MÄLARDALEN UNIVERSITY
SWEDEN

Using Safety Contracts to Verify Design Assumptions During Runtime

Omar T. Jaradat and Sasikumar Punnekkat

Mälardalen University

(Västerås, Sweden)

{omar.jaradat, sasikumar.punnekkat}@mdh.se

23rd International Conference on Reliable Software Technologies (Ada-Europe)

Lisbon, Portugal

June 19, 2018

MÄLARDALEN UNIVERSITY
SWEDEN

Outline

- Background and Motivation
- Research Question
- Research Contributions
- The proposed technique
- An Illustrative use case
- Conclusion & Future Work

Safety Critical Systems

- Safety critical systems are those systems whose failure could result in loss of life, significant property damage, or damage to the environment

- Safety Critical System — Example:

A possible hazard:
Loss of all wheel braking during landing or RTO

Safety Certification

- Many safety critical systems are subject to compulsory or advisory certification process

- Certification processes necessitate building the systems in compliance with domain-specific safety standards

- Safety standards form the basis for the approval and certification of those systems (e.g., ISO 26262, IEC61508 and ARP 4761).

- Safety certification is costly since safety standards require a lot of V&V activities during the development and maintenance of hardware and software parts of safety critical systems

- The V&V of safety critical embedded systems may consume up to 70% of the total development cost

Safety Cases

Developers of safety critical systems might/should make a case about the safety of their systems in which they:

1. manage the safety of a system,
2. explain how the risks have been acceptably managed,
3. make a reasoned argument that a system is (or will be) safe,
4. address and reduce legal liability,
5. Etc.

