The Role of Formalization and Argumentation in Assurance Cases

Ewen Denney and Ganesh Pai
SGT / NASA Ames Research Center

{ewan.denney, ganesh.pai}@nasa.gov
Safety Risk Management & Assurance (SRM&A)

System Development and Verification

- Hardware Development and Verification
  - Safety requirements (Function, system, hardware, software)
  - Risk elimination / control mechanisms, ...

- Software Development and Verification
  - Requirements (Function, system, hardware, software)
  - Design Artifacts, e.g., models
  - Executable e.g., code
  - Verification artifacts, ...

Domain / Concept Modeling and Analysis

- Concept of operations
- Stakeholder needs
- Regulatory requirements
- Domain model, e.g., ontology
- ...

Hazard Identification

- Hazards, failure modes, ...

Risk Analysis and Assessment

- Likelihood, Severity
- Risk levels, classification

Safety Analysis

Risk Control

- Safety requirements, goals
- Hazards, failure modes, ...
- Risk levels
- Risk control strategies, ...

Safety Case (Argument) Development

- Safety arguments
Notions of Assurance Case in Aviation

- UK MoD Defence Standard 00-56, Issue 4, June 2007
  - “... Safety case shall consist of a structured argument, supported by a body of evidence, that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given environment”.

- Civil Aviation / UAS operations in civil airspace
  - Preference for using normative regulations
  - Performance-based standards
  - “Safety cases” for one-off systems,
    - i.e., Concepts that are built once and fielded
    - e.g., RVSM implementation over some airspace sector
  - Notion of safety case is compatible but seems to be different
Notions of Assurance Case in Aviation

  – “Safety case is the document assurance (i.e., argument and supporting evidence) of the achievement and maintenance of safety”

• ICAO Guidance Material for Building a Safety Case for ADS-B separation service, May 2011
  – “A safety case is a document which provides substantial evidence that the system to which it pertains meets its safety objectives”
  – “… An explicit documentation of a safety-critical system, its corresponding safety objectives, and the associated safety risk assessment and risk management of the system, at appropriate milestones in the life of the system”.

Notions of Assurance Case in Aviation

• FAA
  – Order 8900.1 Flight Standards Information Management System, Vol. 16, UAS, Ch. 7, SRM, Safety Case Template
  • “Core” content
    – Environment (airspace system) description
    – System description and system change description
    – Airworthiness description of affected items
    – Aircraft capabilities and flight data
    – Accident / incident data
    – Hazard analysis and details of risk analysis, risk assessment, and risk control
    – Emergency and contingency procedures
    – Pilot / crew roles and responsibilities
  – Safety Risk Management Plan
    • Hazard tracking

• No expectation of an explicit, or structured, argument containing claims, argument, evidence, etc.
Notions of Assurance Case in Aviation

- CAA – Congested Areas Operating Safety Case (CAOSC) IN-2014/184
  - “For SUAS (small UAS) and SUAS applications, it is not expected that complex hazard identification and risk assessment techniques will be used (e.g., Goal Structured Notation) ...”

- Safety Case Template
  - Core content: System, Operations, and Hazard and Risk Assessment
  - Additionally, a “Self assessment”
    - Textual Claims, Arguments and Evidence
    - “There is no mandatory requirement to use complex techniques (e.g. Goal Structured Notation).”
Our Position

• Arguments are useful
  – To organize safety information, also to organize airworthiness claims and evidence
    • “Internal” complexity management and “confidence” on having done due diligence
  – Need not always be shown to / seen by regulator
    • Queries, views
  – Hide arguments à la hiding formalism in requirements using structured natural language
    • Report generation

• For UAS
  – Operations may continue to require safety cases
    • Only if they represent unique concepts needing one-off safety assessments
  – Airworthiness will follow traditional process as regulations get formulated
    • Likely to be a combination of performance based and normative
  – Not all assurance will require assurance cases
    • Structures, Physical modeling, ...
Instantiated Methodology for SRM&A
The Role of Formalization

