Towards Developing Formalized Assurance Cases

Baoluo Meng
GE Research
Niskayuna, NY, USA
baoluo.meng@ge.com

Abha Moitra
GE Research
Niskayuna, NY, USA
moitra@ge.com

Andrew W. Crapo
GE Research
Niskayuna, NY, USA
crapo@ge.com

Saswata Paul
GE Research
Niskayuna, NY, USA
saswata.paul@ge.com

Kit Siu
GE Research
Niskayuna, NY, USA
siu@ge.com

Michael Durling
GE Research
Niskayuna, NY, USA
durling@ge.com

Daniel Prince
GE Aviation
Niskayuna, NY, USA
daniel.prince@ge.com

Heber Herencia-Zapana
GE Research
Niskayuna, NY, USA
heber.herencia-zapana@ge.com

Abstract— **The ever-increasing complexity of cyber physical systems drives the need for assurance of critical infrastructure and embedded systems. Building assurance cases is a way to increase confidence in systems. In general, the construction of assurance cases is a manual process and the resulting artifacts are not machine analyzable. The High Assurance Systems team at GE Research is developing technology to support generation of formalized assurance cases for systems, which are both human-readable and machine-analyzable. We have developed a Semantic Application Design Language Assurance Toolkit (SADL-AT) including a semantic model to formalize the Goal Structuring Notation for assurance cases. This paper describes the toolkit SADL-AT and demonstrates the capabilities and effectiveness of SADL-AT by building security and safety assurance case fragments for an unmanned aerial vehicle-based example – a delivery drone.**

Keywords—assurance case, GSN, SADL

I. INTRODUCTION

Software systems are prevalent in various areas of our life such as medicine, transportation, energy, finance, military and so on. The ever-increasing complexity of software systems drives the need for assurance of critical infrastructure and embedded systems. Testing [20] [23] has been utilised as the primary way to increase stakeholders' trust in systems, whereas building assurance cases is an another way to boost the trustworthiness of systems. In general, the construction of assurance cases is a manual process and the resulting artifacts are not machine analyzable. The High Assurance Systems team at GE Research is developing technology to support generation of formalized assurance cases for software systems, which are both human-readable and machine-analyzable. We have developed a Semantic Application Design Language Assurance Toolkit (SADL-AT) including a semantic model to formalize the Goal Structuring Notation for assurance cases. The assurance case fragments built using SADL can be queried and visualized using the SADL reasoning engine. An assurance case is a structured argument tree, where leaf nodes are evidences; non-leaf nodes are strategies, context and goals (claims); non-leaf nodes are only connected with either leaf nodes or non-leaf nodes; and a goal (or a sub-goal) node is justified by its immediate children nodes. An assurance fragment is a composable piece of the assurance case for the overall system. The ultimate goal of generating assurance case fragments is to provide the building blocks for establishing an assurance case for the overall system. The toolkit is based on structured natural language created on top of the Semantic Application Design Language (SADL) tool [1]. By adopting the Goal Structuring Notation (GSN) [2], SADL-AT enables users to capture, visualize and analyze goals, strategies, assumptions, justifications, contexts and evidences. This paper describes the toolkit SADL-AT and demonstrates the capabilities and effectiveness of SADL-AT by building safety and cyber-resiliency assurance case fragments for an unmanned aerial vehicle-based use case – *delivery drone*. A cyber resiliency assurance case can deliver guarantees of a system's resilience to cyber-attacks. The goal of a safety assurance case is to ensure a system is acceptably safe in case of component failures. Evidences for goals are collected from the model-based architectural safety and security analysis of the VERDICT tool (Verification Evidence and Resilient Design in Anticipation of Cybersecurity Threats) [4]. Finally, we show that assurance case fragments built using SADL-AT can be queried and analyzed using the SADL reasoning engine.

II. PRELIMINARIES

In this section, we will provide some background on assurance cases, the Semantic Application Design Language (SADL) and the VERDICT tool.

A. Assurance Cases

Assurance cases [5] were referred to as safety cases in the past, but nowadays they have broader applications and have regained a lot of interests from research fields and industries. An assurance case by definition is a reasoned and compelling argument, supported by a body of evidence, that a system will operate as intended in a defined environment. It is a structured argument tree, where leaf nodes are evidence; non-leaf nodes are strategies, context and goals (claims); non-leaf nodes are connected with leaf nodes or non-leaf nodes; and goal (claim)

978-1-7281-9825-5/20/$31.00 ©2020 IEEE

nodes are justified by its immediate children nodes under certain contexts. Assurance cases have been used for expressing the safety, security and reliability of systems and products in given contexts, and ultimately for their certifications. Two approaches are normally adopted to establish assurance cases for systems or products: *standards-based* and *product-based*. The standards-based approach evaluates whether a system satisfies certain security or safety standards such as DO-178B for avionics safety. The product-based approach creates an assurance case with explicit claims about system behavior. Supporting evidences are linked to the claims via arguments. Established assurance cases are evaluated by independent assessors. Several approaches have been developed over the years to construct assurance cases including CAE (Claim, Argument and Evidence) [6], GSN (Goal Structuring Notation), and SACM (Structured Assurance Case Metamodel) [7]. CAE is an earlier methodology for safety case development, where claim, argument and evidence are the main constituents of a safety case. SACM defines a metamodel released by the Object Management Group (OMG) for representing structured assurance cases. GSN, a graphical argument notation, further expands CAE with additional support for encoding assumptions, justifications, contexts and models to help organize and structure assurance cases in a readily reviewable form. GSN has been widely used in safety case development for aircraft avionics, rail signaling, air traffic control, and nuclear reactor shutdown. Several assurance case tools based on GSN have been developed including AdvoCATE [8] and Resolute [9].