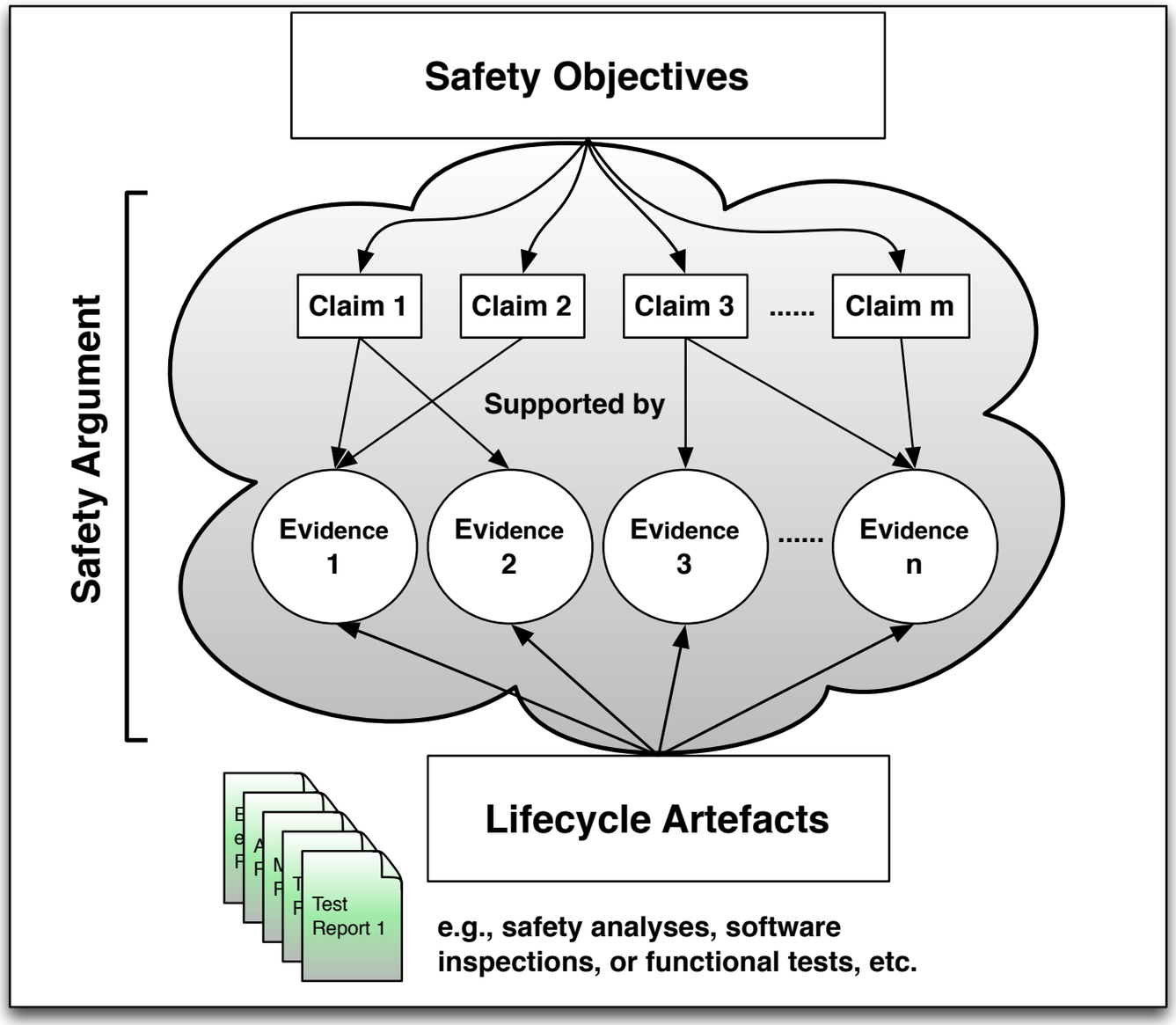
Safety Cases: Definition

- There are different motivations/ purposes to build safety cases. Hence, there are different definitions of the term 'Safety Case'.

- The most common definition is:

  "*A structured **argument**, supported by **evidence**, intended to justify that a system is **acceptably safe** for a specific application in a specific operating environment*"

  [The UK Defence Standard 00-56]



Safety Argument Presentation

A safety argument might be:

- Written in prose
- Presented in tabular notations
- Presented by graphical notations, such as:
  - ✦ **The Goal Structuring Notation (GSN)**
  - ✦ The Claims Argument Evidence (CAE) notation

Safety Cases and Certification

- Safety critical systems are expected to operate for a long period of time and they are frequently subject to preventive, perfective, corrective or adaptive changes during their lifecycle
- Any change that might compromise system safety involves repeating the certification process (i.e., re-certification) and thus, ultimately, necessitates maintaining the corresponding safety case

A safety case is built as a living document that should always be maintained to justify the safety status of the associated system and evolves as this system evolves

Safety Contracts

- Contracts have been exploited as a means for helping to manage system changes in a system domain or in its corresponding safety case

Guarantee: The WCET of task X is $\leq$ 10 milliseconds
Assumptions:
X is:

1. compiled using compiler $[C]$,
2. executed on microcontroller $[M]$ at 1000 MHz with caches disabled, and
3. not interrupted

- The concept of contract is familiar in software development and it was first introduced to constrain the interactions that occur between objects

Research Question

- Having a sufficient confidence in safety evidence is essential to avoid any unanticipated surprise during operational phase

