Proceedings of the 53rd Hawaii International Conference on System Sciences | 2020

# Measuring Confidence of Assurance Cases in Safety-Critical Domains

Chung-Ling Lin
Western Michigan University
chung-ling.lin@wmich.edu

Wuwei Shen
Western Michigan University
wshen@wmich.edu

Betty H.C. Cheng
Michigan State University
chengb@cse.msu.edu

## Abstract

*Evaluation of assurance cases in the Goal Structuring Notation typically requires certifiers' domain knowledge and experience, and, as such, most software certification has been conducted manually. Given the advancement in uncertainty theories and software traceability, we envision that these technologies can synergistically be combined and leveraged to offer some degree of automation to improve the certifiers' capability to perform software certification. To this end, we present DS4AC, a novel confidence calculation framework that 1) applies the Dempster-Shafer theory to calculate the confidence of a claim; and 2) uses the vector space model to evaluate the confidence for the evidence items using traceability information. We illustrate our approach on two different applications, where safety is the key property of interest for both systems; and provide proof of concept results that demonstrate the DS4AC framework can automate portions of the evaluation of assurance cases, thereby reducing the burden of manual certification process.*

## 1. Introduction

Assurance cases [1] specify an argument structure linking different artifacts from the software development process to support assurance claims and properties, which are increasingly used in emerging standards for demonstrating system assurance [2, 3, 4], as well as certification [5]. Central to certification is the evaluation of an assurance case (AC) that requires calculating the confidence of a root claim in the AC, using a bottom-up strategy starting with confidence evaluation of leaf claims by evaluating their respective supporting evidence/solution. As such, while some techniques to assist certifiers to efficiently and effectively evaluate a system [6] , have been proposed, the certification process is hindered by the volume of domain knowledge and experience needed by certifiers. As such, most software certification, including that used in the safety-critical sectors, has been conducted manually. Even worse, as the complexity of software continues to grow, ACs are exponentially increasing in size and complexity. For example, the preliminary safety-based AC for co-operative airport surface surveillance operations is approximately 200 pages long [7], where the size is expected to grow as more detailed argument structures are considered. Furthermore, manual certification is thus not only time consuming but also error prone and expensive.

Two complementary strategies have been pursued to make the certification process more systematic and efficient. First, work has been done to support the systematic development of ACs, with a specific focus on facilitating certification. For example, safety-based AC templates have been proposed to directly link certification requirements imposed by safety standards (e.g., ISO26262) to elements of the AC template [8, 9]. Also, safety-based AC patterns have been developed to leverage the common argument structure used in ACs, where the specific argument nodes may be different from one pattern instantiation to another, depending on the certification requirements and system artifacts [10, 11, 12]. Second, in an attempt to introduce automation into the certification process, researchers have applied various mathematical/probabilistic models to approximate the bottom-up evaluation strategy used by a certifier [13, 14, 15, 16]. But these approaches still require extensive human involvement due to two obstacles. One is the confidence calculation of leaf claims. The other is how to assess the relative contribution of sub-claims (i.e., weight distribution) for parent claims.

This paper proposes the DS4AC framework that applies the Dempster-Shafter (D-S) theory [15] as an approximation of a certifier's prior certification decisions for selected ACs and leverages recurring AC argument structure to automate the certification evaluation of other structurally similar ACs. Specifically, DS4AC takes as input two assurance cases, specified in terms of the Goal Structuring Notation (GSN) [17], where the first AC has been certified as *acceptable* by the certifier, and the objective of the DS4AC framework is to calculate confidence of the second yet-to-be-certified AC as an approximation of the certifier's evaluation without requiring an actual review. DS4AC exploits the emerging use of safety templates and the development of safety pattern-based techniques, and thus requires that the two assurance cases have the same argument structure. Moreover, we also observe that many standards documents require traceability to be established between artifacts at different phases of a software development lifecycle (SDLC) [2, 3, 4, 18]. Traceability information has been successfully employed to assess the validity of environmental assumptions for safety-critical products [19]. Thus, DS4AC assumes that the leaf claims in both assurance cases should include assertions about the traceability information amongst the supporting evidence (i.e., development artifacts).

To determine how each sub-claim can independently contribute to the belief of its parent (in the form of a weight distribution), DS4AC employs the D-S theory as a confidence calculation model, where multiple sources of supporting evidence for a given claim can be combined to determine a degree of belief for the claim. The D-S approach comprises two phases: a *learning phase* and an *application phase*. In the learning phase, we assume that, since a certifier accepts a first input AC, she acknowledges its main argument structure. Moreover, she must have compared the AC with some other ACs that have different system artifacts as solutions. Specifically, when accepting the AC in Figure 1 (i), during evaluation, she might consider another system artifact such as a traceability check report between *Sys R01* and *HR4* as the new *0Sn1.1.1* solution. Accordingly, the assertion of the *0G1.7.1* leaf claim is updated to read "*Sys R01 traces to HR4…"* in the new AC. Then, she must have chosen the first AC over the other after review. As such, DS4AC generates a set of ACs, called a training data set, where each AC has the same argument structure as the first AC but is instantiated with different system artifacts for the solution/evidence, as well as their corresponding supporting leaf claims. Using the training data set, DS4AC automatically learns a weight distribution of all sub-claims so that the distribution has the acceptable AC ranked higher in the training data set than most other distributions using the D-S theory. In the application phase, DS4AC applies the learned weight distribution to a target AC to approximate the certifier's decision. For the leaf claim evaluation, we take advantage of assertions of leaf claims on the traceability information [20, 21] and then apply information retrieval techniques [22, 23] to deduce confidence values for leaf claims.

We apply DS4AC to two case studies: the Coupled Tanks System [24], and the Gear Controller System [25]. Our proof of concept results shows that DS4AC can successfully evaluate a new AC based on an initial acceptable assurance case. In summary, we make the following contributions in this paper:

- Calculate the confidence of an assurance case using the D-S theory by means of automatically inferring disjoint contributing weights from an initially-certified assurance case that has the same structure as the assurance case to be certified;
- Apply a General Vector Space Model (GVSM) [22] to evaluate the confidence of a leaf claim based on its supporting evidence nodes; and
- Illustrate the applicability of DS4AC on two cyber-physical applications obtained from the literature.