B. Semantic Application Design Language

Semantic Application Design Language (SADL) is a controlled-English grammar that expresses Web Ontology Language (OWL) [10] ontologies plus rules. SADL expressivity is equivalent to OWL 1 plus qualified cardinality from OWL 2. SADL is also an integrated development environment (IDE), available as a set of Eclipse plugins or as a set of services providing a web browser-based interface for creating, testing, and maintaining semantic models. The grammar includes constructs for querying, testing, and modeling lifecycle management. SADL is open source and has been used in a wide variety of applications. The team has previously extended the SADL grammar to create the SADL Requirements Language (SRL), which enables capturing requirements in controlled English. The SADL IDE has been extended to create the Requirements Capture Environment of ASSERT™ [11]. Requirements are translated to a representation appropriate as input to theorem provers for requirement analysis and to SMT solver based automated test generation tools [20].

C. VERDICT Tool

GE Research, with support from the DARPA Cyber Assured Systems Engineering (CASE) program, collaborated with GE Aviation Systems and the Unversity of Iowa to create the Verification Evidence and Resilient Design in Anticipation of Cybersecurity Threats (VERDICT) [16] tool. The VERDICT tool consists of two major analysis capabilities: Cyber-Resillience Verification (CRV) and Model-Based Architecture Analysis (MBAA). MBAA enables system engineers to model, jointly analyze safety and security based on architectural models and mission scenarios, generate fault and attack-defense trees, and synthesize an architecture that meets all the design constraints. The attacks are based on Mitre's Common Attack Pattern Enumeration and Classification (CAPEC™) [3] set and the defenses are based on the NIST 800-53 [18] Security and Privacy Controls for Federal Information Systems and Organizations with accomadations made for application to critical embedded systems. In this paper, we will mainly use the MBAA (Model Based Architecture Analysis) feature of VERDICT to generate evidence for safety and security assurance case fragments.

The VERDICT tool can provide assurance case fragements that may be used to assemble a more complete assurance case. For example, the formal mission, cyber and safety requirements, and formal behavioral properties may be used as a hierarchy of goals. The text based descriptions of the formal requiremets and behavioral properties captured in the AADL model may be used as context to help clarify the goals. The AADL model and application of the VERDICT tool may be used as a strategy to gain trust in the architectural and behavioral models. The fault trees, calculated probability of failure, attack-defense tree, likelihood of successful attack metrics, formal proofs and merit asignment output may be used as evidences to support the formal mission, cyber, safety and behavioral property goals identified in the model.

III. Semantic Application Design Language Assurance Toolkit

In this section, we will describe our toolkit – SADL-AT, which is built on top of SADL. We leverage our prior experience in formalizing systems and requirements in controlled English grammars that are both easily understandable by humans and unambiguously translatable to formal languages that can be analyzed using formal methods. The formal representation to which we translate is OWL . OWL constructs are presented in SADL.

A. Key Elements of an Assurance Case in GSN

GSN is a graphical argumentation notation used to document the structure of any argument in terms of claims, evidence, and the context in which the argument is claimed to hold. GSN is described in the GSN Community Standard, Version 2, 2018 [12]. The core elements of GSN are as follows.

- **Goals** – the claims and sub-claims of an argument
- **Strategies**—the nature of the argument connecting claims to sub-claims
- **Solutions**—items of evidence
- **Contexts**—the context (conditions) in which the claims or reasoning steps should be interpreted
- **Assumptions**—conditions which must hold for the claim or strategy to be valid
- **Justifications**—explanations of why a claim or argument is considered acceptable

This research was partially funded by the Defense Advanced Research Projects Agency (DARPA).

B. A Semantic Model of GSN in SADL

```
GSN (alias "Goal Structuring Notation") (note "a graphical
argument notation, capturing a hierarchy of claims")
  is a type of Argument,
  described by rootGoal with a single value of type Goal.
GSN_Element (note "Notation elements of GSN, see
https://scsc.uk/r141B:1?t=1") is a class.
 {Goal (note "a requirements statement; a claim forming
    part of an argument"),
  Strategy (note "mediates between a goal and its subgoals,
    explaining basis of inference"),
  Solution (note "presents a reference to an evidence
    item"),
  Context (alias "scope over which claim of goal is made
    or strategy applies") (note "applies to argument
    substructure without repetition"),
  gsn:Assumption (alias "an unsubstantiated statement")
    (note "a goal or strategy in the context of an
     assumption is only applicable when the assumption is
     true")
    (note "applies to argument substructure without
     repetition")
    (note "complete sentences in the form of a noun-phrase
     + a verb-phrase"),
  Justification (alias "rational for the associated claim
    of the goal or strategy") (note "applies only to goal
    or strategy and not to argument substructure")
    (note "complete sentences in the form of a noun-phrase
     + a verb-phrase")} are types of GSN_Element.
UndevelopedElement is a type of GSN_Element.
UndevelopedGoal is a type of {Goal and
UndevelopedElement}.
UndevelopedStrategy is a type of {Strategy and
UndevelopedElement}.
```

Fig. 1. The SADL-AT snippet of GSN classes.

In our semantic model of GSN, we create a *class* for each of these core elements with the same name except in singular form, e.g., the class *Goal*. Fig. 1 shows the SADL code for defining these classes. There are two relationship types defined in the GSN Standard.

- **SupportedBy** – an inferential or evidential relationship
- **InContextOf** – a contextual relationship

```
supportedBy (note "indicated by solid arrow head")
describes {Goal or Strategy} with values of type {Goal or
Strategy or Solution}.
inContextOf (note "indicated by hollow arrow head",
"Kelly's thesis included Solution in the domain")
describes {Goal or Strategy} with values of type {Context
or Assumption or Justification}.
```

Fig. 2. The SADL snippet for GSN relationships.