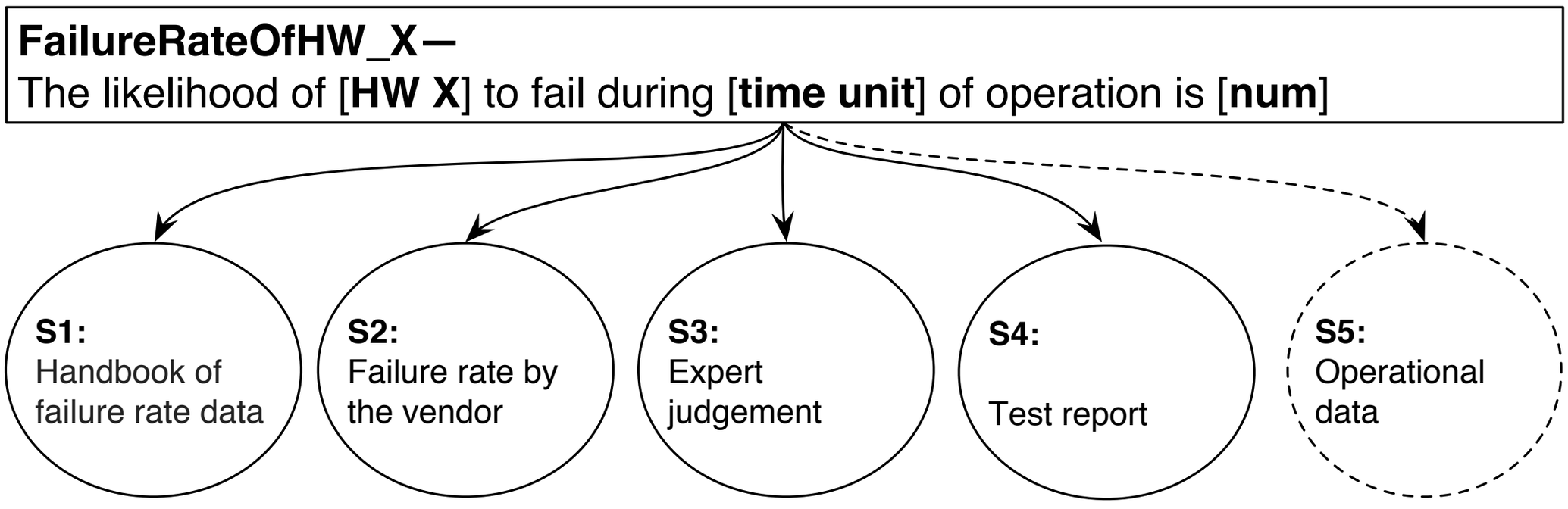
- Sometimes it is impractical to wait for high quality evidence during the development phase, where developers have no choice but to rely on evidence with some uncertainty

- There is a room for mismatches between our communicated understanding of the system safety by the safety case and the safety performance of the system in actual operation

  Safety cases shall always contain only *trustworthy* and *appropriate* items of evidence to support the safety claims. But how to gain and manage a sufficient confidence?

Using Safety Contracts to Verify Design Assumptions During Runtime

- The technique comprises 8 steps and it:
  - 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
  - determines λ_D of particular HW components in their operational life



Step 1: Determine the PFD/PFH in the FTA

1. Calculate the Failure Probability of the Basic Events

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

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

2. Determine Minimal Cut Set (MCS) in the FTA

- We apply Mocus cut set algorithm

3. Calculate the Failure Probability of the Determined MCS

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

4. Calculate the PFD or PFH of the Top Event

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

Step 2: Identify the Most Critical Components

- Monitoring every single component in safety critical systems is infeasible

- 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

- We use Risk Achievement Worth (RAW) because we are interested in the components whose failures have the maximum impact on system safety

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

Step 3: Refine the Identified Critical Parts

- This step is important, since it:
- embeds the system level knowledge and experience of engineers regarding the uncertainty in a generic λ
  - 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
- helps as a validation step in the decision making process

Step 4: Perform Sensitivity Analysis

- The idea is to determine the maximum allowable λ_D (λ_{D_Max}) of the system components which are selected for monitoring

- The upper- and lower bounds of the acceptable λ_D of each event in the MCS should be determined

- 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 $PFH_{Req}(Top)$

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

Step 5 & 6: Derive Safety Contracts and associate them with safety arguments

- The main objectives of deriving safety contracts are:

  1. highlight the most important components to make them visible up front for developers attention
  2. record the thresholds of $\lambda_D(i)$ to continuously compare them with the monitoring results (λ_{D_O})

- **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}

- Contracts should be associated with safety arguments as reference points so that developers know the related part of the argument when they review an FTA and vice versa

Step 7: Determine λ_{D_O} Using the Data from Operation and Compare it to the Guaranteed λ_{D_Max} in Safety Contracts

1. Start monitoring under the mission time
2. Start monitoring under the test interval time
3. Add an observed failure from a diagnosis log file
4. Add an observed failure which is no detected automatically
5. Calculate the failure rate and the level of confidence continuously based on the number of failures and counted time
6. Compare the determined FR with the guaranteed FR in the safety contract and highlight the contract if it is “broken”