Two distinct notions of formalization

• Formal languages
  – Natural language
  – Controlled natural language
  – Formal assurance language

• Formal structures
  – Formalize the “scaffolding” to support automation
  – Support range of languages
  – Support range of reasoning structures
The Role of Automation

• Maintaining consistency and supporting evolution
  – Systems and safety cases evolve
  – Keep consistent during development / in operation

• Structuring large arguments
  – Modularization
  – Hierarchisation

• Aiding stakeholder comprehension
  – Diverse stakeholders care about different things

• Supporting analysis and review
  – Assess progress, coverage, confidence

• Supporting reuse
  – Extract reusable safety artifacts
Argument Structures and Safety Cases

**Argument Structures**
e.g., in GSN
with well-formedness constraints

**External Documents**
e.g., hazard logs, requirements, etc.

**Models / Artifacts of the System**
e.g., in MATLAB / Simulink, etc.

**Ontologies**
e.g., in OWL
- System organization
- Regulations
- Environment / Domain, etc.

All of this constitutes the safety case
Lightweight Semantics

- Modeling domain knowledge
  - Ontologies provide additional semantics to argument structures
  - Capture as metadata associated with argument structure nodes
  - Attribute syntax

```plaintext
attribute ::= attributeName param*
param ::= String | Int | Nat | nodeId | sameNodeTypeID | goalNodeID | strategyNodeID |
     evidenceNodeID | assumptionNodeID | contextNodeID | justificationNodeID |
     contextNodeID | userDefinedEnum
```

- userDefinedEnum
  - severity ::= catastrophic | hazardous | major | minor | noSafetyEffect
  - likelihood ::= frequent | probable | remote | extremelyRemote |
  - extremelyImprobable

- Examples
  - Attribute: risk(severity, likelihood), formalizes(sameNodeTypeID)
  - Attribute instance: risk(severity(catastrophic), likelihood(remote))
  - Parameter type synonyms: requirement == string
Example

requirement(id, hierarchyLevel, assuranceConcern)
formalClaim(id), informalClaim(id), hazard(id)
  id ::= int | string
  hierarchyLevel ::= highLevel | lowLevel
  assuranceConcern ::= functional | safety | reliability | availability | maintenance
requirementAppliesTo(elementLevel, elementType, element)
  elementLevel ::= system | subsystem | component | module | function | model | signal
  elementType ::= hardware | software
  element ::= aileron | elevator | flaps | propulsionBattery | avionicsBattery | actuatorBattery
             | avionics | autopilot | FMS | AP | aileronPIDController | elevatorPIDController
             | propulsion | engine | propeller | engineMotorController | actuator |
             | flightComputer | wing | actuatorMotorController pilotReceiver | IMU |
references(variable)
  variable ::= aileronValue | pitchAttitude | flareAltitude | vRef | vNE | thrust | vS1
regulation(part)
  part ::= 14CFR23.73 | 14CFR23.75
risk(severity, likelihood)
  severity ::= catastrophic | hazardous | major | minor | noSafetyEffect
  likelihood ::= frequent | probable | remote | extremelyRemote | extremelyImprobable
isFormalizedBy(sameNodeTypeID)
Consistency and Evolution

Artifacts
Requirements, Hazard Logs, Design documents, Test / verification records, ...

Argument Fragments

Pattern Library

Bidirectional Mapping

• Automation in
  - Argument generation
  - Change update & impact analysis
  - Task generation
  - Confidence
  - ...

