Using Safety Contracts to Verify Design Assumptions During Runtime

Omar Jaradat (✉) and Sasikumar Punnekkat

School of Innovation, Design and Engineering
Mälardalen University, Västerås, Sweden
{omar.jaradat,sasikumar.punnekkat}@mdh.se

Abstract. A safety case comprises evidence and argument justifying how each item of evidence supports claims about safety assurance. Supporting claims by untrustworthy or inappropriate evidence can lead to a false assurance regarding the safe performance of a system. Having sufficient confidence in safety evidence is essential to avoid any unanticipated surprise during operational phase. Sometimes, however, it is impractical to wait for high quality evidence from a system's operational life, where developers have no choice but to rely on evidence with some uncertainty (e.g., using a generic failure rate measure from a handbook to support a claim about the reliability of a component). Runtime monitoring can reveal insightful information, which can help to verify whether the preliminary confidence was over- or underestimated. In this paper, we propose a technique which uses runtime monitoring in a novel way to detect the divergence between the failure rates (which were used in the safety analyses) and the observed failure rates in the operational life. The technique utilises safety contracts to provide prescriptive data for what should be monitored, and what parts of the safety argument should be revisited to maintain system safety when a divergence is detected. We demonstrate the technique in the context of Automated Guided Vehicles (AGVs).

Keywords: Confidence, safety contracts, safety case, safety argument, monitoring, runtime, failure rate, probability of failure, through-life safety assurance

1 Introduction

Safety critical systems are those systems whose failure could result in loss of life, significant property damage or damage to the environment [1]. Factories are often categorised as safety critical systems since failures of these systems, under certain conditions, can lead to severe consequences [2]. Assuring safety for such systems should provide justified confidence that all potential risks due to system failures are either eliminated or acceptably mitigated. Hence, all failures which might expose the manufacturing processes to hazards shall be analysed and controlled as part of pre-deployment safety assurance and monitored and controlled as part of operational phase.

Developers of some safety critical systems build a safety case to demonstrate the safety aspect of their system by identifying all unreasonable risks and describing, in the light of the available evidence, how these risks have been eliminated or adequately mitigated. Typically, a safety case comprises both safety evidence (e.g. safety analyses, software and hardware inspection reports, or functional test results) and a safety argument (i.e., reasoning) explaining that evidence. The safety argument shows which claims the developer uses each item of evidence to support and how those claims, in turn, support broader claims about system behaviour, hazards addressed, and, ultimately, acceptable safety [3].

An organisation building a safety case should be accountable for the ownership of the risks to be controlled by adopting an appropriate safety management system, performing a hazard assessment, selecting appropriate controls, and implementing them [4]. In order to help building a sufficient and credible (i.e., on a scientific basis) confidence in the safe performance of a system, its safety case shall always communicate the actual safe performance of the system, and shall always contain only acceptable items of evidence that this system meets its safety requirements. However, an item of evidence is valid only in the operational and environmental context in which it is obtained or to which it applies. More clearly, as the system evolves after deployment, there could be a mismatch between our communicated understanding of the system safety by the safety case and the safety performance of the system in actual operation, which might invalidate many of the prior assumptions made, undermine the collected items of evidence and thus defeat safety claims [5]. Despite the improvements in operational safety monitoring, there is insufficient clarity on how to utilise the analysis results of the monitored data on the documented confidence in safety cases.

In safety critical systems, failure rates are sometimes used as quantitative criteria while performing safety assessment (i.e., Probabilistic Safety Assessment (PSA)). Failure Rate (FR=λ) is defined as the probability per unit time that a component experiences a failure at time "t", given that the component was operating at time "0" and has survived to time "t" [6]. Failure rates can be deemed as a reliability prediction that together with the consequences (*Risk = probability of failure * consequence of failure*) determine the Safety Integrity Level (SIL), which in turn specifies a target level of risk reduction that should be considered by a safety function or instrument. The quality of the failure rate measure determines the quality of the PSA. Hardware components are usually provided by generic failure rates which are derived by the statistical analyses of the failure frequency [7]. Failure frequency is usually obtained by the test results and the historical data of the components. Although the calculation of a generic failure rate is based on complex models which include factors using specific component data such as temperature, environment, and stress [6], it is, at its best, just a probability that is still subject to a percentage error even if it is used in the same context as in specifications. Assuming the perfection of the failure rate calculations is not judicious and can be misleading. Hence, a minimum level of fault tolerance in the architectural design of the safety functions should be considered. For example, the functional safety standards IEC 61508 [8] and

IEC 61511 [9] recognise that there is always some degree of uncertainty in the assumptions made in calculation of failure rate and probability [10].

In this paper, we propose a novel technique to detect the discrepancies between the failure rates of system's components during their operational life and their generic failure rates used for analysis and assurance during the design time. Since it is infeasible to monitor the failure rates of all components of a system, the technique utilises probabilistic Fault Tree Analysis (FTA) to evaluate the criticality of the system components, and selects the most critical ones for monitoring. The technique derives safety contracts for the selected components and associate them with the relevant events in the FTA and the relevant parts in the safety case. If a discrepancy is detected between an observed failure rate (λ_O) and a generic failure rate (λ_G) of the same component, where $\lambda_O > \lambda_G$, then the relevant contract should be flagged and the referred parts of both the FTA and the safety case should be revisited.

Our hypothesis is that using safety contracts for monitoring the failure rates during the operational life of a system can help to provide essential feedback on the overall confidence in safety. More clearly, getting more precise measure of failure rates than the predicted ones will 1) improve the efficacy of the system design to reduce the risk (mitigate by design), 2) define stronger evidence (e.g., refine or rectify the test results) and 3) highlight the required preventive, corrective, perfective or adaptive maintenance for safer operation

In this paper, we specifically make the following four contributions:

1. A novel technique to continuously reassess the failure rates and use the results to suggest system changes or maintenance
2. A new way to derive safety contracts to facilitate the traceability between the system design, safety analysis and the safety case
3. An example of how to argue more compelling over the failure rate in the light of the derived evidence from the operational phase
4. An example of how to carry out a through-life safety assurance

The rest of the paper is organised as follows: In Section 2, we present our approach to verify the design assumptions during runtime by safety contracts. In Section 3, we apply our technique to an AGV system to illustrate the main steps. In Section 4, we discuss how the suggested approach enables a through-life safety assurance. Finally, we conclude and describe the future directions in Section 5.

