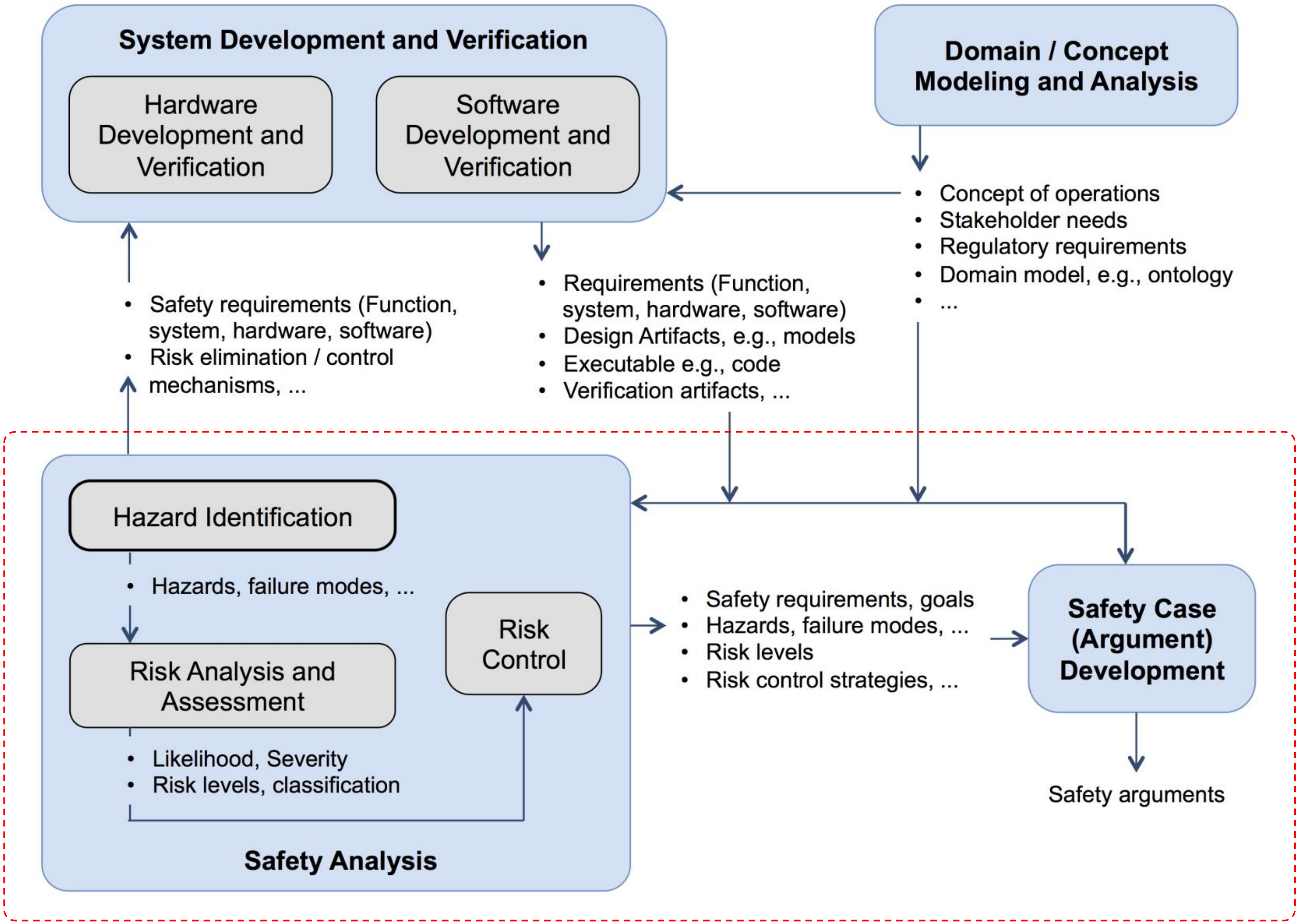
Automated Assurance Cases: Why and How?