Pattern Library
# Tabular Requirements Specifications

## Hazards Table

<table>
<thead>
<tr>
<th>ID</th>
<th>Hazard</th>
<th>Cause / Mode</th>
<th>Mitigation</th>
<th>Safety Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>HR.1.3</td>
<td>Propulsion system hazards</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HR.1.3.1</td>
<td>Motor overheating</td>
<td>Insufficient airflow</td>
<td>Monitoring</td>
<td>RF.1.1.4.1.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Failure during operation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HR.1.3.7</td>
<td>Incorrect programming of KD motor controller</td>
<td>Improper procedures to check programming before flight</td>
<td>Checklist</td>
<td>RF.1.1.4.1.9</td>
</tr>
</tbody>
</table>

## System Requirements Table

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement</th>
<th>Source</th>
<th>Allocation</th>
<th>Verification Method</th>
<th>Verification Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS.1.4.3</td>
<td>Critical systems must be redundant</td>
<td>AFSRB</td>
<td>RF.1.1.1.1.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RS.1.4.3.1</td>
<td>The system shall provide independent and redundant channels to the pilot</td>
<td>AFSRB</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Functional Requirements Table

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement</th>
<th>Source</th>
<th>Allocation</th>
<th>Verification Method</th>
<th>Verification Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF.1.1.1.1.3</td>
<td>FCS must be dually redundant</td>
<td>RS.1.4.3</td>
<td>FCS</td>
<td>Visual Inspection</td>
<td>FCS-CDR-20110701, TR20110826</td>
</tr>
<tr>
<td>RF.1.1.4.1.2</td>
<td>CPU/autopilot system must be able to monitor engine and motor controller temperature.</td>
<td>HR.1.3.1</td>
<td>Engine systems</td>
<td>Checklist</td>
<td>Pre-flight checklist</td>
</tr>
<tr>
<td>RF.1.1.4.1.9</td>
<td>Engine software will be checked during pre-deployment checkout</td>
<td>HR.1.3.7</td>
<td>Pre-deployment checklist</td>
<td>Checklist</td>
<td>Pre-deployment checklist</td>
</tr>
</tbody>
</table>
Mapping Multiple Tables

From hazards table

Linking tables using common content

From functional requirements table
Mapping Modifications

Hazards Table

<table>
<thead>
<tr>
<th>ID</th>
<th>Hazard</th>
<th>Cause / Mode</th>
<th>Mitigation</th>
<th>Safety Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>HR.1.3</td>
<td>Propulsion system hazards</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HR.1.3.1</td>
<td>Motor overheating</td>
<td>Insufficient airflow</td>
<td>Monitoring</td>
<td>RF.1.1.4.1.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Failure during operation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HR.1.3.7</td>
<td>Incorrect programming of KD motor controller</td>
<td>Improper procedures to check programming before flight</td>
<td>Checklist</td>
<td>RF.1.1.4.1.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Evidence linking (Strategy definition)

Claims definition
Comprehension: Motivating Queries and Views

• Real argument structures / safety cases are large
  – EUROCONTROL Airport surface surveillance with ADS-B preliminary safety case is 200 pages!

• Safety cases contain diverse information and heterogeneous reasoning
  – Results of various analyses, inspections, audits, reviews, simulations, other verification activities, etc.
  – Evidence of safe prior operations, if available / applicable

• Safety cases evolve
  – Assumptions validated / invalidated
  – Counterevidence, additional corroborative evidence, new evidence

• Need to improve comprehension, change management, assessment
  – Present role-specific information to stakeholder(s)
    • e.g., show traceability of different kinds to regulator
  – Updates safety case to be consistent with reality
  – Change safety case during as it evolves
  – Need to locate specific information for all of the above
Arguments, Queries, and Views

- **Query**
  - A pre-query $Q$, of *arity* 1, according to well-formedness rules

  \[
  \text{applied to}
  \]

- **Argument structure / diagram**
  - Diagram in GSN showing the structure and elements of an argument

  \[
  \text{produces}
  \]

- **View: Sub-argument derived from query**
  - Represented as a *View diagram*
    - Shows argument structure that satisfies the query
    - Hides all nodes that do not satisfy the query
    - Abstracted into *concealment* nodes (C-nodes)
Example Argument for Querying

Unanticipated UA nose pitch down during descent and landing hazard mitigation

Metadata
Regulatory requirements
System Organization
Requirement types, and relations

Arguments over safety requirements
Arguments over functional breakdown
Arguments over physical architecture

Diverse evidence
- Reviews
- Inspections
- System Testing
- …
AQL Queries and Views: Example

• Natural language query
  – Which parts of the argument structure address the FARs 14 CFR Parts 23.73 and 23.75?