2 Using Safety Contracts to Verify Design Assumptions During Runtime

Failures of components in safety critical systems are typically divided into four modes, namely, Safe Detected (SD), Safe Undetected (SU), Dangerous Detected (DD), and Dangerous Undetected (DU). DD and DU failures can cause loss of a safety function while we believe that we are protected and this might happen in fraction of diagnostic interval in case of DD failures or during the unknown downtime in case of DU failures [11]. DU failures are typically due to either random or systematic failures. In this paper, we specifically focus on dangerous failures (DD and DU). Whenever FTAs are constructed to evaluate hazards, the basic event failure data must describe only failures that contribute to that hazard and thus only dangerous failure rates (λ_D) should be included for the basic events, where $\lambda_D = \lambda_{DD} + \lambda_{DU}$.

In this section, we propose a technique that aims to determine the λ_D of particular HW components in their operational life (observed $\lambda_D = \lambda_{D\_O}$) and compare the results with the design assumptions of these components (generic $\lambda_D = \lambda_{D\_G}$) to ultimately highlight any discrepancies between λ_{D_O} and λ_{D_G}. The technique uses criticality importance measure to rank the components from the most to the less critical so that safety engineers can select particular components for monitoring when it is infeasible to monitor all of them. The technique also uses sensitivity analysis to determine whether a highlighted discrepancy is acceptable or not. The technique heavily depends on probabilistic FTAs, and it comprises 8 steps as follows:

2.1 Determine the PFD or the PFH in the FTA

In this step, we calculate the PFD (Probability of Failure on Demand) or the PFH (Probability of Failure per Hour) using a probabilistic FTA where each component is specified by its λ_{D_G}. The selection between PFD and PFH is based on the demand of a safety function. More clearly, if the safety function will be working in a continuous mode, then we have to select PFH [8]. However, if the safety function is expected to work once per year (at most), then PFD should be selected [8]. To calculate the PFD or PFH of an FTA, four sub-steps should be performed as follows:

A. Calculate the Failure Probability of the Basic Events: There are different formulas used to calculate PFD depending on different factors, such as system's structure (K-out-of-N structures), Common Cause Factor (CCF), operational maintenance, safety standards obligations, etc. For example, Exida (a leading product certification and knowledge company) provides a realistic formula to calculate the PFD [12]. However, the difference between PFD formulas will not be influential in our technique. For the sake of simplicity, we adopt the PFD formula given in [13]. Formula 1 shows how we calculate the PFD for the basic events:

$$PFD(i) = \lambda_{D,i} * \tau \tag{1}$$

where i denotes the basic event and τ is the proof test interval. The component reparation or replacement time is assumed to be short and thus it is negligible.

The main difference between calculating PFD and PFH is in the logic of determining the probability of failures for the basic events. To calculate the PFH for the FTA's events, Formula 1 should be replaced with Formula 2, which is basically the famous unreliability exponential equation where only λ_D is considered. Unreliability in the context of functional safety is interpreted as the probability of a function to fail during a given time interval.

$$PFH(i) = 1 - e^{-\lambda_D t} \tag{2}$$

For calculating the PFD or PFH, we assume the failure rates of all components are constants, independent and have the same τ. We also assume that all potential CCFs are explicitly modelled as basic events in the FTAs. The rest of the sub-steps (B, C and D) are the same irrespective of we use PFD or PFH.

B. Determine Minimal Cut Set (MCS) in the FTA: The MCS is defined as: *"A cut set in a fault tree is a set of basic events whose (simultaneous) occurrence ensures that the top event occurs. A cut set is said to be minimal if the set cannot be reduced without losing its status as a cut set"* [14]. There are several algorithms to find the MC. We apply Mocus cut set algorithm [14].

C. Calculate the Failure Probability of the Determined MCS: Calculating the probability of occurrence for the top event in a FTA with many MCS requires calculating the probability of those MCS. The failure probability of each determined MCS in the previous sub-step should be calculated according to formula 3 [11], as follows:

$$\check{Q}_j(t) = \prod_{i \in C_j} q_i(t) \tag{3}$$

where $q_i(t)$ denotes the probability of basic event i at time t, $\check{Q}_j(t)$ is the probability that minimal cut set j is in failed state at time t, $i \in C_j$ denotes the minimal cut set j that contains the basic event i.

D. Calculate the PFD or PFH of the Top Event: We calculate the actual PFD or PFH by the *upper bound approximation formula* 4 [11] using the determined MCS, as follows:

$$PFD_{Act}(Top), PFH_{Act}(Top) = \sum_{j=1}^{k} \check{Q}_j(t) \tag{4}$$

So far, all PFD or PFH calculations are based on λ_{D_G}. We refer to the result of the probability calculation based on λ_{D_G} as *Actual or Act.* The PFD$_{Act}$(Top) or PFH$_{Act}$(Top) are design assumptions which will be compared with the observed λ to check the correctness/validity of the design assumptions.

2.2 Identify the Most Critical Components

Monitoring every single component in safety critical systems is infeasible especially since such systems become bigger and more sophisticated over time. However, some components in a system are more critical for the system safety than other components. The objective of this step is to identify the most critical components in a system w.r.t the FTA. There are different measures through which FTA's events can be ranked based on their importance (e.g., Birnbaum, Criticality Importance, Fussel-Vesely Importance, Risk Achievement Worth (RAW)). In our technique, however, we are interested to rank the components based on their contributions to system safety. More specifically, we are interested in the components whose failures have the maximum impact on system safety. RAW is a measure that focuses on the 'worth' of the basic event in 'achieving' the present level of risk and indicates the importance of maintaining the current level of reliability for the basic event [14]. RAW is often used as an importance measure to rank components in terms of safety significance [15] and hence we will adopt it for our work .

