bound of the system vector is adjusted according to the false negatives (FN) specified in Eq. 3. In detail, FP_1 and FN_1 corresponds to the first failure type whereas the FP_M and FN_M belongs to the m-th failure type in accordance to the failure model.

By applying this knowledge to the afore-mentioned example, in Fig. 4a we depict the validity restriction due to false positives (FP) and false negatives (FN) on the outlier failure type. Furthermore, we show the effect of detection



(a) Restricted system vector due to false positives (FP) and false negatives (FN) of the detection

$$\begin{bmatrix} 0.01 & 0.12 \\ 0.08 & 0.5 \\ 0.01 & 0.15 \end{bmatrix} = SV$$

(b) System vector (SV)

Fig. 4. Definition of the system vector (SV)

characteristics on the system vector in Fig. 4b.

b) Filter characteristics: The filter characteristics expresses the impact of a filter mechanism on the sensor data. After a detection mechanism identified a failure, a smart sensor application usually employs filter techniques to mitigate the adverse effect of a failure. But a filter technique may come along with considerable side effects. For instance, an outlier failure filtered by using a moving average technique will add an offset and also a delay to the original signal.

To express the characteristics of a filter technique, we use parameters to describe the effect of the filter on each failure type. A filter technique may affect the validity of a failure type in a static fashion, in a proportional way or leads to an elimination of a failure type. The static parameter is used to express a modification by a fixed number. Static parameters are well suited to describe delay effects. The proportional parameter is generally used to express the mitigation of a failure due to a filter method. In this case, the validity will be affected by a percentage. The capability of a filter to suppress

the effect of a failure type is expressed by the elimination parameter. An elimination might occur if the non-linearity of a sensor is flawless calibrated.

$$T_f = [T_{fmin}; T_{fmax}] \tag{4}$$

$$T_{fmin} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & \dots & M \end{matrix} \\ \begin{pmatrix} 0 & -P & 1 & \dots & 1 \\ 1 & P & 0 & \dots & 0 \\ 0 & 0 & S & \dots & 0 \end{pmatrix} \end{array} \tag{5}$$

$$T_{fmax} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & \dots & M \end{matrix} \\ \begin{pmatrix} 1 & -P & 0 & \dots & 0 \\ 0 & P & 1 & \dots & 1 \\ 0 & 0 & -S & \dots & 0 \end{pmatrix} \end{array} \tag{6}$$

In Eq. 4, we assign the filter characteristics (T_f) in order to alter the lower and upper validity bound given by the system vector (SV). In detail, we express the positive effect a filter mechanism can have by raising the lower validity bound up. The respective matrix representation is defined in Eq. 5. Each row in this matrix affects a respective failure type. For instance, we show with the first row the elimination of a failure. The second row is used for proportional alterations. In the third row, we show a static parameter using a homogenous coordination. A neutral element is specified by the M-th row, which is used if the filter mechanism do not affect a particular failure type. The same parameter set is applied for the upper validity bound in Eq. 6, which is to express the negative effect a filter mechanism will have. To sum up, the Eq. 5 is used to increase the validity and to express the mitigation of a occurred failure due to the filter operation. With respect to the negative side effects of a filter mechanism, we defined Eq. 6 in order to drop the upper bound of the validity down.

C. Validity vector

The validity vector informs about the current validity of the sensor data during run-time. Based on the validity bounds given by the system vector, we determine the validity vector in consideration of the outcome of the used detection and filter mechanisms. In fact, the system vector defined a range of validities taking into account the knowledge of the failure model and the characteristics of the applied detection and filter mechanisms. This knowledge is used to determine the precise validity according to each failure type while run-time.

First of all, the validity vector needs to be initialized using the outcome of a component. Concerning a detector the detection result will be taken, but the validity vector is assigned to the sensor, which does not have a comparable outcome. The initialization of the validity vector is rather problematic, because there is no knowledge or mechanism available in a pure sensing device informing about the validity on run-time. A plausible argumentation is to initialize the validity vector with the mean value of the validity bound defined by the system vector. Such an argumentation is neutral and will be neither interpreted as a failure nor considered to be failure-free.