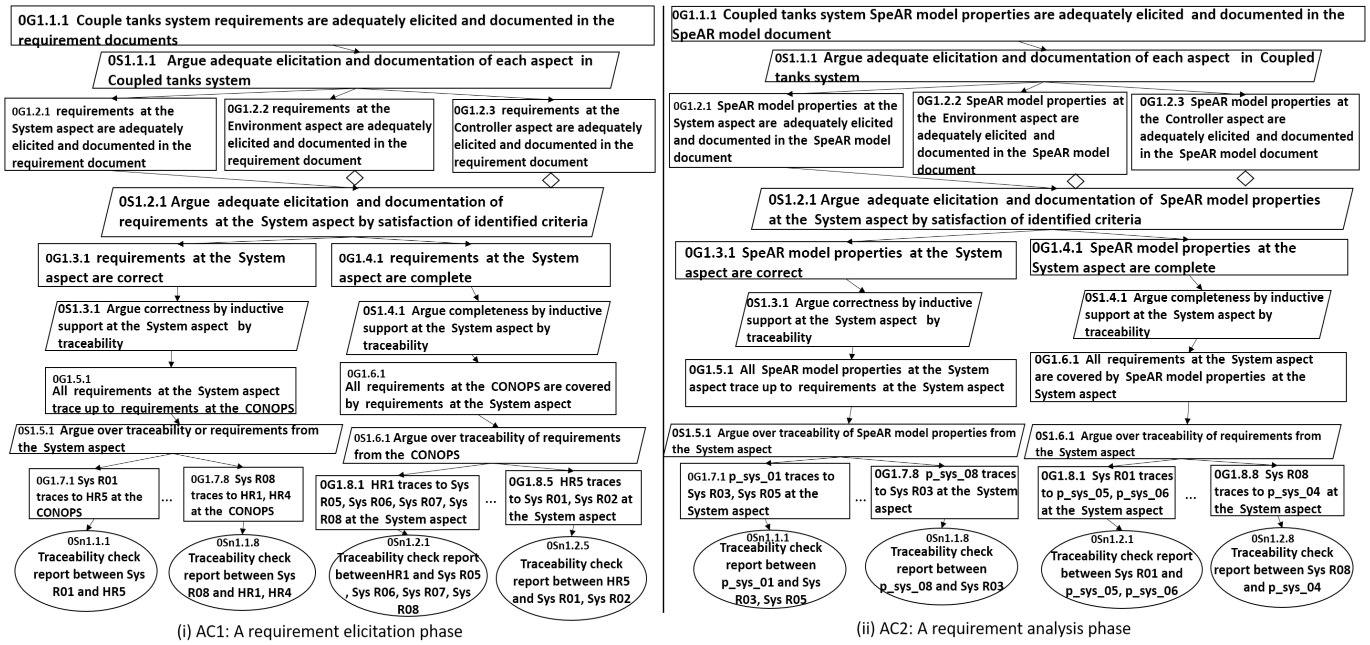
The remainder of the paper is organized as follows. Section II describes background material, including the Coupled Tanks System application as illustration of the use of GSN for specifying assurance cases, the D-S theory as a mathematical modeling for confidence calculation between sub-claims and parent claims, and the GVSM as an information retrieval technique for evaluating confidence of leaf claims. Section III presents an overview of DS4AC and some technical details for DS4AC are given in Section IV. We demonstrate the applicability of DS4AC using empirical data from the two case studies to address a research question in Section V. We discuss threats to validity in section VI and overview related work in Section VII. Finally, we summarize and draw conclusions in Section VIII.

## 2. Background

This section provides background material used for the remainder of the paper. We start with an overview of an industrial-strength application, i.e., the Coupled Tanks System as a running example. Then we briefly describe the D-S theory and the GVSM method, two key enabling technologies used for DS4AC.

### A. *Case Study: Coupled Tanks System*

The Coupled Tanks Challenge Problem was initially developed by the AFRL (Air Force Research Laboratory) to illustrate a formal methods-based early design and analysis process [24]. The Coupled Tanks System draws liquid from a limitless source, temporarily stores the liquid for a process to occur (e.g., mixing), and finally releases the liquid into a bottomless sink. Here we consider two phases: the requirements elicitation phase and the requirements analysis phase in [24]. The requirements elicitation phase starts with the concept of operations (CONOPS) that denotes five high-level requirements for the system, denoted as HR1,…, HR5, where the AFRL team derives eight system requirements, denoted as Sys R01,..,Sys R08, based on three aspects: the system, controller, and environment. Then in the second phase, the specification and analysis of requirements (SpeAR) framework is used to develop and analyze the system requirements; this process is part of the (requirements) analysis phase. Namely, the system requirements are further decomposed into eight SpeAR properties and/or requirements, denoted as p_sys_01,..,p_sys_08, respectively. In this case, we applied the safety pattern from Lin et al. [12] where variables in the safety pattern are replaced by the corresponding system artifacts and new GSN nodes are generated based on concrete artifacts. Figure 1(i) and (ii) show the two assurance cases that make claims about the requirements elicitation and requirement analysis phases, respectively. To support the top claim, rendered as a box in GSN, i.e., *0G1.1.1*, both assurance cases employ three sub-claims, i.e., *0G1.2.1*, *0G1.2.2*, and *0G1.2.3*, that refer to the system, environment, and controller aspects respectively. A strategy node, rendered as a parallelogram, e.g., *0S1.1.1*, represents how a claim, e.g., *0G1.1.1*, is supported by its sub-claims, e.g., *0G1.2.x* where $x \in \{1,2,3\}$ using the *SupportedBy* link, rendered as a line with a solid arrowhead. Next, for each sub-claim *0G1.2.x*, two sub-claims are further developed to support its correctness and completeness via a strategy node *0S1.2.1*. For instance, for the child claim "Requirements at the System Aspect are adequately elicited and documented in the requirement document", i.e., *0G1.2.1* in Figure 1(i), the correctness part claims that all system requirements at the system aspect correctly implement the high-level requirement (CONOPS) via claim *0G1.3.1*. The completeness part asserts that all high-level requirements are completely considered by the system requirements at the system aspect via claim *0G1.4.1*. Likewise, in the second assurance case shown in Figure 1(ii), for the "SpeAR model properties at the System Aspect are adequately elicited and documented in the SpeAR model document" sub-claim *0G1.2.1*, term "SpeAR model document" replaces term "requirement


0G1.1.1 Couple tanks system requirements are adequately elicited and documented in the requirement documents
0S1.1.1 Argue adequate elicitation and documentation of each aspect in Coupled tanks system
0G1.2.1 requirements at the System aspect are adequately elicited and documented in the requirement document
0G1.2.2 requirements at the Environment aspect are adequately elicited and documented in the requirement document
0G1.2.3 requirements at the Controller aspect are adequately elicited and documented in the requirement document
0S1.2.1 Argue adequate elicitation and documentation of requirements at the System aspect by satisfaction of identified criteria
0G1.3.1 requirements at the System aspect are correct
0G1.4.1 requirements at the System aspect are complete
0S1.3.1 Argue correctness by inductive support at the System aspect by traceability
0S1.4.1 Argue completeness by inductive support at the System aspect by traceability
0G1.5.1 All requirements at the System aspect trace up to requirements at the CONOPS
0G1.6.1 All requirements at the CONOPS are covered by requirements at the System aspect
0S1.5.1 Argue over traceability or requirements from the System aspect
0S1.6.1 Argue over traceability of requirements from the CONOPS
0G1.7.1 Sys R01 traces to HR5 at the CONOPS
...
0G1.7.8 Sys R08 traces to HR1, HR4 at the CONOPS
0G1.8.1 HR1 traces to Sys R05, Sys R06, Sys R07, Sys R08 at the System aspect
...
0G1.8.5 HR5 traces to Sys R01, Sys R02 at the System aspect
0Sn1.1.1 Traceability check report between Sys R01 and HR5
0Sn1.1.8 Traceability check report between Sys R08 and HR1, HR4
0Sn1.2.1 Traceability check report betweenHR1 and Sys R05 , Sys R06, Sys R07, Sys R08
0Sn1.2.5 Traceability check report between HR5 and Sys R01, Sys R02
0G1.1.1 Coupled tanks system SpeAR model properties are adequately elicited and documented in the SpeAR model document
0S1.1.1 Argue adequate elicitation and documentation of each aspect in Coupled tanks system
0G1.2.1 SpeAR model properties at the System aspect are adequately elicited and documented in the SpeAR model document
0G1.2.2 SpeAR model properties at the Environment aspect are adequately elicited and documented in the SpeAR model document
0G1.2.3 SpeAR model properties at the Controller aspect are adequately elicited and documented in the SpeAR model document
0S1.2.1 Argue adequate elicitation and documentation of SpeAR model properties at the System aspect by satisfaction of identified criteria
0G1.3.1 SpeAR model properties at the System aspect are correct
0G1.4.1 SpeAR model properties at the System aspect are complete
0S1.3.1 Argue correctness by inductive support at the System aspect by traceability
0S1.4.1 Argue completeness by inductive support at the System aspect by traceability
0G1.5.1 All SpeAR model properties at the System aspect trace up to requirements at the System aspect
0G1.6.1 All requirements at the System aspect are covered by SpeAR model properties at the System aspect
0S1.5.1 Argue over traceability of SpeAR model properties from the System aspect
0S1.6.1 Argue over traceability of requirements from the System aspect
0G1.7.1 p_sys_01 traces to Sys R03, Sys R05 at the System aspect
...
0G1.7.8 p_sys_08 traces to Sys R03 at the System aspect
0G1.8.1 Sys R01 traces to p_sys_05, p_sys_06 at the System aspect
...
0G1.8.8 Sys R08 traces to p_sys_04 at the System aspect
0Sn1.1.1 Traceability check report between p_sys_01 and Sys R03, Sys R05
0Sn1.1.8 Traceability check report between p_sys_08 and Sys R03
0Sn1.2.1 Traceability check report between Sys R01 and p_sys_05, p_sys_06
0Sn1.2.8 Traceability check report between Sys R08 and p_sys_04