The failure probability of the component i at time t may be described as:

$$P(i) = \begin{cases} 0 & \text{if the component is functioning at time } t \\ 1 & \text{if the component is in a failed state at time } t \end{cases}$$

The RAW, I^{RAW}(i|t) is the ratio of the (conditional) system unreliability if component i is $P(1)$, and it is calculated as follows [14]:

$$I^{RAW}(i|t) = \frac{1 - h(0_i, p(t))}{1 - h(p(t))} \text{ for } i = 1,\, 2,\, ...,\, n \tag{5}$$

where $h(0_i, p(t))$ is the probability of top event with component $i = P(1)$, and $h(p(t))$ is probability of top event. All basic events should be ranked from the most important to the less important. The most important event is the event for which Formula 5 has the maximum value.

2.3 Refine the Identified Critical Parts

The idea of this step is to discuss with system developers (e.g., safety engineers) and refine the ranked list of the critical components. This step is important, since it embeds the system level knowledge and experience of engineers regarding the uncertainty in a generic λ as well as helps as a validation step in the decision making process. For example, it could be the case that a high ranked critical component in the list has a stable λ_G and systems engineers decide not to monitor it. That is, it is envisaged that some events may be removed from the list or the rank of some of them change. Moreover, the list can be extended to add any additional events by the developers.

2.4 Perform Sensitivity Analysis

The idea of this step is to determine the maximum allowable λ_D (λ_{D_Max}) of the system components which are selected for monitoring. More specifically, we need to define the upper- and lower bounds of the acceptable λ_D of each event in the MCS, where PFD$_{Act}$(Top) or PFH$_{Act}$(Top) is less than or equal to the required probabilities PFD$_{Req}$(Top) or PFH$_{Req}$(Top), respectively. The required probability is described as safety requirements by the safety standards (e.g., SIL, ASIL and DAL). It is important for our technique to determine to which extent PFD$_{Act}$(i) or PFH$_{Act}$(i) can be deviated while PFD$_{Act}$(Top) or PFH$_{Act}$(Top) still satisfies PFD$_{Req}$(Top) or PFH$_{Req}$(Top), respectively. To this end, two main activities should be performed, as follows:

Determine the maximum allowable $q_{i,Max(t)}$ for each component The $q_{i,Max}$(t) for each component should be determined with respect to PFD$_{Req}$(Top) or PFH$_{Req}$(Top). Formula 6 should be used to determine $q_{i,Max}$(t) for each component at a time.

$$\frac{PFD_{Req}(Top), PFH_{Req}(Top) - (\sum \check{Q}_{i \notin C_j}(t))}{\sum \check{Q}_{i \in C_j}(t) \neg q_i(t)} = \frac{\sum \check{Q}_{i \in C_j}(t)}{\sum \check{Q}_{i \in C_j}(t) \neg q_i(t)} \tag{6}$$

where $i \notin C_j$ denotes the minimal cut set j that does not contain basic event i.

Determine λ_{D_Max} for Each Component Once we have $q_{i,Max}$(t) for a component it is easy to determine its $\lambda_{D,Max}$. Formula 7 determines $\lambda_{D,Max}$ in case of PFD, as follows:

$$\lambda_{D,Max} = \frac{q_{i,Max}(t)}{\tau_i} \tag{7}$$

Formula 8 determines $\lambda_{D,Max}$ in case of PFH, as follows:

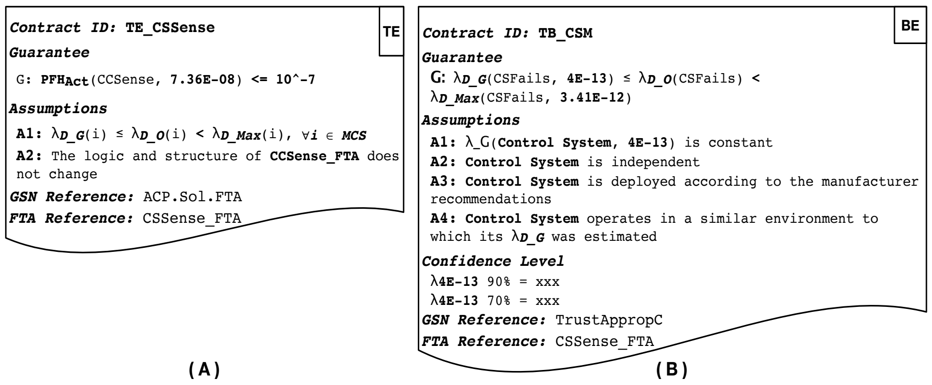
$$\lambda_{D\_Max} = \frac{-\ln(q_{i,Max}(t))}{\tau_i} \tag{8}$$

After calculating $\lambda_{D,Max}$ for all events, the latter should be ranked from the most sensitive to the less sensitive to change. The most sensitive event is the event for which Formula 9 is the minimum:

$$Sensitivity(\lambda_{D_i,G}) = \frac{\lambda_{D_i,Max} - \lambda_{D_i,G}}{\lambda_{D_i,G}} \tag{9}$$

2.5 Derive Safety Contracts

In this step, safety contracts should be derived from FTAs. The main objectives of deriving safety contracts are: 1) highlight the most important components to make them visible up front for developers attention [16], and 2) record the thresholds of $\lambda_D(i)$ to continuously compare them with the monitoring results (λ_{D_O}). Hence, if λ_{D_O} of component i exceeds the guaranteed $\lambda_{D\_Max}(i)$ in the contract of that component, then we can infer that the contract in question is broken and the related FTA should be re-assessed in the light of the λ_{D_O}. Another objective to derive safety contracts is to associate these contracts with safety arguments as reference points so that developers know the related part of