By focusing on the detection, the outcome of a detection mechanism will be either: failure is detected or failure is not detected. In Eq. 7, we show how an outcome of a detector could be represented. In an ideal system, we expect to receive a high validity when no failures occur and in case of failures a decrease of the validity. This idea can be implemented by taking the upper bound of the system vector if the detector does not detect any failure. On the other side, a detected failure leads to a validity, which is given by the lower bound of the system vector. Therefore, the first row in Eq. 7 selects the upper bound of the system vector. Whereas the second row chooses the lower bound of the system vector. The inability to detect a certain failure type is shown by the M-row, which leads to a negative outcome and clearly separates the cases not detected failure (1-row), detected failure (2-row) and unable to detect a failure (M-row).

$$R_d = R_f = \begin{array}{c} \begin{matrix} 1 & 2 & \dots & M \end{matrix} \\ \begin{pmatrix} 0 & 1 & \dots & 0 \\ 1 & 0 & \dots & -1 \end{pmatrix} \end{array} \tag{7}$$

Smart sensor applications trust in the capabilities of filter mechanisms to suppress the impact of a failure. Following this idea, we argue that a filter technique is used to boost the validity. The outcome of a filter mechanism (R_f) is represented accordingly to the detection defined in Eq 7.

D. Validity value

The validity value is an application specific generated value out of the validity vector. In general, an application is incapable to exploit the details of the validity vector. The application is mainly interested in a single value informing about the confidence of the sensor data. Exactly, this need is fulfilled by the validity value. The generation of the validity value has to distinguish two cases.

First, the set of relevant failures considered by the failure model is larger than the set of relevant failures of the application. In this case, we have to sort out the failures in the validity vector which are not relevant for the application. To generate the validity value, we select and rebalance the relevant failures in a way that the significances of the selected failures are bounded by the assumption coverage. Subsequently, to receive the validity value, we sum up the validity vector.

Second, the set of relevant failures considered by the failure model is lower than or equal to the set of relevant failure of the application. If the assumption coverage of the failure model is close to one, we generate the validity value by calculating the sum of the validity vector. But if the assumption coverage is expected to be significant lower than one, we have to deal with a serious issue. Because there may exist an unconsidered failure in the failure model, which is considered as a relevant failure by the application. In this situation, the generation of an application specific validity value is prohibited.

IV. Failure algebra

The failure algebra is to calculate the impact a certain detector or filter might have on the validity of sensor data. We start to define the failure algebra and specify the operations to transform the validity representation forms described in Sec. III. Furthermore, we present rules for these operations in order to describe the conditions under which the validity calculation will be consistent and reproducible.

A. Definition

The failure algebra is defined by a set of validity representations (val_rep), a set of transformations ($trans$) and a set of operations as shown in Eq. 10. The validity representation (val_rep) ranges from zero to the assumption coverage, which limits the validity representation due to the imperfection of the underlying failure model. In Eq. 8, the validity representation is defined as a set of the failure vector (FV), system vector (SV), validity vector (VV), validity value (V), which are described in Sec. III. The transformation ($trans$) holds the necessary information in order to update and to transform the validity representation specified in Eq. 9.

$$rep\_val = \{FV, SV, VV, V\}, rep\_val = [0, acov] \in \Re \quad (8)$$

$$trans = \{S, T_d, T_f, R_d, R_f\}, trans \in \Re \quad (9)$$

$$(\{rep\_val, trans\}, \otimes, \ominus, \oplus, \odot) \quad (10)$$

B. Operations

In order to update and to calculate the validity representations (rep_val), we define the update operator ($\otimes$), the detection operator ($\ominus$), the filter operator ($\oplus$) and the selection operator ($\odot$).

The update operator ($\otimes$) conforms to a matrix multiplication and is used to update the system vector and to calculate the validity vector. Second, the detection operator ($\ominus$) updates the validity vector using the detection result. We define in Eq. 11, that a detection result (VV_d) will overwrite a validity, which is initialized by a sensor (VV_i) or is changed due to a filter mechanism (VV_f). This definition holds because a detector informs about the existence of a failure, which is neither achieved by an initialization nor by a filter mechanism. In the case of an already performed detection shown in Eq. 12, we apply the mean of two detector outputs (VV_{d1}, VV_{d2}) weighted according to their false positives and false negatives given by the system vector (SV_1, SV_2). This prevents a distortion of an accurate detector by a less precise detector. By considering Eq. 13, a negative validity ($-VV_2$) does not affect an existing validity (VV_1). Because a negative validity indicates that a detector or filter does not operate on a particular failure type. Third, the filter operator ($\oplus$) affects an existing validity (VV) according to a mathematical sum, as specified in Eq. 14.