• Interpretation
  – Those fragments of the argument structure whose root goals contain claims related to the regulatory requirements 14 CFR 23.73, 23.75.

• Formulating an AQL query
  – Goal(s) where attributes (or description) have references to the regulations, or
  – Complete sub-trees with the goals above as root(s)
AQL Queries and Views: Example

AQL

(type has goal) and (attributes has (regulation (14CFR23.73) or attributes has regulation(14CFR23.75)) or

E (isSolvedBy+)((attributes has (regulation (14CFR23.73) or attributes has regulation(14CFR23.75)))

Resulting View
Structuring: Motivating Hierarchy

• Safety cases aggregate heterogeneous reasoning and evidence
  – Safety / System / Subsystem / Component / Software Analysis
  – Requirements, Design information, Models, Code
  – Verification, Inspections, Reviews, Simulations
  – Data and records from prior/ongoing operations, maintenance, ...

• Aggregation of large amounts of information
  – Preliminary safety case ~ 200 pages
  – Slice of safety argument ~ 500+ nodes

• Structures that are inherently hierarchical
  – Requirements decomposition
  – Formal property decomposition
  – Physical / structural breakdown

• Represent argument at multiple levels of abstraction
  – Refine abstract to concrete, retaining trace between levels

• Modules vs hierarchy
  – Horizontal vs vertical decomposition
Abstraction Types

• Hierarchical node types
  – Hierarchical Goal: abstract well-developed argument fragments, hiding intermediate decomposition steps
    • e.g., Refinement and formalization of a requirement
  – Hierarchical Strategy: aggregate meaningful chain of strategies (plus supplemental reasoning)
    • e.g., Decomposition over system breakdown, followed by decomposition over operating phases
  – Hierarchical Evidence: fully developed argument chain (hierarchical strategy with no outgoing goals)
    • e.g., Formal decomposition of a requirement ending in proof
Example
MIZOPEX Ground-based Sense and Avoid (GBSAA)

• Performing Earth Science measurements in the Arctic Ice
  – Off the coast of Alaska (Oliktok Point)
  – Satellite-based solution was too expensive
  – Use airborne instruments on UAS
    • Two classes of small UAS
    • NASA SIERRA; University of Alaska’s Boeing Insitu ScanEagle
  – Too dangerous for visual observers
    • So use ground-based air defense RADAR for “sense-and-avoid”

• Considered an alternative means of compliance (AMOC) by the FAA
  – Hard requirement to submit a safety case for approval of operations by means of a Certificate of Authorization (COA)
  – Use N 8900.207, FAA National Policy Document on UAS operational approval guidance (now replaced by N 8900.227)
  – Our role
    • Create an operational safety case for this AMOC
MIZOPEX GBSAA Concept

Air Defense RADAR for monitoring and airspace deconfliction

- Threat Volumes
- Corridor of operations
- Due regard airspace
- RADAR Surveillance Volume
- Boundary of US NAS
MIZOPEX GBSAA Operational Safety Case

- Accepted by the FAA, COAs granted
  - Primarily a report
  - Explicit argumentation not required to be communicated by the regulator
  - However, we are preparing safety arguments
  - First known example of GBSAA use for civilian UAS operations in the NAS
  - First known accepted safety case for civilian UAS operations in the NAS
  - Explicitly required hazard tracking and monitoring to validate assumptions and safety case
Example

Flat Safety Argument
- Fragment of larger argument for Ground-based Detect and Avoid (GBDAAA)
Example
A. Hierarchical Strategy (Open)