TE

Contract ID: **TE_CSSense**

Guarantee

G: **$\mathbf{PFH}_{\mathbf{Act}}$**(CCSense, **7.36E-08**) **<= 10^-7**

Assumptions

A1: λ_{D_G}(i) ≤ λ_{D_O}(i) **<** λ_{D_Max}(i), ∀***i*** ∈ ***MCS***

A2: The logic and structure of **CCSense_FTA** does not change

GSN Reference: ACP.Sol.FTA

FTA Reference: CSSense_FTA

(A)

BE

Contract ID: **TB_CSM**

Guarantee

G: λ_{D_G}(CSFails, **4E-13**) ≤ λ_{D_O}(CSFails) **<** λ_{D_Max}(CSFails, **3.41E-12**)

Assumptions

A1: λ_G(**Control System, 4E-13**) is constant

A2: Control System is independent

A3: Control System is deployed according to the manufacturer recommendations

A4: Control System operates in a similar environment to which its λ_{D_G} was estimated

Confidence Level

λ**4E-13** 90% = xxx

λ**4E-13** 70% = xxx

GSN Reference: TrustAppropC

FTA Reference: CSSense_FTA

(B)

Fig. 1. A. Contract template: Top Event. **B.** Contract template: Basic Event

the argument when they review a FTA and vice versa. To this end, we introduce two templates to derive contracts. The first contract template is for deriving a contract for the top event only. The *top event safety contract* is annotated with the abbreviation "**TE**" in the upper-right corner of the contract to denote that this contract is derived for a **T**op **E**vent as shown in Figure 1-A.

The second contract template is for deriving a safety contract for each event in the MCS (i.e., events related to important components). This type of contracts is referred to as "*monitoring safety contracts*" and it is is annotated with the abbreviation "**BE**" in the upper-right corner to denote that this contract is derived for a **B**asic **E**vent as shown in Figure 1-B.

2.6 Associate Safety Contracts with Safety Arguments

In this step, all safety contracts which were derived in Step 4 should be associated with safety arguments. This step assumes that the safety argument should come down to a claim that the "probability of failure of hazard H due to component failure is acceptable", in turn supported by a context element about what that probability is in the context of an applicable definition of acceptable, in turn supported by the FTA as evidence. An Assurance Claim Points (ACP) [17] should be created between the claim about the acceptable probability and the evidence, where a separate confidence argument should extend this ACP to argue over the quality of the used failure rates to calculate PFD_{Act}(Top) or PFH_{Act}(Top).

It is necessary that the argument should be clearly structured and the items of evidence to be clearly asserted to support the argument [18]. There are several ways to represent safety arguments (e.g., textual, tabular, graphical, etc.). In this paper, we use the Goal Structuring Notation (GSN) [18], which provides a graphical means of communicating (1) safety argument elements, claims (goals), argument logic (strategies), assumptions, context, evidence (solutions), and (2) the relationships between these elements. The basic notations of GSN are shown in Figure 2 (in the upper left side corner). A goal structure shows how goals are successively broken down into ('solved by') sub-goals until eventually supported by direct reference to evidence. GSN can clarify the argument strategies adopted (i.e., how the premises imply the conclusion), the rationale for the approach (assumptions, justifications) and the context in which goals are stated.

Assertions in a safety argument relate to the sufficiency and appropriateness of the inferences declared in the argument, the context and assumptions used and the evidence cited [17]. For example, when an item of evidence is used to support a claim, it is asserted that this evidence is sufficient to support the claim. However, a simple 'SolvedBy' relation between the evidence and the claim will not satisfy a reviewer's concerns to reach a certain level of confidence, such as, 'why the reviewer should believe that the evidence is appropriate for the claim?' or 'whether it is trustworthy'.

Hawkins et al., [17] introduced "An assured safety argument" as a new structure for arguing safety in which the safety argument is accompanied by a confidence argument that documents the confidence in the structure and bases of the safety argument. Hawkins suggests that instead of decomposing the arguments further to argue over the appropriateness and trustworthiness of the supporting evidence, an ACP can be created to indicate an assertion in the safety argument. An ACP is indicated in GSN with a named black rectangle on the relevant link and a confidence argument should be developed for each ACP [17]. Three types of assertions were defined as ACPs as follow:

1. Asserted inference: the ACP for an asserted inference is the link between the parent claim and its strategy or sub-claims