(i) AC1: A requirement elicitation phase

(ii) AC2: A requirement analysis phase

Figure 1: Two assurance cases for the Coupled Tanks System

document" in Figure 1(i). Furthermore, each sub-claim *0G1.2.x* in Figure 1(ii) is further supported by two sub-claims in terms of correctness and "SpeAR model properties" replaces term "Requirements" and term completeness as its counterpart in Figure 1(i). Finally, a leaf claim such as *0G1.7.1* is supported by an evidence node, rendered as a circle, such as *0Sn1.1.1*, which is produced as a system artifact by the developer during SDLC. The difference between the two assurance cases is that *0S.1.6.1* in Figure 1(i) has 5 leaf claims, while Figure 1(ii) has 8 leaf claims representing 5 CONOPS requirements and 8 system requirements respectively. But due to space, we only show the first and last leaf claims in both cases, skipping the middle leaf claims.

## *B. Enabling Technologies*

We largely follow Wang et al's formulation for the D-S theory calculation [15]. For a claim A, a frame of discernment $\Omega_P$ is $\{A, \overline{A}\}$, where $\overline{A}$ denotes logical negation of A. The mass $m^{\Omega_A}(A)$ shows the degree of belief committed to the hypothesis that truth lies in A [15]. When applying the D-S theory, confidence of a claim A is denoted as a 3-tuple (bel(A), dis(A), uncer(A)) representing belief, disbelief, and uncertainty of A, respectively. The 3-tuple of a claim is thus defined as follows:

$$\begin{cases} \mathrm{bel}(A) & = m^{\Omega_A}(A) = g_A \\ \mathrm{dis}(A) & = \mathrm{bel}(\overline{A}) = m^{\Omega_P}(\overline{A}) = f_A \\ \mathrm{uncer}(A) & = m^{\Omega_A}(\Omega_A) = 1 - g_A - f_A \end{cases} \qquad (1)$$

A confidence model [15] consists of claims and evidence nodes that are connected to each other via the *SupportedBy* link. An argument is composed of a claim as a conclusion and the corresponding sub-claims as predicates via its all *SupportedBy* links. For instance, the top argument in Figure 1(i) consists of claim *0G1.1.1* as a conclusion and three sub-claims *0G1.2.1*, *0G1.2.2*, and *0G1.2.3*, as predicates. The confidence calculation of a claim depends on the nature of the argument relating the claim to its all sub-claims; different types of assessment parameters are used for showing the nature of an argument. The first assessment parameter is the type of an argument: either a dependent or redundant argument.

A *redundant argument* means the contribution of one sub-claim to support its parent claim does *not* depend on another (i.e., sibling) sub-claim. For example, the argument−claim A: "the system is acceptably safe" is supported by claim B: "the system is passed by verification" and claim C: "the system is passed by testing"− is redundant since two different techniques, B and C supports A to some degree without being dependent on the other.

A *dependent argument* means that the contribution of a sub-claim for supporting its parent claim has some degree of overlap with another sub-claim. For instance, the following argument − claim A: "The system is acceptably safe" is supported by claim B "the test process is sound" and claim C "the test results are correct" − is a dependent argument since the test results given by C to support claim A depend on the test process given by claim B. The second assessment parameter is the completeness of an argument, denoted as *v*, referring to a degree value between 0 and 1 and showing a scenario where its claim as a conclusion cannot be fully derived from all its sub-claims as predicates. For instance, for the above dependent argument, unless the validity of the claim made by A can be verified, we cannot completely guarantee the claim of A via any testing approach and so *v* can only have a value less than 1.

Another assessment parameter relates to the *degree of correspondence* of all sub-claims, denoted as *co*, which collectively sum to a value between 0 and 1 to capture the contributions of all sub-claims to their parent claim. All these assessment parameters are based on an argument. The final parameter is a disjoint contributing weight that differs from the other three assessment parameters because its value is set based on a claim instead of an argument. A disjoint contributing weight of a claim, say A, denoted as $w_A$, denotes a degree value

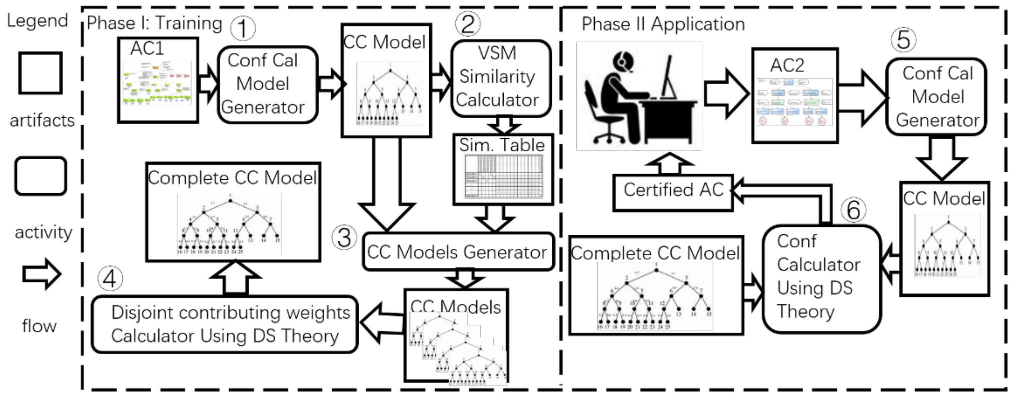


Figure 2 Overview of the DS4AC framework

between 0 and 1 showing how much the claim independently contributes to the belief/confidence of its parent claim along the argument.

Once the type of an argument is set, confidence of a claim can be calculated using the D-S theory. For example, assume the type of an argument is dependent. Then the confidence of a claim A via the 3-tuple is given by the following formula:

$$\begin{cases} \mathrm{bel(A)} = \mathrm{v}[(co)\prod_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{g_i} + \sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{g_i w_i}] = \mathrm{g_A} \\ \mathrm{dis(A)} = \mathrm{v}[(co)[1 - \prod_{\mathrm{i=1}}^{\mathrm{n}} (1 - \mathrm{f_i})] + \sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{f_i w_i}] = \mathrm{f_A} \\ \mathrm{uncer(A)} = 1 - \mathrm{g_A} - \mathrm{f_A} \end{cases} \quad (2)$$

where $\mathrm{w_i}$ (i=1,2,…,n) denotes a disjoint contributing weight of the i-th sub-claim to support its parent claim. Likewise, the formula for a redundant argument with n sub-claims can be given; due to space constraints, it is omitted but can be found in [15]. Since the number of claims in an assurance case can grow dramatically, the research goal of this paper is to find an appropriate value of all $\mathrm{w_i}$ in an assurance case.