$$VV_i \ominus VV_d = \qquad VV_f \ominus VV_d = VV_d \quad (11)$$

$$VV_{d1} \ominus VV_{d2} = \qquad mean_{SV1,SV2}(VV_{d1}, VV_{d2}) \quad (12)$$

$$VV_1 \ominus -VV_2 = \qquad VV_1 \oplus -VV_2 = VV_1 \quad (13)$$

$$VV \oplus VV_f = \qquad sum(VV, VV_f) \quad (14)$$

Finally, the selection operator ($\odot$) is defined to calculate a validity value (V) out of a validity vector (VV). This calculation is controlled by a selection vector (S), which defines the relevant failures of an application. If all failures are considered to be relevant, the selection operator is similar to a sum up of the current validities. Otherwise, the validity vector needs to be rebalanced. Because, an unbalanced validity vector would lead to a significant limited validity value even though no failure has occurred. In fact, this situation leads to an erroneously interpreted validity by the application. In Eq. 15, we define the rebalance process using elementwise operations indicated by the dot. First of all, we have to normalize the current validities ($VV/.FV$), which are originally weighted according to the failure vector. Then, we have to calculate the new weights due to the selection (S). In detail, we select ($FV * .S$) the failures in the failure vector which are relevant for the application. To calculate the new weight ($(FV * .S)/.(FV * S)$), we have to divide the selected failures by the sum of selected failures. Subsequently, we have to bound these weights according to the assumption coverage ($acov$). Otherwise, the rebalanced validities would range from zero to one, which is only representing the ideal case.

$$VV \odot S = v = (VV/.FV)' * (acov * .(FV * .S)/.(FV * S)) \quad (15)$$

C. Rules

We define operations to be commutative, associative or nothing of both. The detection operation $\ominus$ is stated to be commutative as well as associative as shown in Eq. 16 and 18. The reason this properties can be fulfilled lies in the nature of a detection mechanism. Because a detector only reads the sensor data and affect the validity by detecting failures. Consequently, the order of detection does not matter. In contrast, the filter operation $\oplus$ can never reach the commutative or associative property. This fact is expressed in Eq. 17 and 19. A filter mechanism modifies the sensor data, which can be associated to a write operation on sensor data. Therefore, the given sensor data varies from filter to filter.

$$VV_1 \ominus VV_2 = VV_2 \ominus VV_1 \quad (16)$$

$$VV_1 \oplus VV_2 \neq VV_2 \oplus VV_1 \quad (17)$$

$$VV_1 \ominus (VV_2 \ominus VV_3) = (VV_1 \ominus VV_2) \ominus VV_3 \quad (18)$$

$$VV_1 \oplus (VV_2 \oplus VV_3) \neq (VV_1 \oplus VV_2) \oplus VV_3 \quad (19)$$

TABLE I
DEFINITION OF THE FAILURE MODEL

	Outlier	Noise	Offset	acov
Deviation (cm)	16	1.6	1.9	2.4
Significance (per cent)	51	13	16	20

V. SCENARIO

We evaluate the proposed failure algebra by using a general robotic application. The first task deals with obstacle avoidance, whereas the second task is to determine the position of the robot. Regarding the used equipment to master these tasks, we employ an infrared distance sensor (SHARP GP2D12 [21]) to observe the surroundings of the robot. Because of several failures on the delivered sensor data, we add a detection as well as a filter mechanism to this smart sensor application. By using the validity representation, we address the impact a detector and a filter will have and how to exploit this knowledge. Considering this composition, we guide the reader through the different validity representations and demonstrate the usage of the defined operations.

A. Failure vector

In order to keep our failure model manageable, we solely consider outlier, noise and offset failures, which lead us to an assumption coverage of 80 per cent. The remaining 20 per cent are due to unconsidered failures like delay, drift and etc. The precise significances of this failures types are determined by a reference system fusing laser, sonar and camera results [22]. In Tab. I, we specify the deviation and the significance for each failure type. By using this knowledge, we assign in Eq. 20 the failure vector (FV) for our smart sensor application. The defined failure vector represents the significance of an outlier, noise and offset failure.

$$FV = \begin{bmatrix} 0.51 \\ 0.13 \\ 0.16 \end{bmatrix} \tag{20}$$

B. System vector

The system vector (SV) is initialized by the failure vector (FV) and ranges from zero to the defined significance, as shown in Eq. 21. In detail, a failure lead to a validity of zero. Respectively, the validity is set to the defined significance if no failure occur.