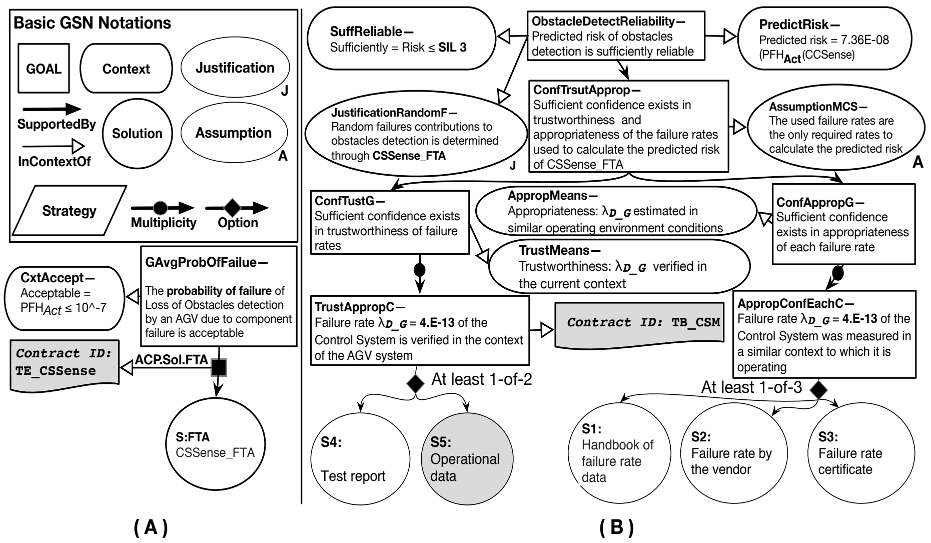

Basic GSN Notations
GOAL
Context
Justification
J
SupportedBy
InContextOf
Solution
Assumption
A
Strategy
Multiplicity
Option
CxtAccept— Acceptable = PFHAct ≤ 10^-7
GAvgProbOfFailue— The probability of failure of Loss of Obstacles detection by an AGV due to component failure is acceptable
Contract ID: TE_CSSense
ACP.Sol.FTA
S:FTA CSSense_FTA
SuffReliable— Sufficiently = Risk ≤ SIL 3
ObstacleDetectReliability— Predicted risk of obstacles detection is sufficiently reliable
PredictRisk— Predicted risk = 7.36E-08 (PFHAct(CCSense)
JustificationRandomF— Random failures contributions to obstacles detection is determined through CSSense_FTA
J
ConfTrsutApprop— Sufficient confidence exists in trustworthiness and appropriateness of the failure rates used to calculate the predicted risk of CSSense_FTA
AssumptionMCS— The used failure rates are the only required rates to calculate the predicted risk
A
ConfTustG— Sufficient confidence exists in trustworthiness of failure rates
AppropMeans— Appropriateness: λD_G estimated in similar operating environment conditions
TrustMeans— Trustworthiness: λD_G verified in the current context
ConfAppropG— Sufficient confidence exists in appropriateness of each failure rate
TrustAppropC— Failure rate λD_G = 4.E-13 of the Control System is verified in the context of the AGV system
Contract ID: TB_CSM
AppropConfEachC— Failure rate λD_G = 4.E-13 of the Control System was measured in a similar context to which it is operating
At least 1-of-2
At least 1-of-3
S4: Test report
S5: Operational data
S1: Handbook of failure rate data
S2: Failure rate by the vendor
S3: Failure rate certificate
(A)
(B)

Fig. 2. **A.** A probability of failure argument with an association of a top event safety contract. **B.** Confidence argument with an association of a monitoring safety contract

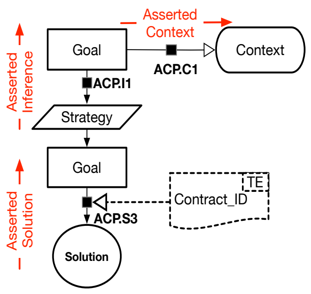


Fig. 3. Types of ACPs with an example of each usage [17]

2. Asserted context: the ACP for asserted context is the link to the contextual element
3. Asserted solution: the ACP for asserted solutions is the link to the solution element

In this step, we suggest to use the principle of the ACP. Hence, the *top event safety contract* should be associated with the ACP (i.e., asserted solution) between the GSN goal which claims the acceptability of the hazard probability due to a component failure and the GSN solution which refers to the relevant FTA. Whereas, each *monitoring safety contract* should be associated with a GSN goal about the relevant component in the confidence argument. Figure 2-A shows a pattern of PFD or PFH argument and an example of top event safety contract association. Figure 2-B shows a confidence argument pattern with an association of a monitoring safety contract. Figure 3 instantiates an example of each ACP type and it also represents our suggested traceability means which associates the derived contracts from FTAs with safety arguments (the dotted part in the figure).

2.7 Determine λ_{D_O} Using the Data from Operation and Compare it to the Guaranteed λ_{D_Max} in Safety Contracts

In this step, λ_{D_O} of specified components should be obtained during the components' runtime. Using runtime monitors is one way to obtain data from operation. There are many proposed architectures to detect or test a system (or parts of it) for bad behaviour [19]. We provide a monitoring logic which requires two parameters (inputs) from any monitoring framework, namely, the number of recorded failures (i.e., DD and DU) as well as τ in time unit (e.g., hours). Algorithm 1 should be used to determine λ_{D_O} using the data from operation and compare it to the guaranteed λ_{D_Max}. The more we monitor a component and record its failures the more confident we will be in its actual λ_D in a specific context. The calculated level of confidence can reveal how long we still need to monitor a component to reach a certain level of confidence. Hence, our algorithm also calculates the confidence level of λ_{D_O}(i)70% and λ_{D_O}(i)90% continuously and cumulatively using the *Chi-Squared distribution*. The calculated levels of confidence of a monitored component are automatically inserted into its "*monitoring safety contract*" and get updated continuously so that developers and assessors can review them in the FTA and the safety argument.

2.8 Update the Safety Contracts and Re-visit the Safety Argument

If a *monitoring safety contract* is broken it means that there is at least one broken *top event safety contract* as well. In this case, the broken safety contracts should be used to trace the FTA events and elements of safety arguments (for which

Algorithm 1: The monitoring logic to determine λ_{D_O} and compare it to λ_{D_Max}

```
   Data: MissionTime, $\tau$, $\lambda_{D\_Max}$, $\lambda_{DU\_O}$, DUfailures = 0, $\lambda_{DD\_O}$, DDfailures =
         0, $\lambda_{D\_O}$, Num_Comp, CL90, CL70;
   Result: Determine $\lambda_{D\_O}$ and compare it to $\lambda_{D\_Max}$
 1 TotMonTime = clock();          \\Comment: start monitoring the mission time
 2 while TotMonTime $\leq$ MissionTime do
 3     Test_Interval_Monitor = clock();  \\Comment: start the monitoring time of
       the test interval time
 4     while Test_Interval_Monitor $\leq \tau$ do
 5         if a DD failure is found then
 6             DDfailures++;          \\Comment: add an observed failure from a
               diagnosis log file
 7         end
 8         if a DU failure is recorded then
 9             DUfailures++;       \\Comment: add an observed failure which was
               inserted manually
10         end
11         $\lambda_{DU\_O}$ = 1/((TotMonTime * Num_Comp) / DUfailures);  \\Comment:
           calculate $\lambda_{DU\_O}$
12         $\lambda_{DD\_O}$ = 1/((TotMonTime * Num_Comp) / DDfailures);  \\Comment:
           calculate $\lambda_{DD\_O}$
13         $\lambda_{D\_O} = \lambda_{DU\_O} + \lambda_{DD\_O}$;                \\Comment: calculate $\lambda_{D\_O}$
14         CL70 = Chi-
           Squared($X^2_{70\%,2(DUfailures+DUfailures+1)}$)/(2*Num_Comp*TotMonTime);
           \\Comment: $\lambda_{D\_O}$70%
15         CL90 = Chi-
           Squared($X^2_{90\%,2(DUfailures+DUfailures+1)}$)/(2*Num_Comp*TotMonTime);
           \\Comment: $\lambda_{D\_O}$90%
16         if $\lambda_{D\_O} \geq \lambda_{D\_Max}$ then
17             Contract [**C**] is broken;  \\Comment: highlight the broken contract
               whenever $\lambda_{D\_O} \geq \lambda_{D\_Max}$
18         end
19     end
20     Test_Interval_Monitor = 0; \\Comment: reset the $\tau$ timer to start a new one
21 end
```