Another format of confidence of a claim represents a certifier's perspective about the claim, say A, using a 2-tuple (*dec*(A), *conf*(A)), where *dec* denotes a certifier's confidence value and *conf* represents the confidence about the method used by a certifier in determining the value for *dec*. We thus call a *dec* value of a claim its trustworthiness value of the claim since a *dec* value directly denotes a confidence value given by a certifier. Since the 2-tuple and 3-tuple formats reflect two different perspectives on the confidence of a claim (see Figure 3), a type conversion between the two formats is necessary for different purposes. For instance, a certifier's evaluation on a leaf claim via a 2-tuple is converted to a 3-tuple so the D-S theory can be carried out. Likewise, a 3-tuple of a root claim is converted to a 2-tuple after the calculation to obtain certifier's evaluation. Formulas (3) and (4) show the conversions between the 3-tuple and 2-tuple formats of confidence of a claim. Whether an AC is acceptable or not is based on a trustworthiness value of its root claim. Following the common practice, we set 0.7 as a threshold value; and if a trustworthiness value is greater than the threshold value, then the AC is deemed as acceptable.

$$\begin{cases} bel(A) = conf(A) \times dec(A) \\ dis(A) = conf(A) \times (1 - dec(A)) \\ uncer(A) = 1 - bel(A) - disb(A) \end{cases} \quad (3)$$

$$\begin{cases} conf(P) = bel(P) + disb(P) \\ dec(P) = bel(P) \,/\, (bel(P) + disb(P)), \ \ if\ bel(P) + disb(P) \neq 0 \\ dec(P) = 0, \quad if\ bel(P) + disb(P) = 0 \end{cases} \quad (4)$$

GVSM extends the traditional VSM, an algebraic model using vectors of identifiers to represent text documents, by embedding WordNet's semantic information, besides terms, in the representation of documents. As a result, GVSM boosts text retrieval performance after using the semantic information. We apply the GVSM to deduce a confidence value of a leaf claim as an approximation of a certifier's evaluation of the leaf claim after manually reviewing its supporting solution. Following an information retrieval process, DS4AC first removes all stop words, e.g., "to" in verb phrase "traces to", in the context of a leaf claim and then uses a subject in the claim as a query and an object as a document to derive a confidence value for the claim. For instance, for the leaf claim node *0G1.7.1* in Figure 1(i), *Sys R01*, as the subject of the verb, is retrieved as a query and *HR5,* as the object of the verb, is regarded as a document; and the DS4AC calls the GVSM to get a similarity value, i.e., 1.0, as a confidence value *dec* of claim node *0G1.7.1.* As a confidence value from the certifier's perspective, we set 0.8 as a value for *conf* for all leaf claims as confidence about the GVSM in this paper. So*,* the 2-tuple (1.0,0.8) gives confidence for *0G1.7.1* from the certifier's perspective as an approximation of a human decision.

## 3. OVERVIEW OF DS4AC FRAMEWORK

This section overviews the process supported by our DS4AC framework for calculating the confidence of ACs, the generation of training data, and a confidence calculation model.

### *C. The DS4AC Framework Process*

To learn the values for disjoint contributing weights using a certifier-deemed acceptable AC (such as Figure 1(i)) followed by application of these weights to a second AC (such as that shown in Figure 1(ii)), DS4AC consists of two phases (see Figure 2):

- **The Training Phase**. First, denoted by ① in Figure 2, in order to use the D-S theory, the DS4AC converts an initial AC, termed AC1, into a confidence calculation model. Second, (denoted by ②) DS4AC uses the GVSM [22] to calculate similarity values for the confidence values for all leaf claims in a confidence calculation model. Third, (denoted by ③) DS4AC generates a set of ACs with the same structure as AC1 as training data. Finally, (denoted by ④) DS4AC applies the D-S theory to find a set of values for all the disjoint contributing weights such that AC1 is ranked highest when compared with the other sets, to indicate that the original AC is acceptable.

- **The Application Phase**. As denoted by ⑤ , DS4AC initially translates a second assurance case, termed AC2, into a confidence calculation model. Next, (see ⑥) DS4AC applies the disjoint contributing weights obtained to the confidence calculation model for AC2. DS4AC employs the D-S theory as well as the GVSM method to deduce the confidence of the AC2.

*D. Assurance Cases Generation As Training Data*

Central to the training phase is the generation of training data, i.e., a set of assurance cases with the same argument structure as the first assurance case but with different leaf claims and supporting solution nodes. To generate a new AC compared with the acceptable AC, as an approximation of a certifier's certification process, DS4AC only modifies the object of the verb in an assertion in a leaf claim but not its assertion structure. For instance, *0G1.7.1* in Figure 1 (i) asserts that *Sys R01* traces to *HR5*, i.e., the object of the verb phrase "traces to". So, DS4AC replaces *HR5* with *HRi* where $i \in \{1,2,3,4\}$, denoting the other four high-level requirements (CONOPS). Thus, the corresponding solution node is changed to a traceability check report between *Sys R01* and *HRi.* Likewise, claim *0G1.7.8* asserts the trace relationship between *Sys R08* and *HR1* and *HR4* and therefore DS4AC replaces *HR1* and *HR4* with HRi and HRj, respectively, where $i,j \in \{1,2,3,4,5\}$ and $(i,j)!=(1,4)$. Then the solution node *0Sn1.1.8* is updated accordingly. It is noted that DS4SC employs the GVSM to deduce a confidence value for a leaf claim by retrieving its subject and object as a query and a document respectively as an approximation of a certifier's evaluation on a leaf claim after manually reviewing a system artifact via the corresponding supporting solution node. Obviously, the system artifacts linked to the solutions in the ACs are not important as an approximation of a certifier's evaluation during the training phase in DS4AC.

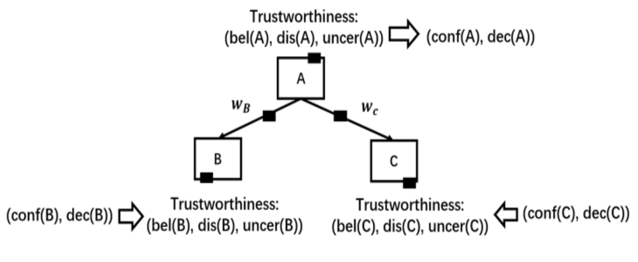


Figure 3 D-S theory for confidence calculation

*E. Confidence Calculation Model*

Since an assurance case includes auxiliary information about how an argument structure is supported (e.g., a strategy node, justification node, and context node in an assurance case), we only keep claims and evidence nodes in a confidence model and abstract away the intermediate nodes. Moreover, if a claim called c_1 is supported by only one child claim c_2 that is further supported by child claim c_3, then we can simplify the confidence calculation model by indicating that claim c_1 is directly supported by child claim c_3 without including child claim c_2. As such, the assurance case shown in Figure 1(i) is converted to the confidence calculation model shown in Figure 4.

## 4. IMPLEMENTATION OF FRAMEWORK

In this section, we provide technical details on the implementation of DS4AC.

*F. Generation of a Set of Training Data.*

To generate ACs based on the first AC1 DS4AC first generates a similarity table for each leaf claim of the AC1 using the trace information given by the leaf claim. In general, when a leaf claim describes a trace relation between n artifacts, DS4AC considers all combinations of the n artifacts from all artifacts of the same type. For instance, for *0G1.7.1* which claims "Sys R01 traces to HR5", DS4AC considers 5 CONOPS requirements − Sys R01 as a query can trace to HR1, HR2,…,HR5, each of which is a document, respectively− and then deduces five similarity values using GVSM as candidates for new leaf claims. Likewise, for *0G1.7.8* which asserts "Sys R08 traces to HR1 and HR4", DS4AC considers all combinations of two HRs out of the five HRs as documents, each of which is traced to by SysR08. So, there are 10 similarity values related to *0G1.7.8* using GVSM. Once similarity values are deduced, DS4AC sorts these values in descending order, and finally stores the best, ideal, average, and worst similarity values in a table, termed similarity table, associated with a leaf claim as a candidate set when generating a new assurance case. An ideal value is the one given by the leaf claim in AC1. This strategy is used so that no extreme training data are generated.