Data: MissionTime, τ, λ_{D_Max}, λ_{DU_O}, DUfailures = 0, λ_{DD_O}, DDfailures = 0, λ_{D_O}, Num_Comp, CL90, CL70;
Result: Determine λ_{D_O} and compare it to λ_{D_Max}

```
1  TotMonTime = clock();          \\Comment: start monitoring the mission time
2  while TotMonTime ≤ MissionTime do
3      Test_Interval_Monitor = clock();  \\Comment: start the monitoring time of
       the test interval time
4      while Test_Interval_Monitor ≤ τ 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         λ_DU_O = 1/((TotMonTime * Num_Comp) / DUfailures);  \\Comment:
           calculate λ_DU_O
12         λ_DD_O = 1/((TotMonTime * Num_Comp) / DDfailures);  \\Comment:
           calculate λ_DD_O
13         λ_D_O = λ_DU_O + λ_DD_O;                  \\Comment: calculate λ_D_O
14         CL70 = Chi-
           Squared(X^2_{70%,2(DUfailures+DUfailures+1)})/(2*Num_Comp*TotMonTime);
           \\Comment: λ_D_O 70%
15         CL90 = Chi-
           Squared(X^2_{90%,2(DUfailures+DUfailures+1)})/(2*Num_Comp*TotMonTime);
           \\Comment: λ_D_O 90%
16         if λ_D_O ≥ λ_D_Max then
17             Contract [C] is broken;  \\Comment: highlight the broken contract
               whenever λ_D_O ≥ λ_D_Max
18         end
19     end
20     Test_Interval_Monitor = 0; \\Comment: reset the τ timer to start a new one
21 end
```

Step 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

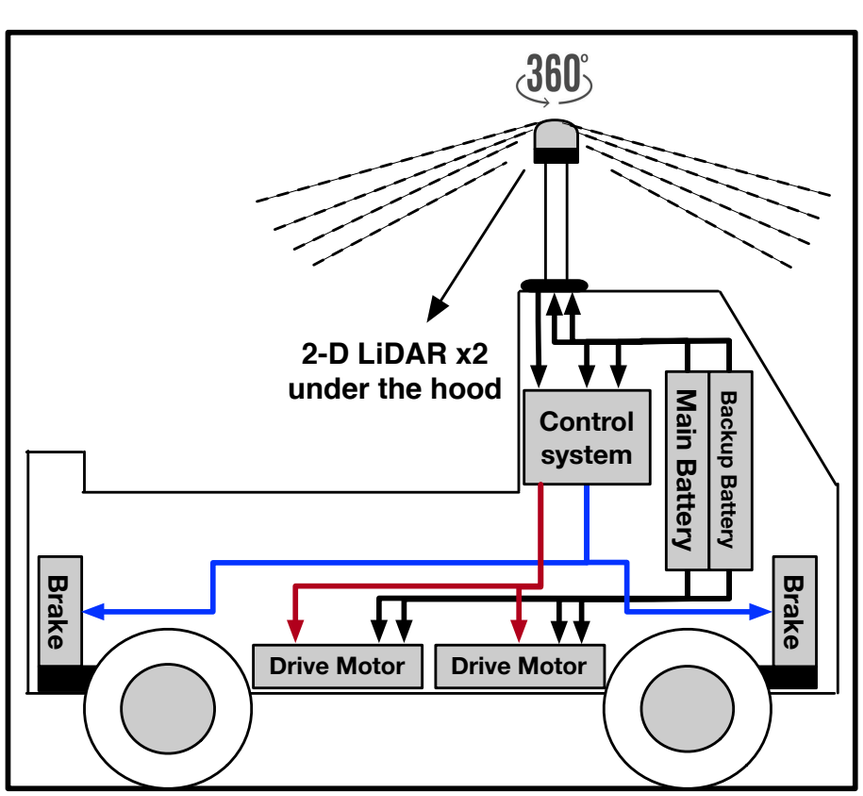
- The broken safety contracts should be used to trace the affected FTA events and safety arguments elements

- The contracts should be updated by the latest λ_{D_O} even when λ_{D_O} is still $\leq \lambda_{D\_Max}$

Automated Guided Vehicles (AGVs): An example

- AGVs are autonomous and battery-powered vehicles
- AGVs are used for intelligent transportation and distribution of materials in warehouses and auto-production lines
- 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
- ***Loss of obstacle detection while the vehicle is in motion***, is a potential hazard
- The obstacle detection function is assigned SIL 3 (Safety Integrity Level) according to IEC 61508
- The maximum allowed frequency of dangerous failure is < $\mathbf{10^{-7}}$ and he proof test interval τ is assumed as 1 year (i.e., 8760 hours) for all components