In our semantic model of GSN, we create a property for each of these with the same name except that we follow the convention of using property names that start with a lowercase letter, making our properties *supportedBy* and *inContextOf*. Fig. 2 shows the SADL code for defining the properties, domains, and ranges. The expression of domain and range only identify the possible types of things that can have these properties and the types of values the properties can have. It is insufficient to completely capture the GSN standard's content.

According to the GSN standard, the following types of arcs, defined by the arc label (property) and the class to which the head (subject) and the tail (Object) belong, are allowed in a GSN graph. Summarizing these in terms of property restrictions in OWL but expressed in SADL, we have the axioms defined in Fig. 3.

```
supportedBy (note "indicated by solid arrow head")
describes {Goal or Strategy} with values of type {Goal or
Strategy or Solution}.
inContextOf (note "indicated by hollow arrow head",
"Kelly's thesis included Solution in the domain")
describes {Goal or Strategy} with values of type {Context
or Assumption or Justification}.
```

Fig. 3. Axioms for the domain and range of properties.

There are two types of *contexts* in GSN.

- **ReferentialContext** A context may be a reference to an artefact that informs the "reasoning step", in which case the context text will be a noun-phrase. We will represent this type of context with the subclass *ReferentialContext*.
- **ExplanatoryContext** A context may also be a statement drawing attention to explanatory information, such as a definition, in which case the text will be a noun-phrase plus a verb-phrase. We represent this with the subclass *ExplanatoryContext*.

```
{ReferentialContext (alias "referential context")
  (note "reference is a noun-phrase"), ExplanatoryContext
  (alias "explanatory contextual information")
  (note "complete sentences of form a noun-phrase + a
verb-
  phrase")} are types of Context.
  reference describes ReferentialContext with a single
value of type class.
Reference is a class, described by description with
values of type string,
  described by content with values of type
InformationObject,
  described by location with values of type anyURI.
explanation describes ExplanatoryContext with values of
type GraphPattern.
```

Fig. 4. The SADL snippet of GSN contexts.

In our semantic representation, the discriminator will be whether the context can be expressed formally with concepts defined in a known ontology, or whether the context is simply a reference to something external. Such an external reference could be a URL if the external thing is a digital artefact. Fig. 4 shows how context is captured in SADL.Note that a *ReferentialContext* is described by the *reference* property with values of type *Reference*, and that a *Reference* has a *description* and a *location*, which is a Unique Resource Identifier (URI). Assuming that the *location* is resolvable, the content at that location may provide either unstructured or semi-structured content that must be processed by an extraction service to try to identify semantically meaningful content, or it may provide a formal semantic model (e.g., an OWL ontology). In the case of extraction from unstructured or semi-structured content, human-in-the-loop collaboration may be necessary to ensure correct semantic meaning due to the ambiguity of natural language and/or due to the limitations of extraction technology.

The *ExplanatoryContext*, conversely, must encapsulate meaningful semantic information within the current model. It is expected that this will be in the form of triple patterns and/or functional patterns, two types of *GraphPattern*.

Claims are at the heart of GSN. A GSN argument is a hierarchy of claims with the supporting evidence and the context in which the claims are asserted to be valid. Our model differentiates between goals and claims because a goal is a node in the graph while a claim is a structured proposition. Invoking propositional and predicate logic terminology, the proposition of the claim is something that is claimed to be true while breaking it down into a set of predicates provides the means of calculating its truth.

Provided that a formal model of the system under consideration exists, the internal structure of a claim can be expressed in terms of predicates and/or functions. In predicate logic a predicate takes concepts from the domain as argument inputs and evaluates to true or false. A function takes concepts from the domain as argument inputs and evaluates to a concept in the domain. OWL is a graph language, so predicates are limited to arity-2; they are directed edges in the graph. OWL does not directly support functions, but both Semantic Web rule languages and graph query languages support functions, often called built-in functions.

To represent the internal structure of a claim, we must be able to express graph patterns as part of another structure, much like rules and queries allow graph patterns as elements of the rule or query. And like rules and queries, representing the internal structure of a claim may involve quantification, either explicitly or implicitly as is the case in rules and queries. For the moment, in the SADL representation of triple patterns in a GSN claim, we use the SADL implicit model concept of GraphPattern, with subclasses TriplePattern for predicates and FunctionPattern for functions. We include functions because GSN may have claims that include non-Boolean operators. Fig. 5 Shows the SADL code for GSN claims.

```
Claim (alias "goal statement") (note "a proposition in the
form of a noun-phrase + a verb-phrase") is a class.
  claim describes Goal with a single value of type Claim.
  claimStatement describes Claim with values of type
GraphPattern.
```

Fig. 5. The SADL-AT snippet of GSN claims.

Evidence in GSN is a reference, expressed as a "noun-phrase" according to the standard, to an "evidence item." Evidence is associated with a solution node, which is always a leaf node in the directed graph of a GSN visualization. Fig. 6 shows the SADL encoding of GSN evidences.

```
EvidenceReference (alias "reference to evidence") (note
"a noun-phrase") is a type of Reference.
evidence describes Solution with values of type
EvidenceReference.
```

Fig. 6. The SADL-AT snippet of GSN evidences.