Figure 5 gives the algorithm creating a training data set. The method, termed *createTrainingData*, takes a first assurance case, denoted as *ac*, as well as the number of assurance cases to be generated, denoted as *s*, as input and outputs a list of

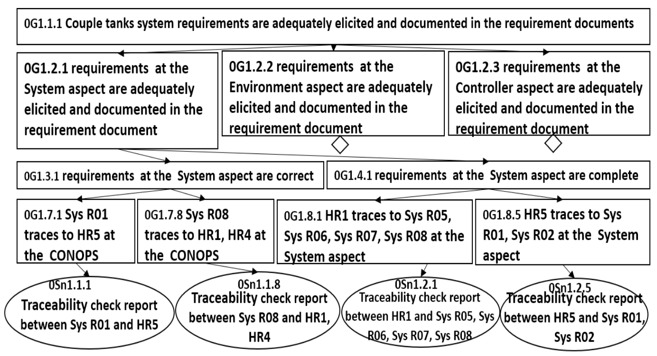

0G1.1.1 Couple tanks system requirements are adequately elicited and documented in the requirement documents
0G1.2.1 requirements at the System aspect are adequately elicited and documented in the requirement document
0G1.2.2 requirements at the Environment aspect are adequately elicited and documented in the requirement document
0G1.2.3 requirements at the Controller aspect are adequately elicited and documented in the requirement document
0G1.3.1 requirements at the System aspect are correct
0G1.4.1 requirements at the System aspect are complete
0G1.7.1 Sys R01 traces to HR5 at the CONOPS
0G1.7.8 Sys R08 traces to HR1, HR4 at the CONOPS
0G1.8.1 HR1 traces to Sys R05, Sys R06, Sys R07, Sys R08 at the System aspect
0G1.8.5 HR5 traces to Sys R01, Sys R02 at the System aspect
0Sn1.1.1 Traceability check report between Sys R01 and HR5
0Sn1.1.8 Traceability check report between Sys R08 and HR1, HR4
0Sn1.2.1 Traceability check report between HR1 and Sys R05, Sys R06, Sys R07, Sys R08
0Sn1.2.5 Traceability check report between HR5 and Sys R01, Sys R02

Figure 4 Requirement elicitation assurance calculation model

assurance cases via the list *TD* variable which initially adds an acceptable assurance case *ac* to the list at line 1. To generate each of the remaining $s$-1 assurance cases (from lines 2 to 12), the method chooses a random value (at line 3) between 1 and the number of leaf claims in the current assurance case for variable $x$ so that a new assurance case, denoted as $ac'$, has $x$ different leaf claims (from lines 5 to 7), each of whose trustworthiness values is different from their counterpart in *ac* by randomly selecting a value from the similarity table associated with a leaf claim in ac (lines 9 and 10). Lastly, the method adds $ac'$ to *TD* as a generated assurance case in the training data and set $ac'$as a new value for *ac* for generation of next assurance case if the total number of generated assurance case is less than $s$-1.

## G. Derivation of Disjoint Contributing Weights

When finding a best configuration of all disjoint contributing weights for an AC, DS4AC uses a preset step value to consider a finite number of values taken by each weight as a candidate configuration. For instance, if a step value is set to 0.1, then framework employs the following formula (5) to find all candidate configurations for $\mathrm{w}_1$, $\mathrm{w}_2$, and $\mathrm{w}_3$ if there are three sub-claims in an argument. Note that *co* is the degree of correspondence of all three sub-claims.

$$\sum_{\mathrm{i}=1}^{3} \mathrm{w}_{\mathrm{i}} = 1 - co, \text{where } \mathrm{w}_{\mathrm{i}} \in \{\, 0.1, 0.2, \dots, 0.8\} \quad (5)$$

DS4AC iterates each candidate configuration and assurance case in a training data set to ensure that the acceptable assurance case is ranked as high as possible. To do so, method *IdentifyConfiguration* as shown in Figure 6 takes the accepted assurance case as parameter *ac*, and the training data as parameter *trainingData*. The method starts with calling method *find_all_confs(ac)* to find all possible configurations according to the structure of *ac* at line 1. After initializing the variables including the outputs from line 2 to line 4, the method checks each configuration from *c* line 5. For each *c*, the method retrieves each *ac* from *trainingData*, calls method *Calculate_Using_DS (..)* to calculate a trustworthiness value of *ac* under configuration *c*, and adds the value to list *declist* from line 7 to line 9. Next, the method gets the rank of the accepted AC among *trainingData* via calling method *findRankofFirstAC (..)* at line 10. If a new rank is better than the best one via variable *best_rank*, or a new rank is equal to *best_rank* but the current configuration has a large *dec* value than the best one's via variable *trustworthiness*, then the method updates the output variables to the current configuration, rank and trustworthiness *dec* value.

There exist scenarios where sections of two assurance cases have different structures even though they are both derived from the same safety pattern (see the bottom of Figure 1). In this case, we skip the calculation of disjoint contributing weights. Instead, DS4AC assigns equal weights to all disjoint contributing weights for all children claims instead of considering all sets of disjoint contributing weights satisfying formula (5).

## H. Confidence Calculation of Second Assurance Case

DS4AC applies the derived values of disjoint contributing weights to the calculation model of the second assurance case

```
procedure CreateTrainingData
  input:1. ac: an assurance case used to generate training data and;
        2. s: the size of training data to be generated
  output: TD=$\{ac, ac_1, ac_2, \dots, ac_{s-1}\}$

1   initialize TD = $\{ac\}$;
2   for i =1 to s-1
3       initialize x = randomly select a value between 1 and ac.leafs.size
4       initialize CN =∅
5       for i = 1 to x
6           CN =CN ∪$\{ac.leaf_i\}$ where $ac.leaf_i$ is randomly chosen from
7               $ac.leafs$ and $ac.leaf_x \neq ac.leaf_y,\ \ x, y \in \{1,2,\dots,i\}$
8       initialize $ac'$= $ac$;
9       for each $ac.leaf_i$ in CN
10          $ac'.leaf_i$ = randomly select one value from $ac.leaf_i.SimilarityTable$
11      TD = TD ∪ $\{ac'\}$
12      $ac = ac'$
```

Figure 5 Create training data algorithm

whose confidence can be automatically generated instead of manually determined by a certifier. In this case, DS4AC calls method *Calculate_Using_DS(conf,ac)* where parameter *conf* is a derived set of disjoint contributing weights and the *ac* parameter is a second input assurance case. A return value is a 2-tuple value where confidence of the input assurance case is given by a trustworthiness value of the 2-tuple value.

# 5. EVALUATION OF FRAMEWORK

In this section, we evaluate DS4AC based on two case studies to address the following research question:

**RQ**: Can the set of disjoint contributing weights generated by the training phase be successfully used in the application phase in terms of the Matthews Correlation Coefficient, a popular measure of the quality of binary classification for machine learning?