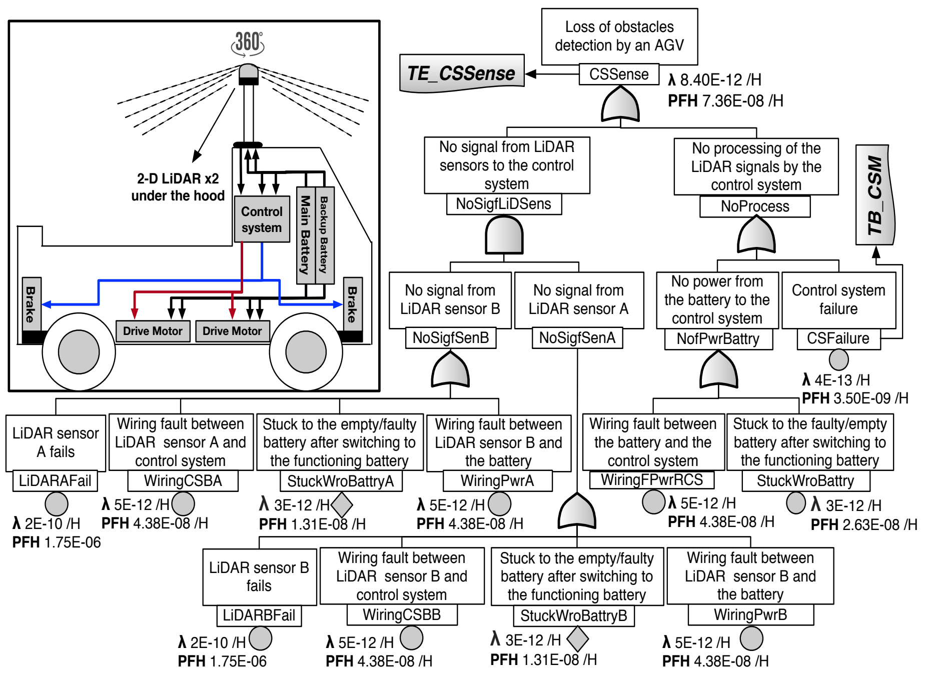

360°
2-D LiDAR x2 under the hood
Control system
Main Battery
Backup Battery
Brake
Brake
Drive Motor
Drive Motor
Loss of obstacles detection by an AGV
TE_CSSense
CSSense
λ 8.40E-12 /H
PFH 7.36E-08 /H
No signal from LiDAR sensors to the control system
NoSigfLiDSens
No processing of the LiDAR signals by the control system
NoProcess
TB_CSM
No signal from LiDAR sensor B
NoSigfSenB
No signal from LiDAR sensor A
NoSigfSenA
No power from the battery to the control system
NofPwrBattry
Control system failure
CSFailure
λ 4E-13 /H
PFH 3.50E-09 /H
LiDAR sensor A fails
LiDARAFail
λ 2E-10 /H
PFH 1.75E-06
Wiring fault between LiDAR sensor A and control system
WiringCSBA
λ 5E-12 /H
PFH 4.38E-08 /H
Stuck to the empty/faulty battery after switching to the functioning battery
StuckWroBattryA
λ 3E-12 /H
PFH 1.31E-08 /H
Wiring fault between LiDAR sensor B and the battery
WiringPwrA
λ 5E-12 /H
PFH 4.38E-08 /H
Wiring fault between the battery and the control system
WiringFPwrRCS
λ 5E-12 /H
PFH 4.38E-08 /H
Stuck to the faulty/empty battery after switching to the functioning battery
StuckWroBattry
λ 3E-12 /H
PFH 2.63E-08 /H
LiDAR sensor B fails
LiDARBFail
λ 2E-10 /H
PFH 1.75E-06
Wiring fault between LiDAR sensor B and control system
WiringCSBB
λ 5E-12 /H
PFH 4.38E-08 /H
Stuck to the empty/faulty battery after switching to the functioning battery
StuckWroBattryB
λ 3E-12 /H
PFH 1.31E-08 /H
Wiring fault between LiDAR sensor B and the battery
WiringPwrB
λ 5E-12 /H
PFH 4.38E-08 /H

Fig. 4. An overview of AGV's and its probabilistic FTA (CSSense_FTA)

the contracts were derived). As a result of doing this, developers can specify the entry point of the impact of failure in the safety analysis and the safety argument. It is worth mentioning that we assume the existence of a redundant component of the failing component. Hence, a broken safety contract does not necessarily lead to a total system failure.