C. The Visualization of SADL-AT

While the content of a formalized assurance case (SADL model) is expressed using the meta-model above, the rendering of the assurance case as a visual GSN diagram is controlled by the portion of the model shown in Fig. 7.

```
NodeShape is a class, must be one of {Rectangle,
Parallelogram, Circle, RoundEndedRectangle,
  OvalWithJ (note "An oval with a 'J' next to it"),
  OvalWithA (note "An oval with an 'A' next to it"),
  RectangleWithDiamondDecorator (note "has diamond
centered beneath"),
  ParallelogramWithDiamondDecorator (note "has diamond
centered beneath")}.
 nodeShape describes GSN_Element with values of type
NodeShape.
 nodeShape of Goal always has value Rectangle.
 nodeShape of Strategy always has value Parallelogram.
 nodeShape of Solution always has value Circle.
 nodeShape of Context always has value
RoundEndedRectangle.
 nodeShape of Justification always has value OvalWithJ.
 nodeShape of Assumption always has value OvalWithA.
 nodeShape of UndevelopedGoal always has value
RectangleWithDiamondDecorator.
 nodeShape of UndevelopedStrategy always has value
ParallelogramWithDiamondDecorator.
```

Fig. 7. The SADL-AT snippet for GSN Visualization.

To the degree that a graphing package is able to render nodes of the shape defined in Fig. 7, the rendered assurance case will conform to the the GSN standard. Graphing packages that are not able to render the finer-grained distinctions illustrated in Fig. 7 can still render the nodes and edges of an assurance case instance as a graph and provide tooltips or other mechanisms to identify the type of each node.

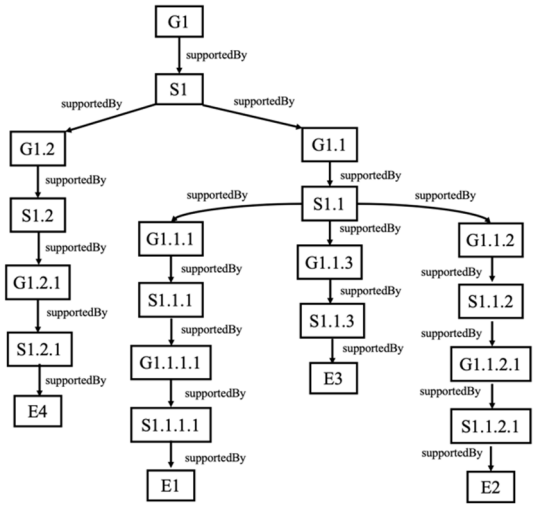

Fig. 8. Goals and strategies in the "supportedBy" hierarchy.

D. Queries of SADL-AT

One of the advantages of SADL-AT is the ability to query assurance cases, which can greatly facilitates the review process for certification authorities. Because the relationships between GSN nodes can only be of two types, as explained above,

querying an assurance case instance is quite straightforward. A goal hierarchy graph can be constructed simply by asking what is "supportedBy" what. Expressed as a SPARQL query to construct a graph, the essence of the query is:

```
construct {?s <supportedBy> ?o} where {?s <supportedBy> ?o}
```

For the delivery drone system described in the next section, this produces a graph of the form shown in Fig 8.

IV. Case Study: Assurance Case Fragments for a Delivery Drone

In this section, we build formalized assurance case fragments for a toy delivery drone system to demonstrate the capabilities of SADL-AT. The assurance case fragments can be queried and rendered into GSN visualizations. More importantly, they are also machine readable and analyzable, which can be used as an input for assurance case analysis tools such as the ones being developed in the DARPA Automated Rapid Certification of Software program. Throughout this case study example, claims, goals and requirements are used inter-changeably.

A. Assurance case based certification for Unmanned Aerial Systems (UAS)

UAS and UAS operations have been certified using guidance provided by various standards, including ASTM F3209, "Standard Practice for Ensuring the Dependability of Software Used in Unmanned Aircraft Systems" (ASTM International, 2016). According to this standard, a structured threat modeling process must be performed by the software manufacturer to identify high-risk areas in the software and suggest mitigations based on the level of severity of the impact. Impacts must be assessed for loss of function (loss of availability) and malfunction (loss of integrity) for the primary functions of the system, but loss of confidentiality will also be assessed in situations where data privacy is a concern. As an effort to comply with the ASTM F3209 standard, we performed an partial evaluation of the resillience of the delivery drone based on delivery missions using VERDICT tool, and constructed assurance case fragments.

B. A Delivery Drone Model

The delivery drone is a part of a delivery mission, and the delivery mission includes an operator, a van, packages to deliver, and one or more delivery drones. When the van arrives at a location which is in proximity to multiple delivery sites, the delivery drones are initialized with current positions, delivery locations, and the packages to deliver. After a delivery drone is launched, it navigates to the delivery site using the inputs from the GPS and IMU (Inertial Measurement Unit). When the drone reaches the delivery site, it captures a picture of the delivery site to ensure that it is free of obstacles and it is safe to drop off the package. Then it will activate the Delivery Item Mechanism to drop off the package.

The system architecture of the delivery drone modeled in AADL is annotated with a set of properties and cyber and safety relations. The set of AADL properties are determined by the security experts and system engineers based on the current implementation of the drone. The cyber and safety relations of a component define how the confidentiality, integrity, and availability of outputs are affected by the confidentiality, integrity and availability of inputs, which will be used to identify attacks and defenses to various components of the system. The source codes for the delivery drone in AADL is available at [21].

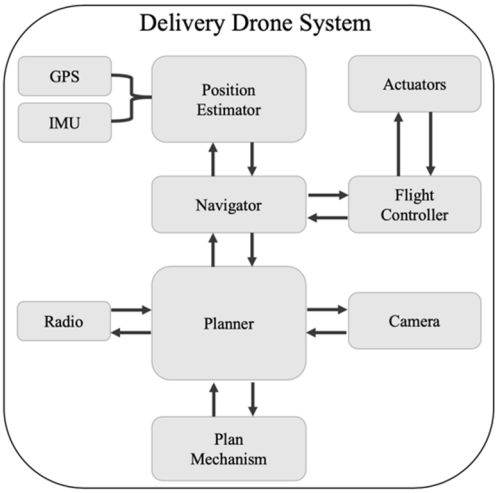

Fig. 9. The delivery drone system architecture overview.