Because of these failures, we employ a detector as well as a filter mechanism. To determine the effect on the validity, we elaborate the respective characteristics by simulations combined with failure injection procedures [23].

$$SV_i = f(FV) = \begin{bmatrix} 0 & 0.51 \\ 0 & 0.13 \\ 0 & 0.16 \end{bmatrix} \tag{21}$$

TABLE II
CHARACTERISTICS OF THE HIDDEN MARKOV MODEL (HMM) DETECTOR

	Outlier	Noise	Offset
False positive (per cent)	0.17	0.07	0.01
False negative (per cent)	0.01	0.02	0.01

TABLE III
CHARACTERISTICS OF THE KALMAN FILTER

Parameter	Outlier	Noise	Offset
Static	0	0	-4
Proportional	4.4	7	0
Elimination	0	0	0

Considering the detection mechanism, we implement a hidden markov model (HMM) [24] in order to detect the defined failures. The HMM shows false positives as well as false negatives as described in Tab. II. The set of false positives and false negative lead to a bounding of the validity range as expressed by the system vector (SV_n) in Eq. 22.

$$SV_n = SV_i \otimes T_d = \begin{bmatrix} 0.08 & 0.5 \\ 0.01 & 0.12 \\ 0.01 & 0.15 \end{bmatrix} \tag{22}$$

Our evaluated smart sensor application uses a Kalman filter [25] to mitigate the impact of failures. The characterization of the filter mechanism is specified in Tab. III. By applying this characteristics to update the system vector, we receive an increased validity for outlier and noise failures. But the validity regarding offset failures drops slightly down due to the auto-regressive implementation. The updated system vector (SV_k) presented in Eq. 23 describes the anticipated validity as a result of the filter.

$$SV_k = SV_n \otimes T_f = \begin{bmatrix} 0.35 & 0.5 \\ 0.07 & 0.12 \\ 0.01 & 0.11 \end{bmatrix} \tag{23}$$

The presented update of the system vector can be summarized as the following. First, the initialized system vector (SV_i) specifies the validity range and points out the possible failures. Assuming an ideal detector this validity range will not be touched. Because the set of failures on sensor data stays the same due to a detector. A detector rather informs about a failure. Therefore, we receive a slightly limited validity range (SV_n) because of the false positives and false negative. A possibility to significantly increase the validity are filter techniques. A filter is able to suppress a failure, which leads to a raised validity (SV_k). This can be interpreted as a validity guarantee, which will be maintained during run-time.

C. Validity vector

The validity vector informs about the current validity on run-time. In Fig. 5a, we depict the raw sensor data obtained by an SHARP GP2D12 infrared distance sensor. These sensor data are fed to the HMM in order to detect outlier, offset and noise failures. The outcome of the HMM is visualized in Fig. 5b. We receive three detected outlier failures, which can be seen by the accordingly decreased validity. Then, we notice an offset failure between timestamp 150 and 200, which is due to an improper calibration. Moreover, we have to deal with noise failures occurred sporadically. In order to express these detection results, we use the validity bounds predefined by the system vector. By applying this knowledge, we obtain the validity vector (VV) shown in Fig. 5b.

D. Validity value

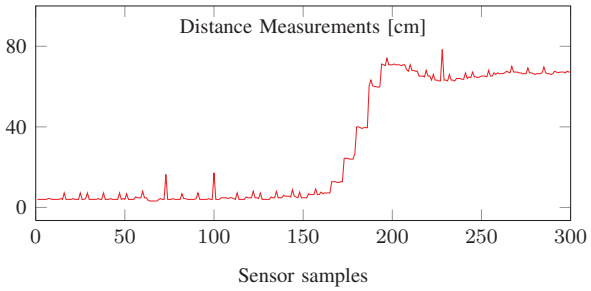
Finally, we generate a one-dimensional validity value out of the m-dimensional validity vector. By analyzing the sensitivity of our robotic applications against the defined failures, we achieve a specific selection vector (S) for each application. These selection vectors specify the set of relevant failures and are used to generate the validity value. Considering the obstacle avoidance task, we identified outlier failures to be relevant. Whereas noise and offset failures do not have a significant impact. This requirement is expressed by the selection vector (S_{ob}) in Eq. 24. In Fig. 5c, we show the resulting validity value which is fed to the obstacle avoidance implementation. Therefore, the validity only drops down if an outlier occur.

Regarding the second task which is responsible for navigation, we figured out a sensitivity against all defined failures. This leads to the selection vector (S_{nav}) defined in Eq. 25. The resulting validity value is illustrated in Fig. 5c, where the validities of the validity vector are summed up.