3 Motivating Example: Automated Guided Vehicles (AGVs)

AGVs are being extensively used for more than 40 years now. They are used for intelligent transportation and distribution of materials in warehouses and auto-production lines. There are different setups and operational assumptions for each application of AGVs in industry. In our example, however, the AGVs are a number of battery-powered vehicles whose movements are autonomous. The AGVs are interfaced to automated warehouse and holding area, and to the machine tools, so that stock movement requirements can be fulfilled. The plant, in our example, is not fully automated so that people cannot be fully excluded from the areas where the AGVs work. Clearly, one of the most important safety features of the AGV vehicles is their ability to detect obstacles and stop quickly in order to avoid a collision with humans, hazardous objects (e.g., flammable

Table 1. A summary of the results of applying the steps 1-5

			STEP 1	STEP 2	STEP 3			STEP 4	STEP 5
No.	**Events**	$\lambda_{D,G}$	**PFH**	**RAW**	**Max PFH**	λ_{D_Max}	**Sensitivity**	**Refine**	**Contract**
1	CSSense (Top)	8.4E-12	7.36E-08		10^{-7}				TE_CSSense
2	CSFails	4E-13	3.50E-09	13589269.0946	2.99E-08	3.41E-12	7.5380	1	TB_CSM
3	WiringFPwrRCS	5E-12	4.38E-08	13589268.5470	7.02E-08	8.02E-12	0.6030		
4	StuckWroBattry	3E-12	2.63E-08	13589268.7851	5.27E-08	6.02E-12	1.0051	3	
5	LiDARAFail	2E-10	1.75E-06	26.3559	1.42E-02	1.63E-06	8137.5	2	
6	WiringCSBA	5E-12	4.38E-08	26.3559	1.42E-02	1.63E-06	325499		
7	StuckWroBattryA	3E-12	2.63E-08	26.3559	1.42E-02	1.63E-06	542499	3	
8	WiringPwrA	5E-12	4.38E-08	26.3559	1.42E-02	1.63E-06	325499		
9	LiDARBFail	2E-10	1.75E-06	26.3559	1.42E-02	1.63E-06	8137.5	2	
10	WiringCSBB	5E-12	4.38E-08	26.3559	1.42E-02	1.63E-06	325499		
11	StuckWroBattryB	3E-12	2.63E-08	26.3559	1.42E-02	1.63E-06	542499	3	
12	WiringPwrB	5E-12	4.38E-08	26.3559	1.42E-02	1.63E-06	325499		

materials, electrical resources, other AGVs, etc.). After performing safety analysis, a number of safety hazards were identified. In this paper, we will focus on one hazard, which is: *Loss of obstacle detection while the vehicle is in motion.* A redundant 2-D LiDAR sensor with all-round (360°) visibility is used for detecting obstacles within up to 30 meters range. Information about detected obstacles are sent to the control system to determine the manoeuvring strategy to ultimately avoid any potential collision.

According to the likelihood of occurrence, potential consequences and other safety countermeasures in the AGVs, the obstacle detection function is assigned **SIL 3** (Safety Integrity Level) according to IEC 61508. Moreover, since the function under discussion operates in a high demand (i.e., in a continuous mode), the allowable frequency of dangerous failure according to the same standard is PFH $< 10^{-7}$. The proof test interval τ is assumed as 1 year (i.e., 8760 hours) for all components. Figure 4 shows an overview of the AGV design (on upper left-hand corner). The figure also shows the FTA of the system where the top event together with the basic events are specified by λ_{D_G}.

Applying the first 5 steps in Section 2 is straightforward. Table 1 provides the results of the steps 1-5. The *Refine* column reflects the experts judgment that is supported by the RAW and Sensitivity ranking. For the sake of giving a clear example of what should be done next, we assume that *Control system* got the highest priority for monitoring (the grey row in Table 1). Hence, two contracts should be derived in the case: 1) TE contract *TB_CSM* and, 2) BE contract (i.e., monitoring contract) *TE_CSSense*.

Step 6 requires associating the derived contracts with the safety argument. For AGV system example, we use our suggested GSN patterns in Section 2.6 to create the confidence argument first and then associate the contracts with it through an ACP. Figure 2 presents our safety argument and the role of the proposed monitoring technique to provide supportive evidence for the articulated claims about the failure rates in the argument. Figure 1 shows the derived TE and BE for the top event *CSSense* and the basic event *CSFails*. The figure also shows the GSN and FTA references which reveal the associations (or traceability) of the contracts with the safety argument and the FTA, respectively.

4 A Through-life Safety Assurance Technique

Denney et al., [5] introduced the term "Dynamic Safety Cases (DSCs)" as a novel operationalisation of the concept of through-life safety assurance. The main motivation for introducing DSCs is that the appreciable degree of certainty about the expected runtime behaviour of a system might not be precise or it perhaps over- or underestimate the actual behaviour, which can create deficiencies in the reasoning about the safety performance of that system. Hence, there is a need for a new class of safety assurance techniques that exploit the runtime related data (operational data) to continuously assess and evolve the safety reasoning to, ultimately, provide through-life safety assurance [5]. The suggested lifecycle of DSCs comprises four main activities as follows [5]:

1. **Identify** the sources of uncertainty in a safety case.
2. **Monitor** the runtime operation of the related system to collect data about system and environment variables, events, and assurance deficits in the safety argument(s).
3. **Analyse** the collected operational data from the former activity to examine whether the defined thresholds are met, and to update the confidence in the associated claims
4. **Respond** to operational events that affect safety assurance. Deciding on the appropriate response depends on a combination of factors including the impact of confidence in new data, the available response options already planned, the level of automation provided, and the urgency with which certain stakeholders have to be alerted.

In this section, we explain how using the described technique in Section 2 enables a through-life safety assurance, where we 1) identify a source of uncertainty, 2) provide a runtime monitoring mechanism, 3) analyse the collected operational data, and 4) suggest a response to the operational events.

1. Identify a source of uncertainty: Evidence supporting a claim about a prediction of a hardware failure rate may be obtained from different sources. Handbooks produced by commercial, military or government sources can support a claimed prediction of a hardware failure rate. A hardware vendor or an expert might also support such claims. The explicit logic of a claim about a failure rate prediction and its supported evidence is that the predicted likelihood of component C to fail during time T of operation is λ because a handbook, a vendor or an expert "says so". The implicit assumption of such claims is that the actual λ will conform to the predicted λ during the operational life. This assumption is an obvious source of *uncertainty* (i.e., lack of confidence) which can influence the level of confidence in the safety argument. Hence, it is particularly important to know whether or not the actual failure rate of a component during the operational life will be similar to the predicted (i.e., generic) rate as the evidence suggests.
2. Monitor the actual failure rate: Algorithm 1 provides the runtime monitoring logic through which the number of failures of a hardware component is continuously calculated during runtime.