C. Methodologies for developing assurance case fragments

A methodology for the development of assurance arguments for Unmanned Aircraft Systems has been introduced in [19]. In general, there are two approaches for the development of assurance arguments: *top-down approach* and *bottom-up approach*. The combination of the two are often employed in practice. The top-down approach starts with a definition of top-level claims followed by a series of refinements into lower-level details; whereas the bottom-up approach builds an argument by inference from the existing evidence. This paper adopts a top-down approach to develop assurance case fragments. We start from top-level mission claims, refine them to lower-level actionable safety and security sub-claims, and leverage evidence generated from the VERDICT tool to support claims.

D. Formalized top-level claims for the Delivery Drone

In the context of delivery missions for the drone, we are interested in whether the drones are able to deliver packages in a reliable manner. To formalize the assurance cases in GSN, we encode a system model of the delivery drone in SADL. Due to the space limit, a sample of code snippets is included for illustration purposes, (the source code is publicly available at [21] for interested readers). The specification imports a more general library of basic concepts and defines some new classes of systems, subsystems, concerns, and severities specific to the delivery drone. In addition, we extend the semantic model to include the quantitative aspect of the assurance cases. We start to define the top-most requirement for the delivery drone as follows. The formalized version is shown in Fig. 10. The `Drone`, `DeliveryEvent`, and `DeliveryInformation` are

new SADL constructs introduced in the aformentioned SADL system model.

- **MissionReq** – *"The drone shall complete delivering a package to an intended location in a reliable manner".*

```
G1 (note "The drone shall complete delivering a package
to an intended location in a reliable manner") is a
Goal,
with inContextOf (an ExplanatoryContext C1 with
reference (a Drone) with description "A package delivery
drone"),
with inContextOf (an ExplanatoryContext C2 with
reference (a DeliveryEvent)
with description "The event of a package being delivered
to an intended location")
has claim (a Claim with claimStatement
  (a TriplePattern with gpSubject (the Drone),
                       with gpPredicate performs,
                       with gpObject (the DeliveryEvent)),
  with claimStatement
  (a TriplePattern with gpSubject (the DeliveryEvent),
                       with gpPredicate completed,
                       with gpObject true),
  with claimStatement
  (a TriplePattern with gpSubject (the DeliveryEvent),
                    with gpPredicate information,
                    with gpObject (a DeliveryInformation)),
  with claimStatement (a TriplePattern
                  with gpSubject (the DeliveryInformation),
                  with gpPredicate reliability,
                  with gpObject true)).
```

Fig. 10. The SADL-AT snippet for MReq.

The top-most requirement can be naturally treated as the root (the top-most goal) of an assurance case fragment tree. To furthur decompose the top-most mission goal to lower-level requirements, we introduce two additional supporting mission requirements as follows.

- **MReq01** – *"Deliver a package to the intended location".*
- **MReq02** – *"Reliability".*

These can be converted into the following claims for formalization purpose in SADL.

- **MReq01 (SADL)** – *"The drone shall complete delivering a package to an intended location".*
- **MReq02 (SADL)** – *"The delivery information shall be reliable".*

```
G1.1 (note "The drone shall complete delivering a package
to an intended location") is a Goal,
with inContextOf (a ReferentialContext C1),
with inContextOf (a ReferentialContext C2),
has claim (a Claim with claimStatement
  (a TriplePattern with gpSubject (the Drone),
    with gpPredicate performs,
    with gpObject (the DeliveryEvent)),
    with claimStatement
  (a TriplePattern with gpSubject (the DeliveryEvent),
    with gpPredicate completed,
    with gpObject true)).
```

Fig. 11. The SADL-AT snippet for MReq01.

For illustration purposes, we show the formalized SADL requirement **MReq01** in Fig. 11. In the following sections, we will furthur decompose the mission requirements into actionable cyber-security and safety requirements (sub-goals) for security and safety analysis, which will then be substantiated by the evidence generated by the VERDICT tool.

E. Formalized assurance case fragments for the security of the Delivery Drone

To satisfy the top-level mission requirements (claims) related to the cyber-security in Secion *D*, in this case **MReq01**, the drone implementation needs to satisfy certain cyber-security (or security) requirements to guarantee that the system is resilient against cyber-attacks. Normally, we consider three different types of *concerns* for the delivery drone (confidentiality/integrity/availability) with varying levels of *severity* (catastrophic / hazardous / major / minor / none). A detailed description of security concerns and severities can be found in [13]. In this case, we only consider cyber attacks associated with integrity and availability concerns. We develop two mission-critical cyber requirements in plain English for the delivery drone as follows:

- **CyberReq01** – "*The drone shall be resilient to the loss of ability to deliver a package to the appropriate consumer location*".
- **CyberReq02** – *"The drone shall be resilient to maliciously commanded improper delivery of a package".*

```
G1.1.1 (note "The drone's resilience to the loss of
delivery ability shall have integrity concern less severe
than Hazardous") is a Goal,
    with inContextOf (a ReferentialContext C1),
    with inContextOf (a ExplanatoryContext C5 with
reference (a Concern) with description "A concern affecting
the resilience to loss of delivery ability"),
has claim (a Claim with claimStatement
  (a TriplePattern with gpSubject (the Drone),
                       with gpPredicate hasConcern,
                       with gpObject (the Concern)),
  with claimStatement
  (a TriplePattern with gpSubject (the Concern),
                       with gpPredicate ofType,
                       with gpObject resDeliveryAbility),
  with claimStatement
  (a TriplePattern with gpSubject (the Concern),
                       with gpPredicate ofClass,
                       with gpObject Integrity),
  with claimStatement
  (a TriplePattern with gpSubject (the Concern),
                       with gpPredicate ofSeverity,
                       with gpObject (a Severity)),
  with claimStatement
  (a TriplePattern with gpSubject (the Severity),
                       with gpPredicate lessSevere,
                       with gpObject HAZARDOUS)).
```

Fig. 12. The SADL snippet for CyberReq01.