Before answering the research question, we briefly introduce the Gear Controller System that was designed according to the informal requirements delivered by Mecel AB to illustrate the applicability of UPPAAL [26]. In the Gear Controller System, there are five components, each of which was modeled by a timed automaton according to the original requirements from Mecel AB. Meanwhile, safety and liveness requirements are converted from the informal description of the system to UPPAAL queries. The UPPAAL queries are further validated against the timed automata to ensure all safety and liveness requirements are satisfied. In this case study, we employ two assurance cases that claim the correct design of timed automata and derivation of the UPPAAL queries

```
Procedure IdentifyConfiguration(trainingData, ac)
Input: trainingData: a set of training data
       ac: the accepted assurance case
Output: wc: a winning set of disjoint contributing weights
        best_rank: the best rank of ac in trainingData
        trustworthiness: a trustworthiness value of ac in wc
1    Initialize wcs = find_all_confs(ac);
2    initialize best_rank = trainingData.size();
3    initialize wc = ∅
4    initialize best_dec = 0;
5    for each weight configuration c in wcs
6        initialize declist = ∅
7        for each assurance case ac in trainingData
8            initialize trustw = Calculate_Using_DS(c,ac)
9            declist.add()
10       new_rank = findRankofFirstAC(declist)
11       if(new_rank < best_rank OR (new_rank == best_rank &&
                 declist.get(0).getDec() > trustworthiness.getDec()))
12           best_rank = new_rank
13           wc = c
14           trustworthiness = declist.get(0)
```

Figure 6 Identify disjoint weights algorithm

respectively shown in Figure 7 (i) and (ii). For the assurance case to claim the design of timed automata being adequate, the assurance case starts with the top claim, i.e., "All gear controller automatons are adequately designed in the Gear Controller System". For the second assurance case to claim the derivation of UPPAAL queries are correct, we start with the top claim, i.e., "All gear controller UPPAAL queries are adequately derived in the Gear Controller System".

To support the top claim, both assurance cases employ a strategy that targets the five components in the Gear Controller System, i.e., the Interface, Gear Controller, Gearbox, Engine, and Engine components. For each component, one child claim, labelled as *0G1.2.x* where x is {1,2,…,5}, is formed to ensure the automaton for the component is adequately designed. Due to space constraints, we only show three of the five components in Figure 7.

For each sub-claim *0G1.2.x*, two sub-claims are further developed to support its correctness and completeness via a strategy node *0S1.2.x*. For the assurance case on automata, the correctness child claim ensures that the automaton of a component traces to the corresponding component description, and the completeness child claim asserts that the component description matches the automaton in terms of similarity. For the assurance case on the UPPAAL queries, the correctness child claim ensures that the UPPAAL queries related to a specific component can trace to the related system requirements, and the completeness child claim asserts that the system requirements related to a component match the UPPAAL queries in terms of similarity. At the bottom part of the two assurance cases, both the correctness claim and the completeness claim are supported by traceability check reports between the related system artifacts.

**Answer to RQ.** We adopt the Matthews Correlation Coefficient (MCC) [27] that, as a measure of the quality of binary (two-class) classifications, has been regarded as a better measure than many popular evaluation measures such as Recall, Precision, F-Factor, and Rand Accuracy in machine learning [28]. The MCC, calculated by Formula (6), has also been applied in software engineering [29]. Considering true and false positives and negatives, the MCC is essentially a correlation coefficient between the observed and predicted binary classifications; it returns a value between −1 and +1. A coefficient of +1 represents a perfect prediction, 0 means no better than random prediction, and −1 indicates total disagreement between prediction and observation. The closer to 1 a MCC value is, the more accurate a prediction is. In our scenario, the MCC is employed to indicate whether the derived set of disjoint contributing weights by DS4AC is better than a randomly generated set.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP+FP)(FP+FN)(TN+FP)(TN+FN)}} \quad (6)$$

To find whether a derived set of the disjoint contributing weights from the training phase can be successfully used in the application phase, we manually generate 100 assurance cases with the same structure as the second assurance case. The first criterion of the manual review is based on the average trustworthiness value of the entire assurance case. We first set a threshold value to be an average trustworthiness value of all

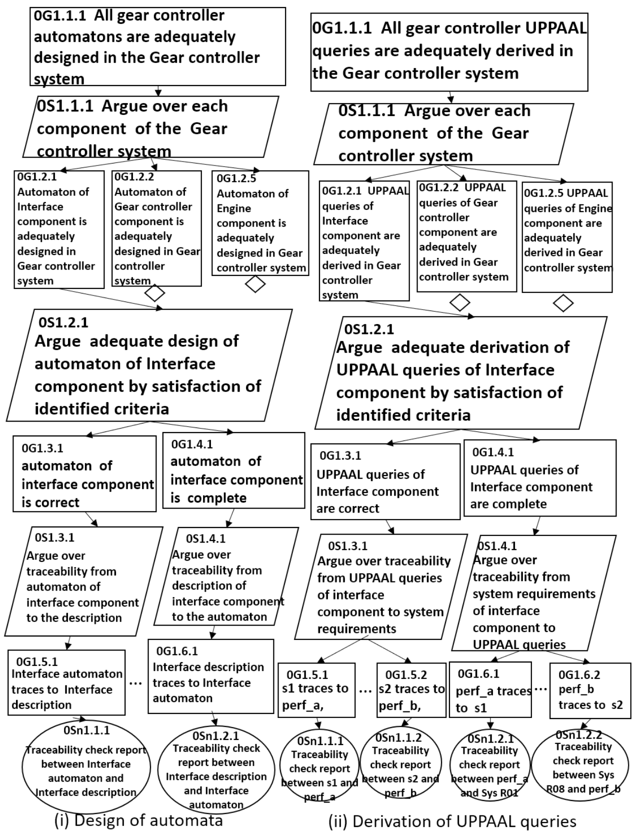

0G1.1.1 All gear controller automatons are adequately designed in the Gear controller system
0S1.1.1 Argue over each component of the Gear controller system
0G1.2.1 Automaton of Interface component is adequately designed in Gear controller system
0G1.2.2 Automaton of Gear controller component is adequately designed in Gear controller system
0G1.2.5 Automaton of Engine component is adequately designed in Gear controller system
0S1.2.1 Argue adequate design of automaton of Interface component by satisfaction of identified criteria
0G1.3.1 automaton of interface component is correct
0G1.4.1 automaton of interface component is complete
0S1.3.1 Argue over traceability from automaton of interface component to the description
0S1.4.1 Argue over traceability from description of interface component to the automaton
0G1.5.1 Interface automaton traces to Interface description
0G1.6.1 Interface description traces to Interface automaton
0Sn1.1.1 Traceability check report between Interface automaton and Interface description
0Sn1.2.1 Traceability check report between Interface description and Interface automaton
(i) Design of automata
0G1.1.1 All gear controller UPPAAL queries are adequately derived in the Gear controller system
0S1.1.1 Argue over each component of the Gear controller system
0G1.2.1 UPPAAL queries of Interface component are adequately derived in Gear controller system
0G1.2.2 UPPAAL queries of Gear controller component are adequately derived in Gear controller system
0G1.2.5 UPPAAL queries of Engine component are adequately derived in Gear controller system
0S1.2.1 Argue adequate derivation of UPPAAL queries of Interface component by satisfaction of identified criteria
0G1.3.1 UPPAAL queries of Interface component are correct
0G1.4.1 UPPAAL queries of Interface component are complete
0S1.3.1 Argue over traceability from UPPAAL queries of interface component to system requirements
0S1.4.1 Argue over traceability from system requirements of interface component to UPPAAL queries
0G1.5.1 s1 traces to perf_a,
0G1.5.2 s2 traces to perf_b,
0G1.6.1 perf_a traces to s1
0G1.6.2 perf_b traces to s2
0Sn1.1.1 Traceability check report between s1 and perf_a
0Sn1.1.2 Traceability check report between s2 and perf_b
0Sn1.2.1 Traceability check report between perf_a and Sys R01
0Sn1.2.2 Traceability check report between Sys R08 and perf_b
(ii) Derivation of UPPAAL queries

Figure 7 Assurance cases of the Gear Controller System

leaf claims in the 100 assurance cases. The rationale behind this criterion is that if an average trustworthiness value is too low, then it is impossible for the case to be accepted. Next, we select all assurance cases whose average trustworthiness value is greater than the threshold value for further manual review. For the Coupled Tanks System assurance cases, there are a total of 45 assurance cases selected for manual review and the other 55 assurance cases are immediately rejected. For the second phase of manual review, after communicating with the developers about the system design multiple times, we find the system aspect artifacts play an important role in the early design and analysis process in that the system aspect, as the first step of the process, is further decomposed into the environmental aspect and controller aspect. Thus, we further review the 45 assurance cases selected from the first phase. Among the 45 assurance cases, we manually select 25 assurance cases for the acceptable set based on the assurance structure under the first branch, *i.e.* the claim *0G1.2.1* in Figure 1(ii).