AGVs: Fault Tree Analysis



Results of 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		

Derived Contracts

TE

Contract ID: **TE_CSSense**

Guarantee

G: **PFH_{Act}**(CCSense, **7.36E-08**) **<= 10^-7**

Assumptions

A1: $\lambda_{D\_G}(i) \leq \lambda_{D\_O}(i) < \lambda_{D\_Max}(i), \forall i \in MCS$

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

GSN Reference: ACP.Sol.FTA

FTA Reference: CSSense_FTA

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

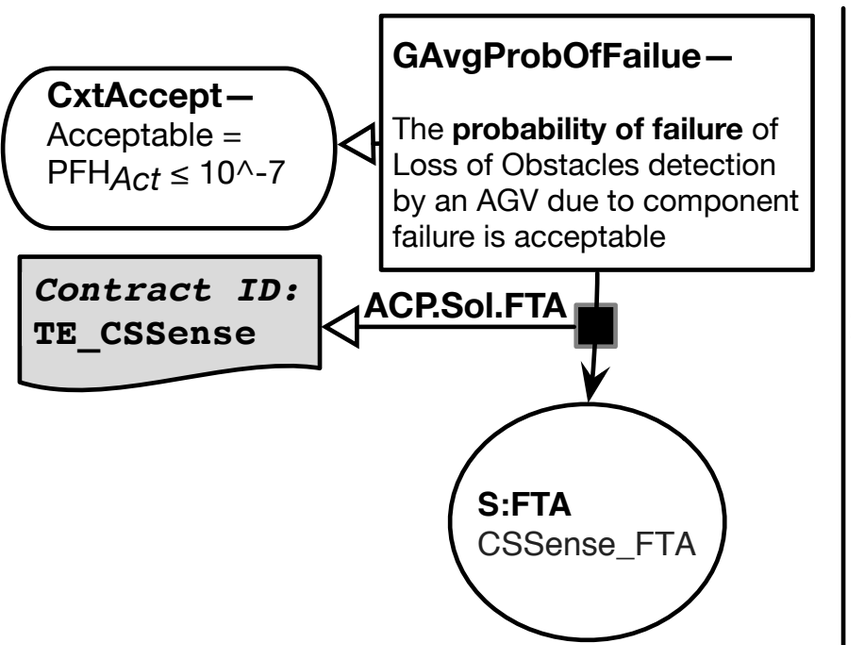
λ**4E-13** 70% = xxx

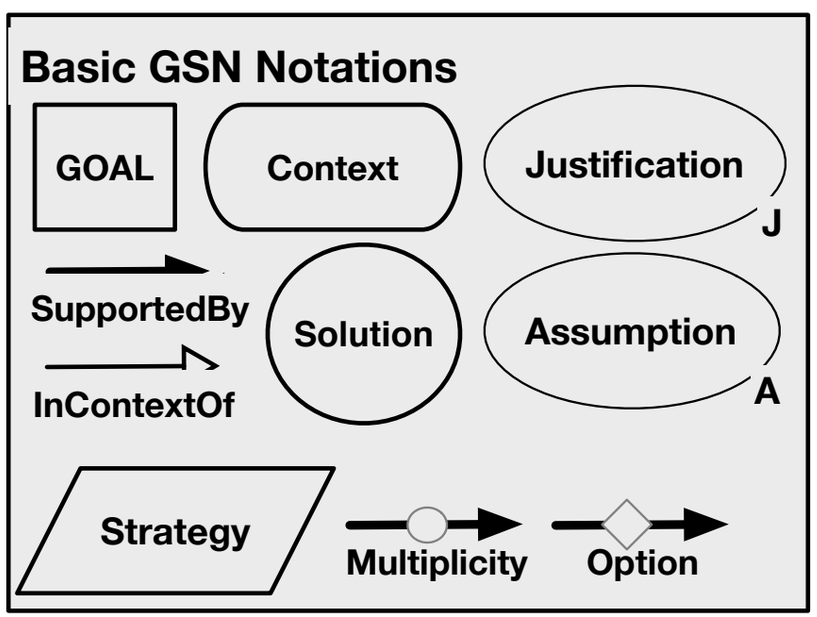
GSN Reference: TrustAppropC

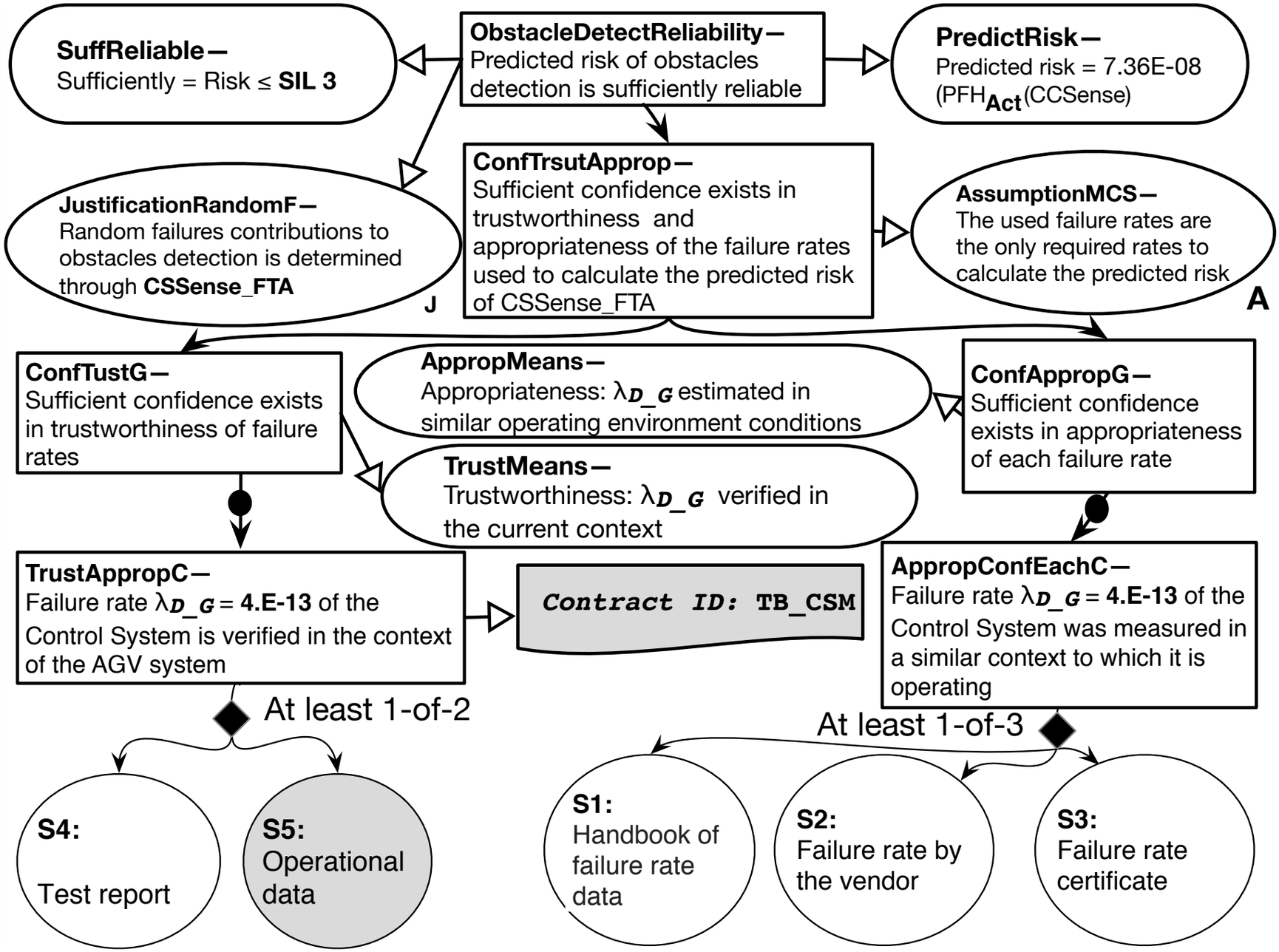
FTA Reference: CSSense_FTA

Illustrative use case

Safety Argument and Associated Contacts







Conclusions

- We proposed:

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

Future Work

- 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

Thank You!

Questions?