Formally, these security requirements are quantified by their respective concerns and severities. To build assurance case fragments using the Goal Structuring Notation (GSN) in SADL, these requirements need to be expressed in the form of claims that can be formally captured using supporting strategies and sub-claims. Below, we have expressed these statements as sub-claims:

- **CyberReq01 (SADL)** – *"The drone's resilience to the loss of delivery ability shall have integrity concern less severe than hazardous".*

- **CyberReq02 (SADL)** – *"The drone's resilience to malicious commands shall have integrity concern less severe than hazardous".*

The formalization of the claim **CyberReq01** in SADL is shown in Fig. 12. To associate the delivery ability with concrete system ports for security analysis, we introduce yet another layer of sub-claims for CyberReq01 in Fig. 13.

```
G1.1.1.1 (note "The actuation and delivery mechanisms
shall have no integrity or availability concerns") is a
Goal,
with inContextOf (a ReferentialContext C1),
with inContextOf (a ReferentialContext C3 with reference
(an Actuation) with description "The actuation subsystem
of a drone"),
with inContextOf (a ReferentialContext C4 with reference
(a DeliveryItemMechanism) with description "The
deliveryItemMechanism subsystem of a drone"),
has claim (a Claim with claimStatement
  (a TriplePattern with gpSubject (the Drone),
                   with gpPredicate hasActuator,
                   with gpObject (the Actuation)),
  with claimStatement
  (a TriplePattern with gpSubject (the Actuation),
                   with gpPredicate hasIConcern,
                   with gpObject false),
  with claimStatement
  (a TriplePattern with gpSubject (the Actuation),
                   with gpPredicate hasAConcern,
                   with gpObject false),
  with claimStatement
  (a TriplePattern with gpSubject (the Drone),
                   with gpPredicate hasDIMechanism,
        with gpObject (the DeliveryItemMechanism)),
  with claimStatement
  (a TriplePattern with gpSubject (the
DeliveryItemMechanism),
                   with gpPredicate hasIConcern,
                   with gpObject false),
  with claimStatement
  (a TriplePattern with gpSubject (the
DeliveryItemMechanism),
                  with gpPredicate hasAConcern,
                  with gpObject false)).
```

Fig. 13. The SADL snippet for a sub-claim for CyberReq01.

Once the claims and sub-claims have been established, the next step is to chain them together via GSN *strategies* and *evidences*. We introduce evidences to support these claims and some strategies to argue how claims are supported by evidences and sub-claims. For the delivery drone, the evidences will be probabilities of minimum cutsets of the attack-defense trees (for security requirements) generated by the MBAA tool of VERDICT . For security requirements, these probabilities are computed in the context of some well-defined properties developed as a part of the DARPA's Cyber Assured Systems Engineering (CASE) program at GE. A list of strategies and evidence for connecting cyber claims with sub-claims and evidence is defined in Fig. 14. The supporting evidence for goals is generated by attack-defense tree analysis from the Soteria++ tool [17]. The URL of concrete evidence is appended as evidence for interested readers.

F. Formalized assurance case fragments for the safety of the Delivery Drone

Similar to the security requirements, the delivery drone also needs to satisfy some mission-critical safety requirements which ensure that the performance of the drone is within certain acceptable error bounds. In this case, the top-level goal related to safety is **MReq02**. Two safety requirements are determined by our security expert and only SafetyReq02 is needed to support **MReq02**. Safety requirement SafetyReq01 is determined to support mission requirement **MReq01**. We used the appropriate concern and severity quantifications to express these safety requirements as the following claims:

- **SafetyReq01 (SADL)** – *"The loss of availability of actuation should be less than 1e-7 pfh".*
- **SafetyReq02 (SADL)** – *"The loss of integrity of delivery information shall be less than 1e-7 pfh".*

```
S1 (note "Argument: By conjunction of G1.1 and G1.2, the
delivery event will be completed in a reliable manner")
is a Strategy.
S1.1 (note "Argument: By conjunction of G1.1.1, G1.1.2,
and G1.1.3, the actuator and delivery mechanism will
perform their respective functions within acceptable
error bounds") is a Strategy.
S1.1.1 (note "Argument: By G1.1.1.1, the actuation and
delivery mechanisms have no integrity or availability
concerns") is a Strategy, with inContextOf (an
ExplanatoryContext C6 with reference (an
InformationObject) with description "The resilience to
the loss of delivery ability has a hazardous integrity
concern if the actuation or delivery mechanism have an
integrity or availability concern.").
S1.1.1.1 (note "Argument: The probability that actuation
and delivery mechanism will have no concerns is confirmed
by attack/defense tree analysis performed by Soteria++")
is a Strategy, with inContextOf (a ReferentialContext C10
with location "https://github.com/ge-high-
assurance/assurance-
case/tree/master/DeliveryDrone/context/CASE_Consolidated_
Properties.aadl" with description "CASE Consolidated
properties.").
G1 supportedBy S1.
S1 supportedBy G1.1.
G1.1 supportedBy S1.1.
S1.1 supportedBy G1.1.1.
G1.1.1 supportedBy S1.1.1.
S1.1.1 supportedBy S1.1.1.1.
S1.1.1.1 supportedBy E1.
E1 (note "Soteria++ output of probability of the minimum
cutsets") is a Solution, with evidence (an
EvidenceReference with location "https://github.com/ge-
high-assurance/assurance-
case/tree/master/DeliveryDrone/evidence/ImplProperties.xm
l").
```

Fig. 14. Strategies and evidence for establishing cyber claims

Fig. 15 and 16 show the SADL GSN formalizations of the mission requirement **MReq02** and safety requirement **SafetyReq02** respectively using the same system model of the delivery drone as used for the security requirements.

```
G1.2 (note "The delivery information shall be reliable ")
is a Goal,
with inContextOf (a ReferentialContext C1),
with inContextOf (a ReferentialContext C2),
has claim (a Claim with claimStatement
  (a TriplePattern with gpSubject (the Drone),
    with gpPredicate performs,
    with gpObject (the DeliveryEvent)),
  with claimStatement
  (a TriplePattern with gpSubject (the DeliveryEvent),
    with gpPredicate information,
    with gpObject (a DeliveryInformation)),
    with claimStatement
  (a TriplePattern with gpSubject (the
DeliveryInformation),
    with gpPredicate reliability,
    with gpObject true)).
```

Fig. 15. The SADL-AT snippet for MReq02.