3. Analyse the collected operational data: Algorithm 1 also analyses the calculated number of failures by comparing it with a predefined threshold.
4. Respond to operational events: If an observed λ exceeds the generic λ and it is not tolerated by the maximum allowed λ, then a safety contract is broken. The monitoring algorithm highlights broken contracts indicating that an additional safety countermeasure should be considered, such as replacing a hardware component with an ultra reliable component or add a redundant component. Since the contracts under monitoring by the algorithm is associated with ACPs in the safety argument, a broken contract indicates the affected GSN elements in the argument.

5 Discussion and Conclusion

Numerous studies and data analysis have shown either a decreasing or increasing failure rate with time. Runtime monitoring enables a new source of data which improves our perception of some functions, components, and behaviours within safety critical systems. Monitoring a property of interest of a system component and analysing the collected data enable us to know more about this component (e.g., the way it behaves, fails, etc.). As a result, we can improve our confidence in safety based upon more conscious reasoning that replaces the intuitive evidence by more cognitive one. Some safety standards require monitoring and re-assessing the reliability parameters which were used during the design time. For example, IEC 61511-1 [9] requires operators to monitor and assess whether reliability parameters of the Safety Instrumented Systems (SIS) are in accordance with those assumed during the design time [10]. Although runtime monitoring is not a new technique, there is no single way to specify what to monitor, why and how. Safety contracts, on the other hand, are useful for building, reusing or maintaining safety critical systems. The cost of maintaining system components can be drastically reduced by using contracts as system developers may rework the components with knowledge of the constraints placed upon them [20].

In this paper, we proposed a novel technique to monitor the runtime of a system and detect the divergence between the failure rates (which were used in the safety analyses) and the observed failure rates in the operational life. The technique enables through-life safety assurance by utilising safety contracts to provide prescriptive data for what should be monitored, and what parts of the safety argument should be revisited to maintain system safety when a divergence is detected. Future work will focus on creating a more in-depth case study to validate both the feasibility and efficacy of the technique for software and hardware applications. We also plan to formally define safety contracts and to fully automate the application of the technique.

Acknowledgment

This work has been partially supported by the Swedish Foundation for Strategic Research (SSF) (through SYNOPSIS and FiC Projects) and the EU-ECSEL (through SafeCOP project).

References

1. J.C. Knight. Safety critical systems: Challenges and directions. In *Proceedings of the 24rd International Conference on Software Engineering (ICSE).*, pages 547–550, May 2002.
2. Omar Jaradat, Irfan Sljivo, Ibrahim Habli, and Richard Hawkins. Challenges of safety assurance for industry 4.0. In *European Dependable Computing Conference (EDCC).* IEEE Computer Society, September 2017.
3. O. Jaradat, P. Graydon and I. Bate. An approach to maintaining safety case evidence after a system change. In *Proceedings of the 10th European Dependable Computing Conference (EDCC)*, UK, 2014.
4. Patrick J. Graydon and C. Michael Holloway. An investigation of proposed techniques for quantifying confidence in assurance arguments. *Safety Science*, 92(Supplement C):53 – 65, 2017.
5. E. Denney, G. Pai, and I. Habli. Dynamic safety cases for through-life safety assurance. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 2, pages 587–590, May 2015.
6. Reliability prediction basics. Technical report, ITEM Software, Inc., 2007.
7. Paolo Pittiglio, Paolo Bragatto, and Corrado Delle Site. Updated failure rates and risk management in process industries. *Energy Procedia*, 45(Supplement C):1364 – 1371, 2014. ATI 2013 - 68th Conference of the Italian Thermal Machines Engineering Association.
8. *Functional safety of electrical/electronic/programmable electronic safety-related systems.* IEC 61508-4:2010.
9. *Functional safety – Safety instrumented systems for the process industry sector.* IEC 61511-1:2016.
10. M. Generowicz and A. Hertel. Reassessing failure rates. Technical report, I&E Systems Pty Ltd, 2017.
11. Marvin Rausand. *Reliability of safety-critical systems: theory and applications.* John Wiley & Sons, 2014.
12. Iwan van Beurden and William M. Goble. The Key Variables Needed for PFDavg Calculation. White paper, Exida, Sellersville, PA 18960, USA, July 2015.
13. William M. Goble. *Control System Safety Evaluation and Reliability.* 2nd edition, 1998.
14. Marvin Rausand and Arnljot Høyland. *System Reliability Theory: Models and Statistical Methods and Applications.* John Wiley & Sons, Inc., 2004.
15. M van der Borst and H Schoonakker. An overview of PSA importance measures. *Reliability Engineering and System Safety*, 72(3):241 – 245, 2001.
16. O. Jaradat, I. Bate, and S. Punnekkat. Using sensitivity analysis to facilitate the maintenance of safety cases. In *Proceedings of the 20th International Conference on Reliable Software Technologies (Ada-Europe)*, pages 162–176, June 2015.
17. Richard Hawkins, Tim Kelly, John Knight, and Patrick Graydon. *A New Approach to creating Clear Safety Arguments*, pages 3–23. Springer London, London, 2011.
18. GSN Community Standard Version 1. Technical report, Origin Consulting (York) Limited, November 2011.
19. Aaron Kane. *Runtime Monitoring for Safety-Critical Embedded Systems.* PhD thesis, Carnegie Mellon University, September 2015.
20. S. Bates, I. Bate, R. Hawkins, T. Kelly, J. McDermid, and R. Fletcher. Safety case architectures to complement a contract-based approach to designing safe systems. In *Proceedings of the 21st International System Safety Conference (ISSC)*, 2003.