For the Gear Controller System assurance cases, we employ the same criterion for the first phase by calculating a threshold value based on an average trustworthiness value of all leaf claims in the 100 assurance cases. Thus, there are a total of 44 assurance cases selected for manual review and the remaining 56 assurance cases are immediately rejected due to a low average trustworthiness value. Next, we study the five components in the Gear Controller System and concentrate on the gear controller artifacts since the gear controller is the main part in this case study, as is confirmed by one of the experts

TABLE 1 MCC VALUE IN APPLICATION PHASE

| | TP | TN | FP | FN | MCC |
|---|---|---|---|---|---|
| Coupled Tanks | 22 | 71 | 4 | 3 | 0.816072 |
| Gear Controller | 13 | 77 | 9 | 1 | 0.690144 |

who was actively involved in the design and verification of the case study. Based on this criterion, after reviewing the 44 assurance cases, we manually select 14 assurance cases for the acceptable set.

Next, we calculate MCC values for both case studies by applying the derived set of the disjoint contributing weights from the training phase to the 100 assurance cases. First, we find the numbers of true positives, false positives, true negatives, and false negatives for both studies, followed by the MCC calculation using formula (6). All the information is shown in Table 1. From the MCC values for both case studies, we conclude that the derived set of the disjoint contributing weights from the training phase is closer to a perfect prediction than a random prediction.

## 6. THREATS TO VALIDITY

Several potential threats to validity exist. First, in order to address the threat that DS4AC may not be applied to the evaluation of assurance cases in multiple safety critical domains, we applied DS4AC to two case studies from two different application areas, but both are the safety-critical domains where the safety property is the main issue in both systems. While we cannot claim that DS4AC is able to evaluate all assurance case correctly, we are confident that the tools such as our framework should be able to assist certifiers to review an assurance case by reducing the burden of manual review.

Another threat to validity is to address the RQ, we manually select the acceptable assurance cases as well as the rejected assurance cases for both case studies. But, we had extensive interactions with the AFRL team who worked on the Coupled Tanks System. As for the Gear Controller System, we received extensive feedback about the case study from one of the experts who involved in the design and verification of the Gear Controller System. Furthermore, it is important to note that this type of research study is a critical precursor to building an industrially relevant framework that can finally help certifiers to automatically deduce confidence of an assurance case.

A third threat is the application of traceability and information retrieval techniques to check the confidence of leaf claims in an assurance case. Since traceability and information retrieval techniques have been widely and successfully employed in various phases of an SDLC [6, 30, 31, 32, 33, 34], and integrating the traceability into an assurance case [35] has been recently proposed. Application of traceability and information retrieval techniques seems to be a promising direction for certification purposes. As part of future work, we will investigate whether the accuracy of the confidence evaluation can be improved by adopting other techniques such as static analysis and verification techniques.

A final threat is the application of the D-S theory to approximate the confidence calculation performed by certifiers. While other mathematical models can serve this purpose, the configuration of assessment parameters in these models remains a challenging issue. Nevertheless, our framework should still be applicable for these models, but further studies are needed to confirm this claim.

## 7. RELATED WORK

Work on confidence and assurance cases dates back to 2003 when Bloomfield and Littlewood raised a question as to how the degree of confidence in a case can be affected [36]. Then Bloomfield et al. [37] proposed that confidence of an assurance case can be related to the concept of safety integrity levels (SILs) that measure the risk of dangerous failure in safety-critical systems. However, some researchers have directly addressed the confidence issue. For instance, Hawkins et al. [38] proposed a new method which splits an assurance case into two pieces of information. The first piece is the safety argument that shows the desired safety property using evidence. The second piece is the confidence argument showing that a degree of confidence in the safety argument is supported by an argument and evidence. The confidence argument is to address uncertainties that underlie the safety argument. For simplicity, Hawkins et al. combined a confidence argument into a safety argument by the use of assurance claim points.

Evaluation of assurance cases is not new to the safety critical sector. Various uncertainty theories and models have been applied to deduce confidence of an assurance case. Goodenough et al. [39] extended Hawkins et al.'s work by quantifying confidence as a Baconian probability. Goodenough et al. used a Baconian probability ratio as a pair of integers where a number of assurance deficits ("doubts" or "defeaters") that have been eliminated or mitigated is the numerator-like value while the total number of doubts identified is the denominator-like value. The Baconian probabilities are summed up from the pairs of leaf claims to a root claim, leading to a confidence probability value for the entire assurance case.

Most approaches tailor a theory or model to an assurance case in order to calculate a confidence value. For instance, Wang et al. [15] successfully revised the Dempster-Shafer (D-S) theory to ensure that can be used as a main calculation model to deduce the confidence of an assurance case. However, when performing confidence propagation, all the assessment parameters should be provided as input including the disjoint contributing weights of all children claims. At the same time, the leaf claim evaluation must be done manually by domain experts. In fact, this approach epitomizes most current research about evaluation of an assurance case using various types of mathematical models. For instance, Denney et al. [13] applied the Bayesian paradigm for uncertainty modelling and assessment. Duan et al. [14] considered the application of the Beta distribution as Baconian Probabilities. However, all the current approaches require human input to configure assessment parameters such as how a child claim contributes to the belief of its parent claim.

Support of assurance case evolution has also drawn great interests in the research community due to the importance of system evolution. Kokaly et al. [40] proposed the application of model evolution, which has been widely studied by the model driven engineering (MDE) community to understand why models change and how that impacts consistency of related models. This information is used to generate a new assurance case by reusing the current assurance case as much as possible when a system evolves. Huang et al. [35] proposed the establishment of traceability among assurance cases for a safety critical system. When an artifact change is detected, properties related to an assurance case are thus analyzed by information retrieval techniques, refactoring crawler, identified patterns, and static analysis tools. Once a property is violated, the corresponding argument in the assurance case should be updated. These approaches do not directly address confidence evaluation of an assurance case as we do in this paper.

**Acknowledgements.** The research described in this paper has been supported by funding from the Air Force Research Laboratory under agreement numbers FA8750-19-2-007, FA8750-16-2-0284, and FA-8750-19-2-0002; NSF Grant DBI-0939454; and support from Ford and General Motors. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory, the U.S. Government, or industrial sponsors.

## 8. CONCLUSIONS

While software certification requires human judgement, our results show that software certification is amenable to automation when the D-S model is used to approximate a certifier's domain model and experience. We are exploiting how the traceability and information retrieval techniques can be combined with other static analysis and verification tools to further improve the accuracy of the confidence achievable within DS4AC. Lastly, we are investigating how DS4AC can be integrated into assurance case evolution especially for autonomous cyber physical systems.

## 9. REFERENCES

[1] J. Rushby, "Assurance and Assurance Cases," in *Dependable Software Systems Engineering (Marktoberdorf Summer School Lectures, 2016)*, IOS Press, 2016, pp. 207-236.

[2] Organización Internacional de Normalización, ISO 26262: Road Vehicles : Functional Safety, ISO, 2011.

[3] Food and Drug Administration (FDA), "Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices," 2005.

[4] European Committee for Electrotechnical Standardization, "CSN EN 50129 - Railway applications - Communication, signalling and processing systems - Safety-related electronic systems for signalling," 2003. [Online]. Available: https://www.en-standard.eu/csn-en-50129-railway-applications-communication-signalling-and-processing-systems-safety-related-electronic-systems-for-signalling/. [Accessed 4 July 2018].