For safety requirements, hazards are identified in the context of events, such as "loss of availability" and "undetected erroneous data", specified in the VERDICT annex of the AADL model. These contexts are therefore specified for the corresponding strategies as seen in Fig. 15.

```
G1.2.1 (note "The loss of integrity of delivery
information shall be less than 1e-07 pfh") is a Goal,
with inContextOf (a ReferentialContext C1),
with inContextOf (a ReferentialContext C4),
has claim (a Claim with claimStatement
  (a TriplePattern with gpSubject (the Drone),
    with gpPredicate hasDIMechanism,
    with gpObject (the DeliveryItemMechanism)),
  with claimStatement
  (a TriplePattern with gpSubject (the
                                DeliveryItemMechanism),
    with gpPredicate lossOfIntegrity,
    with gpObject (a Probability)),
  with claimStatement
  (a TriplePattern with gpSubject (the Probability),
    with gpPredicate lessthan,
    with gpObject 1e-07)).
```

Fig. 16. The SADL snippet for the safety requirement **SafetyReq02**.

The strategy to substantiate goal **G1.2.1** is fault tree analysis by Soteria++. The evidence is the likelihoods computed for all the minimal cut sets. In this case, they are all less than the target likelihood specified in the safety requirement (goal), thus can be directly leveraged to support the safety goal.

```
G1 supportedBy S1 .
S1.2 (note "Argument: By G1.2.1, the delivery mechanism
will perform within acceptable error bounds") is a
Strategy.
G1.2 supportedBy S1.2 .
S1.2 supportedBy G1.2.1 .
E4 (note "Soteria++ output of probability of the minimum
cutset") is a Solution, with evidence (an
EvidenceReference with location "https://github.com/ge-
high-assurance/assurance-
case/tree/master/DeliveryDrone/evidence/ImplProperties-
safety.xml").
S1.2.1 supportedBy E4 .
```

Fig. 17. Strategy and evidence for the safety requirement **SafetyReq02**

The strategy and evidence for the safety requirement **SafetyReq02** is shown in Fig. 17.

G. *Queries and visulization for the assurance case fragments of the delivery drone*

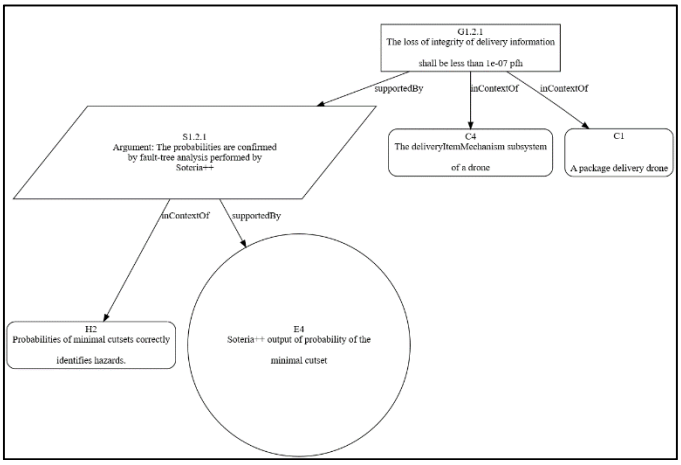

Fig 18. The portion of the GSN with the top node **SafetyReq02**.

The SADL-AT tool allows the visualization of assurance case fragments in GSN by using the GraphViz plugin [14]. Fig. 18 shows a portion of the delivery drone GSN that has **SafetyReq02** as the root node. SADL-AT also allows user to query the GSN to obtain textual results. For example, the simple query given below outputs all possible branches of length 6 for the delivery drone GSN which start at the rootgoal..

```
Ask: "select ?gsn  ?sub (count(?inter)
as ?pathLength)
Where {?gsn <rootGoal> <G1>
. <G1> <supportedBy>* ?inter
. ?inter <supportedBy>+ ?sub
} group by ?gsn ?sub having (?pathLength =
6)".
```

OUTPUT:

```
"gsn","st","subg","st2","subg2","st3","subg3"
"GSNDeliveryDrone","S1","G1.2","S1.2","G1.2.1
","S1.2.1","E4"
"GSNDeliveryDrone","S1","G1.1","S1.1","G1.1.3
","S1.1.3","E3"
"GSNDeliveryDrone","S1","G1.1","S1.1","G1.1.2
","S1.1.2","G1.1.2.1"
"GSNDeliveryDrone","S1","G1.1","S1.1","G1.1.1
","S1.1.1","G1.1.1.1"
```

V. Future Work

Assurance cases in the form of GSN allow system engineers to provide a tangible representation of the confidence in a system's ability to perform as intended. However, developing an assurance case for a non-trivial task such as the delivery drone requires one to have a deep understanding of the SADL language and the VERDICT tool in which the requirements are specified. This calls for the need of creating

assurance case templates that can be used by developers to quickly and easily design assurance case fragments. We plan to develop customized assurance case templates that can be used to design comprehensive assurance cases to represent security and safety properties of critical systems. These templates can be used to create assurance case fragments that closely follow the architectural and behavioral specifications used by the VERDICT tool and the outputs generated by it. This would confine the scope of the templates, but make them highly declarative in nature, allowing them to be easily used by domain experts. To create these templates, we will need to formalize the basic concepts necessary for expressing the specification, behavior, and requirements in a manner that they remain flexible over any critical system, but at the same time, be easy to understand and apply [15].