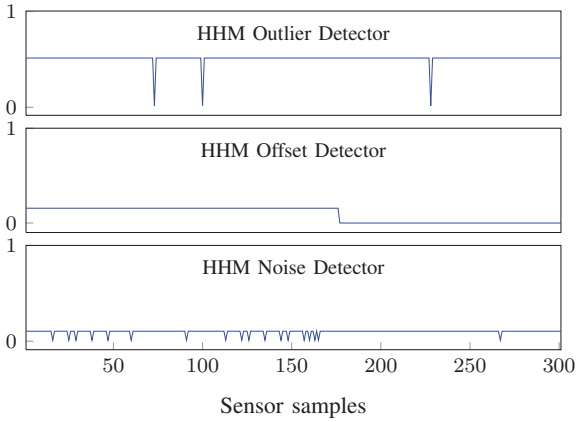
In the end, both applications are explicitly informed about the confidence of sensor data. The analysis of the applications identified relevant failures, which match with the failures defined by the failure model. Therefore, the employed sensor cannot be used to perform the considered tasks without any further detection or processing. Using the validity representation, we mastered the mismatch of sensor and application. Because the validity informs the application whether the current sensor data can be used.

$$S_{ob} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{24}$$

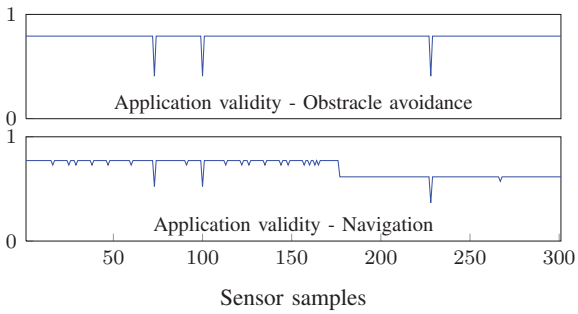
$$S_{nav} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{25}$$



(a) Raw sensor data of SHARP GP2D12 IR-distance sensor



(b) Validity vector informing about outlier, offset and noise failures



(c) Validity values fed to the navigation task and to perform obstacle avoidance

Fig. 5. Evaluation using validities for navigation and obstacle avoidance based on distance sensor data

VI. Conclusion

In this paper, we described a new approach to calculate the validity for distributed sensor systems. The proposed approach is not limited to a specific set or a certain order of processing components e.g. detection or filter mechanisms. Consequently, an implicit assumption to handle sensor failures is not needed anymore. In detail, our approach informs about the current confidence an application can have in remote obtained sensor data. In addition to the sensor data, we provide a validity value, which can be linked to an application need. Because each application might consider a different set of relevant failures. Our approach consists of:

- First, an appropriate representation describing the validity of a processing chain.

- Second, a definition of operations on the validity combined with rules for the calculus.

Both are necessary to calculate the validity of sensor data, which propagates through a distributed processing chain equipped with several detection and filter mechanisms. But this calculation requires a detailed knowledge on the processing chain characteristics, which is determined on design-time and leads to static bounds on the validity. In order to exploit this knowledge for the validity calculation, we also have to consider the outcome of a processing chain, which is available on run-time. At this point, we profit from the defined failure algebra operations to bridge the gap between both worlds. In fact, the representations can only be used in conjunction with a failure model, where the relevant failures are defined and the calculated validity gets comparable and interoperable.

The key to broaden the usage of the failure algebra is an enlargement of operations. Future work will investigate on the fusion of validities fed by different sensors. Because the proposed failure algebra only deals with sequential processing chains. Further, the consideration of redundant sensor configurations as a special case of the sensor fusion seems to be promising. Another research will be directed on the validity calculation of two detection results, where the current mean calculation can be recognized as an interim solution. Then, the system vector needs to be idempotent to deal correctly with a repeated detection of the same implementation. Because a detector does not gets more precise if this particular mechanism is repeated. Additional research will investigate on the initialization of the validity vector, where the mean of the failure vector is not a satisfactory solution. Finally, we will provide an implementation of the failure algebra integrated as a framework in Simulink.

ACKNOWLEDGMENT

This work has been supported by the EU under the FP7-ICT programme, through project 288195 Kernel-based ARchitecture for safetY-critical cONtrol(KARYON).

REFERENCES

[1] K. Marzullo, "Tolerating failures of continuous-valued sensors," *ACM Transactions on Computer Systems (TOCS)*, vol. 8, no. 4, pp. 284–304, 1990.