[5] J. Rushby, "The Interpretation and Evaluation of Assurance Cases," Computer Science Laboratory, SRI International, SRI-CSL-15-01, Menlo Park, CA, 2015.

[6] L. Briand, D. Falessi, S. Nejati, M. Sabetzadeh and T. Yue, "Traceability and SysML design slices to support safety inspections: A controlled experiment," *ACM Transactions on Software Engineering and Methodology (TOSEM),* vol. 23, no. 1, pp. 9:1-9:43, 2014.

[7] EUROCONTROL—European Organisation for the Safety of Air Navigation, "Preliminary Safety Case for ADS-B Airport Surface Surveillance Application, V 1.2," [Online]. Available: https://www.eurocontrol.int/sites/default/files/publication/files/surveillance-cascade-preliminary-safety-case-for-airports-surface-surveillance-applications-201111.pdf.

[8] T. Chowdhury, C.-W. Lin, B. Kim, M. Lawford, S. Shiraishi and A. Wassyng, "Principles for Systematic Development of an Assurance Case Template from ISO 26262," in *Proceedickngs of 2017 IEEE International Symposium on Software Reliability Engineering, Industry Track*, Toulouse, France, 2017.

[9] Y. Zhang, *A Safety Argument Template for Medical Device Software,* Personal Email, 2016.

[10] E. W. Denney and G. J. Pai, "Safety Case Patterns: Theory and Applications," NASA/TM–2015–218492, 2015.

[11] R. Hawkins, I. Habli, D. Kolovos, R. Paige and T. Kelly, "Weaving an Assurance Case from Design: A Model-Based Approach," in *Proc. of HASE'15*, Daytona Beach, FL, 2015.

[12] C.-L. Lin, W. Shen and S. Drager, "A Framework to Support Generation and Maintenance of an Assurance Case," in *IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2016.

[13] E. Denney, P. Ganesh and H. Ibrahim, "Towards Measurement of Confidence in Safety Cases," in *Proceedings of ESEM'11*, Banff, Alberta, Canada, 2011.

[14] L. Duan, S. Rayadurgam, M. Heimdahl, O. Sokolsky and I. Lee, "Representation of Confidence in Assurance Cases Using the Beta Distribution," in *Proceedings of HASE'16*, Orlando, FL,, Jan, 2016.

[15] R. Wang, J. Guiochet and G. Motet, "Confidence Assessment Framework for Safety Arguments," in *Proc. of SafeComp'17*, Trento, Italy, 2017.

[16] E. Denney and G. Pai, "Tool support for assurance case development," *Automated Software Engineering,* vol. 25, no. 3, pp. 435-499, 2018.

[17] Goal Structuring Notation Working Group, "GSN Community Standard Version 1," 2011.

[18] RTCA, Inc., "DO-178C: SOFTWARE CONSIDERATIONS IN AIRBORNE SYSTEMS AND EQUIPMENT," 2011.

[19] M. Rahimi, W. Xiong, J. Cleland-Huang and R. Lutz, "Diagnosing assumption problems in safety-critical products," in *Proc. of the International Conference on Automated Software Engineering (ASE 2017)*, 2017.

[20] B. Ramesh and M. Jarke, "Toward reference models for requirements traceability," *IEEE transactions on software engineering,* vol. 27, no. 1, pp. 58-93, 2001.

[21] J. Cleland-Huang, O. C. Z. Gotel, J. Huffman Hayes, P. Mäder and A. Zisman, "Software traceability: trends and future directions," in *Proc. of the Future of Software Engineering (FOSE 2014)*, 2014.

[22] G. Tsatsaronis and V. Panagiotopoulou, "A Generalized Vector Space Model for Text Retrieval Based on Semantic Relatedness," in *Proc. of EACL'09*, 2009.

[23] T. K. Landauer, Latent Semantic Analysis, John Wiley & Sons, Ltd, 2006.

[24] K. H. Gross, A. W. Fifarek and J. A. Hoffman, "Incremental Formal Methods Based Design Approach Demonstrated on a Coupled Tanks Control System," in *Proceedings of HASE'16*, Orlando, FL,, 2016.

[25] M. Lindahl, P. Pettersson and Y. Wang , "Formal design and analysis of a gear controller," *Software Tools for Technology Transfer,* vol. 3, no. 3, pp. 353-368, 2001.

[26] J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson and Y. Wang, "UPPAAL — a tool suite for automatic verification of real-time

systems," in *Proc. of the International Hybrid Systems Workshop*, 1995.

[27] B. W. Matthews, "Comparison of the predicted and observed secondary structure of T4 phage lysozyme," *Biochimica et Biophysica Acta (BBA) - Protein Structure,* vol. 405, no. 2, pp. 44-451, 1975.

[28] D. Powers, "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation," *Journal of Machine Learning Technologies.,* vol. 2, no. 1, p. 37–63, 2011.

[29] R. Gopalakrishnan, P. Sharma, M. Mirakhorli and M. Galster, "Can latent topics in source code predict missing architectural tactics?," in *Proc. of the 39th International Conference on Software Engineering ( ICSE)*, Buenos Aires, Argentina, 2017.

[30] D. Poshyvanyk, M. Gethers and A. Marcus, "Concept location using formal concept analysis and information retrieval," *ACM Transactions on Software Engineering and Methodology ,* vol. 21, no. 4, pp. 23:1-23:34, 2012.

[31] M. Mirakhorli and J. Cleland-Huang, "Detecting, Tracing, and Monitoring Architectural Tactics in Code," *IEEE Transactions on Software Engineering,* vol. 42, no. 3, pp. 205-220, 2015.

[32] X. Yang, D. Lo and X. Xia, "Combining Word Embedding with Information Retrieval to Recommend Similar Bug Reports," in *Proc. of the International Symposium on Software Reliability Engineering (ISSRE)*, 2016.

[33] J. Guo, J. Cheng and J. Cleland-Huang, "Semantically enhanced software traceability using deep learning techniques," in *Proc. of the International Conference on Software Engineering (ICSE)*, Buenos Aires, 2017.

[34] V. Arnaoudova, S. Marcus, A. Marcus and G. Antoniol, "The Use of Text Retrieval and Natural Language Processing," in *International Conference on Software Engineering (ICSE)*, 2015.

[35] J. Cleland-Huang and R. Lutz, "Traceability Support For Evolving Safety Assurance Cases," Air Force Research Laboratory's Safe & Secure Systems and Software Symposium (S5) 2017, Dayton, Ohio, 2017.

[36] R. Bloomfield and B. Littlewood, "Multi-Legged Arguments: The Impact of Diversity upon Confidence," in *Proc. of the International Conference on Dependable Systems and Networks (DSN 2003)*, San Francisco, 2003.

[37] R. E. Bloomfield, B. Littlewood and D. Wright, "Confidence: Its Role in Dependability Cases for Risk Assessment," in *Proc. of the International Conference on Dependable Systems and Networks (DSN 2007)*, Edinburg, 2007.

[38] R. Hawkins, T. Kelly, J. Knight and P. Graydon, "A New Approach to creating Clear Safety Arguments," in *Advances in Systems Safety*, 2011.

[39] J. Goodenough, C. B. Weinstock and A. Z. Klein, "Toward a theory of assurance case confidence," CMU/Software Engineering Institute Technical Report CMU/SEI-2012-TR-002 ESC-TR-2012-002, 2012.

[40] S. Kokaly, R. Salay, V. Cassano, T. Maibaum and M. Chechik, "A model management approach for assurance case reuse due to system evolution," in *Proc. of the International Conference on Model Driven Engineering Languages and Systems (MODELS' 16)*, 2016.