VI. Conclusion

In this paper, we introduce a toolkit – SADL-AT including a semantic model for modeling GSN-compliant assurance case fragments. The semantic model is built on top of SADL, which is a flexible and extensible language. The toolkit enables users to formalize assurance cases that are machine-analyzable, visualizable and queriable. We demonstrate the capabilities of the toolkit through the construction of security and safety assurance case fragments for a delivery drone.

Acknowledgment

The authors would like to thank the Computational Logic Center at the University of Iowa for providing the delivery drone model template.

This research was developed with funding partially from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

DISTRIBUTION A. Approved for public release: Distribution Unlimited. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) and Space and Naval Warfare Systems Center, Pacific (SSC Pacific) under Contract No. N66001-18-C-4006.

References

[1] "Semantic Application Design Language (SADL)," [Online]. Available: http://sadl.sourceforge.net/index.html. [Accessed 3/5/2019].

[2] T. Kelly and R. Weaver, "The goal structuring notation–a safety argument notation," Proceedings of the dependable systems and networks 2004 Workshop on Assurance Cases, 2004.

[3] Mitre, "Common Attack Pattern Enumeration and Classification. https://capec.mitre.org/".

[4] "VERDICT: An OSATE plugin for architectural and behavioral analysis of AADL models," GE, 2020. [Online]. Available: https://github.com/ge-high-assurance/VERDICT. [Accessed 10/3/2020].

[5] John Rushby SRI International, "The Interpretation and Evaluation of Assurance Cases," Computer Science Laboratory, Menlo Park, CA, 2015.

[6] Bloomfield, Robin, and Kateryna Netkachova. "Building blocks for assurance cases." In *2014 IEEE International Symposium on Software Reliability Engineering Workshops*, pp. 186-191. IEEE, 2014.

[7] Hawkins, Richard, Ibrahim Habli, Dimitris Kolovos, Richard Paige, and Tim Kelly. "Weaving an assurance case from design: a model-based approach." In *2015 IEEE 16th International Symposium on High Assurance Systems Engineering*, pp. 110-117. IEEE, 2015.

[8] E. Denney, G. Pai and J. Pohl, "AdvoCATE: An assurance case automation toolset," in *International Conference on Computer Safety, Reliability, and Security*, 2012.

[9] A. Gacek, B. J. D. Cofer, K. Slind and M. Whalen, "Resolute: an assurance case language for architecture models," *ACM SIGAda Ada Letters,* vol. 34, no. 3, pp. 19-28, 2014.

[10] "OWL: Web Ontology Language," [Online]. Available: https://www.w3.org/OWL/. [Accessed 8 2 2019].

[11] C. McMillan, A. Crapo, M. Durling, M. Li, A. Moitra, P. Manolios, M. Stephens and D. Russell, "Increasing development assurance for system and software development with validation and verification using ASSERT™," in *SAE Technical Paper*, No. 2019-01-1370, 2019.

[12] "GSN Community Standard, Version 2," January 2018. [Online]. Available: https://scsc.uk/r141B:1?t=1.

[13] U.S. Department of Transportation Federal Aviation Administration, "Advisory Circular AC 25.1309-1A (Subject: System Design and Analysis)," 1988.

[14] Ellson, John, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Gordon Woodhull. "Graphviz and dynagraph—static and dynamic graph drawing tools." In *Graph drawing software*, pp. 127-148. Springer, Berlin, Heidelberg, 2004.

[15] Hawkins, Richard, Kester Clegg, Rob Alexander, and Tim Kelly. "Using a software safety argument pattern catalogue: Two case studies." In *International Conference on Computer Safety, Reliability, and Security*, pp. 185-198. Springer, Berlin, Heidelberg, 2011.

[16] Kit Siu, Abha Moitra, Meng Li, Michael Durling, Heber Herencia-Zapana, John Interrante, Baoluo Meng; Cesare Tinelli, Omar Chowdhury, Daniel Larraz, Moosa Yahyazadeh, M. Fareed Arif, and Daniel Prince, "Architectural and Behavioral Analysis for Cyber Security," in IEEE DASC, San Diego, 2019.

[17] Kit Siu, Heber Herencia-Zapana, Daniel Prince, Abha Moitra , "A Model Based Framework for Analyzing the Security of System Architectures," in Reliability and Maintainability Symposium (RAMS 2020), Palm Springs, California, 2020.

[18] National Institute of Standards and Technology (NIST), "Security and Privacy Controls for Federal Information Systems and Organizations," 2013.

[19] Denney, Ewen & Pai, Ganesh, "A Methodology for the Development of Assurance Arguments for Unmanned Aircraft Systems," in International System Safety Training Symposium, 2015.

[20] M. Li et al., "Requirements-based Automated Test Generation for Safety Critical Software," 2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC), San Diego, CA, USA, 2019

[21] GSN source code for delivery drone: https://github.com/ge-high-assurance/assurance-case/tree/master/DeliveryDrone/DeliveryDroneGSN

[22] AADL source code for delivery drone : https://github.com/ge-high-assurance/assurance-case/tree/master/DeliveryDrone/DeliveryDrone_AADL

[23] Yu, Han, et al. "System and method for test generation from software specification models that contain nonlinear arithmetic constraints over real number ranges." U.S. Patent No. 10,169,217. 1 Jan. 2019.