- Representing a chain of strategies
- “Operator directed avoidance” followed by “Categories of avoidance procedures”
B. Hierarchical Evidence (Open)

- Representing procedures for avoidance based on aircraft location
Tool Support
**AdvoCATE: Assurance Case Automation Toolset**

- **Functionality**
  - Report generation
  - Generation of to-do lists
  - Generation of traceability matrices
  - Computation of metrics
  - Queries, views
  - Verification

- **Structuring**
  - Patterns
  - Modules
  - Hierarchy

- **Integration/generation**
  - Requirements tables
  - Formal methods

---

**Vision**

Safety information, assurance and risk management (SMART) Dashboard
Concepts
## Concepts: Syntax and Semantics

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proof</td>
<td>Inference Tree</td>
</tr>
</tbody>
</table>

**Diagram:**

- **Syntax:**
  - Proof

- **Semantics:**
  - Inference Tree
Concepts: Syntax and Semantics

Syntax
- Proof
- Argument Structure

Semantics
- Labeled DAG
- Inference Tree
Concepts: Syntax and Semantics

Syntax

- Argument Structure
- Proof

Semantics

- Labeled Directed Hypergraph
- Labeled DAG
- Inference Tree
- Argument Pattern
Concepts: Syntax and Semantics

**Syntax**
- Argument Pattern
- Argument Structure
- Hierarchical Proof (Hiproof)
- Proof

**Semantics**
- Ordered DAG
- Inference Tree
- Labeled Directed Hypergraph
- Labeled DAG
Concepts: Syntax and Semantics

Syntax

Hierarchical Argument Structure (Hicase)
Hierarchical Proof (Hiproof)

Semantics

Ordered Labeled DAG
Labeled Directed Hypergraph

Argument Pattern
Ordered DAG
Inference Tree

Order DAG
Concepts: Syntax and Semantics

**Syntax**
- Hierarchical Pattern
  - Hierarchical Argument Structure (Hicase)
  - Hierarchical Proof (Hiproof)
  - Argument Pattern
  - Argument Structure
  - Proof

**Semantics**
- Ordered Labeled Directed Hypergraph
  - Ordered Labeled DAG
  - Labeled Directed Hypergraph
  - Labeled DAG
  - Inference Tree
Concepts: Syntax and Semantics

Syntax
- Hierarchical Pattern
- Hierarchical Argument Structure (Hicase)
- Hierarchical Proof (Hiproof)
- Argument Pattern
- Argument Structure
- Proof

Semantics
- Ordered Labeled Directed Hypergraph
- Ordered Labeled DAG
- Labeled Directed Hypergraph
- Ordered DAG
- Inference Tree
- Tree of (Ordered) Labeled Directed (Acyclic) (Hyper)graphs

Modules
Conclusions

• An argument is a means to an end

• Automation: Why?
  – Consistency and evolution
  – Comprehension, analysis, and review
  – Reuse

• Automation: How?
  – Pattern instantiation and transformation
  – Querying, views, metrics, verification
  – Confidence

• Rigorous basis
  – Family of reasoning structures: arguments + metadata
  – Spectrum of language formality: natural $\rightarrow$ lightweight $\rightarrow$ formal
  – Ongoing work on integrating confidence quantification
  – Formal basis for dynamic safety cases

• Raising the level of abstraction of arguments
  – cf. Model-based development
  – Implemented in AdvoCATE
  – Need to qualify argument generation tool
Questions

• When are arguments appropriate, and when performance standards?
• When is formalism appropriate?
• What is appropriate level of abstraction? Can we assign automatically?
• What is basis for round-trip engineering?
• What is relation between language structure and reasoning structure?
• What is high-level domain-specific query language?
• How to combine hierarchy and patterns?
• What are views for modules, hierarchy?
Please consider attending


Collocated with SAFECOMP 2015

http://ti.arc.nasa.gov/events/assure2015/