[2] J. Kaiser and S. Zug, "A fault-aware sensor architecture for cooperative mobile applications," in *2012 IEEE 26th International Parallel and Distributed Processing Symposium (IPDPS)*, Shanghai, China, 5 2012.

[3] D. H. Stamatis, *Failure Mode and Effect Analysis: Fmea from Theory to Execution*. Milwaukee,: ASQ Quality Press, 2003.

[4] R. J. Moffat, "Describing the uncertainties in experimental results," *Experimental Thermal and Fluid Science*, vol. 1, no. 1, pp. 3 – 17, 1988. [Online]. Available: http://www.sciencedirect.com/science/article/pii/089417778890043X

[5] A. Dietrich, S. Zug, and J. Kaiser, "Detecting External Measurement Disturbances Based on Statistical Analysis for Smart Sensors," in *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE)*, 7 2010, pp. 2067–2072.

[6] ——, "Model based Decoupling of Perception and Processing," in *ERCIM/EWICS/Cyberphysical Systems Workshop, Resilient Systems, Robotics, Systems-of-Systems Challenges in Design, Validation & Verification and Certification*, Naples, Italy, 9 2011.

[7] H. Kopetz, M. Holzmann, and W. Elmenreich, "A universal smart transducer interface: TTP/A," in *Third IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*. Newport Beach, California: IEEE, 3 2000, p. 16.

[8] M. Duta and M. Henry, "The fusion of redundant seva measurements," *Control Systems Technology, IEEE Transactions on*, vol. 13, no. 2, pp. 173–184, 2005.

[9] W. E. Vesely and N. Roberts, *Fault tree handbook*. Nuclear Regulatory Commission, 1987.

[10] E. B. Jensen, "Root cause analysis," *Compendium for use by Patient*, 2004.

[11] Y. Huanga, J. Gertler, and T. J. McAvoy, "Sensor and actuator fault isolation by structured partial pca with nonlinear extensions," *Journal of Process Control*, pp. 459–469, 2000.

[12] W. Bolton and W. Bolton, *Control engineering*. Longman, 1998.

[13] R. Isermann, *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*. Springer Verlag, 2006.

[14] T. Yairi, M. Nakatsugawa, K. Hori, S. Nakasuka, K. Machida, and N. Ishihama, "Adaptive limit checking for spacecraft telemetry data using regression tree learning," in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, vol. 6. IEEE, 2004, pp. 5130–5135.

[15] S. Zug, T. Brade, J. Kaiser, and S. Potluri, "An approach supporting fault-propagation analysis for smart sensor systems," in *Computer Safety, Reliability, and Security*. Springer Berlin Heidelberg, 2012, pp. 162–173.

[16] H. Piontek and J. Kaiser, "Self-describing devices in cosmic," in *Proc.: 10th IEEE International Conference on Emerging Technologies and Factory Automation*, Catania, 2004.

[17] Main Commission Aircraft Accident Investigation Warsaw, available on http://www.rvs.uni-bielefeld.de/publications/Incidents/DOCS/ComAndRep/Warsaw/warsaw-report.html, 9 1993.

[18] K. Chintalapudi, T. Fu, J. Paek, N. Kothari, S. Rangwala, J. Caffrey, R. Govindan, E. Johnson, and S. Masri, "Monitoring civil structures with a wireless sensor network," *Internet Computing, IEEE*, vol. 10, no. 2, pp. 26–34, 2006.

[19] S. Zug, A. Dietrich, and J. Kaiser, *Fault Diagnosis in Robotic and Industrial Systems*. St. Franklin, AUS: Concept Press Ltd., 2012, ch. Fault-Handling in Networked Sensor Systems.

[20] J. Davis and G. Mark, "The relationship between precision-recall," in *Proceedings of the 23rd international conference*. ACM, 2006, pp. 233–240.

[21] Sharp Cooperation, *GP2D120 Data Sheet*, 2007, http://sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y0a21yk_e.pdf.

[22] S. Zug, F. Penzlin, A. Dietrich, T. T. Nguyen, and S. Albert, "Are Laser Scanners Replaceable by Kinect Sensors in Robotic Applications?" in *IEEE International Symposium on Robotic and Sensors Environments (ROSE 2012)*, Magdeburg, Germany, 2012.

[23] M.-C. Hsueh, T. K. Tsai, and R. K. Iyer, "Fault injection techniques and tools," *Computer*, vol. 30, no. 4, pp. 75–82, 1997.

[24] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[25] R. E. Kalman *et al.*, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.