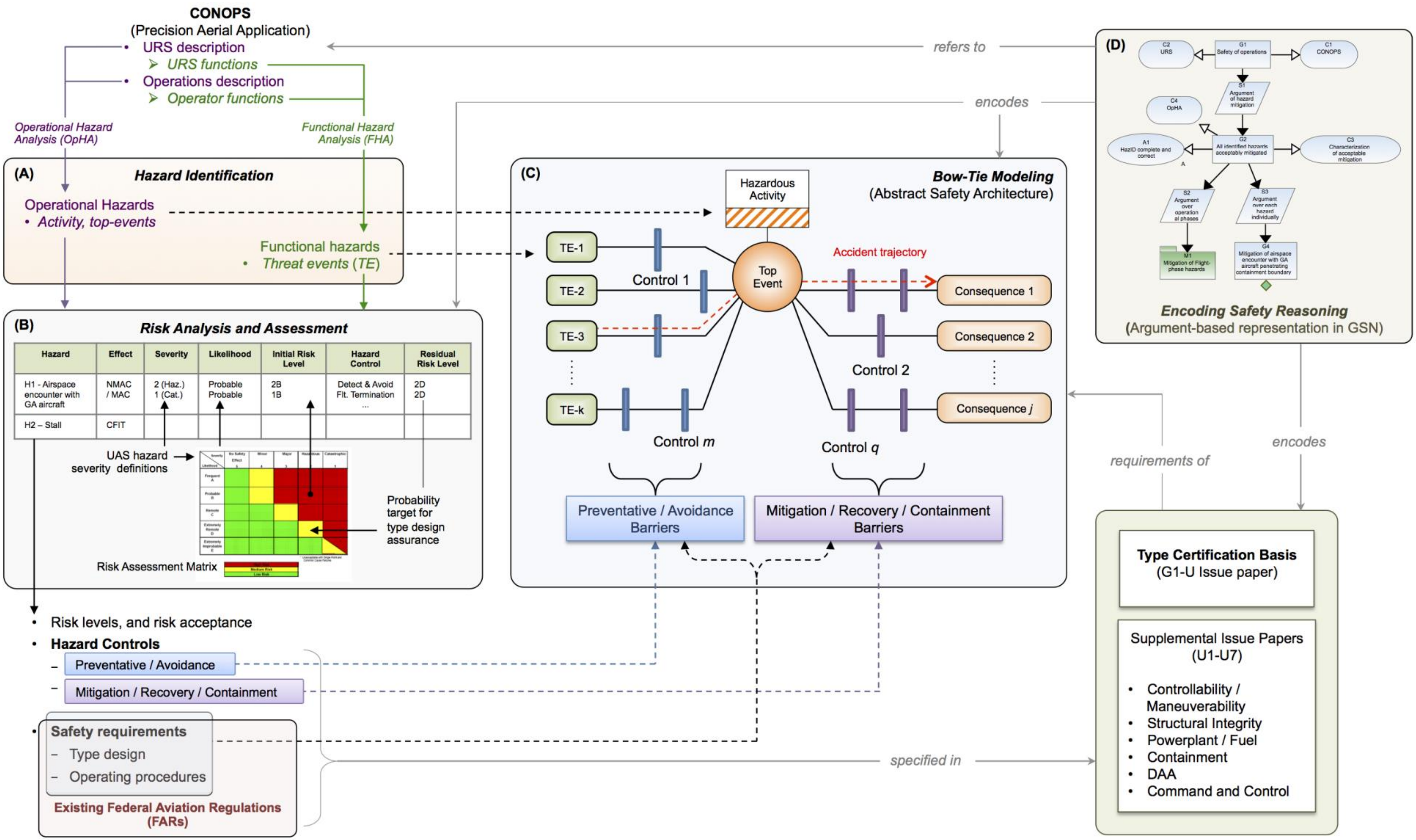
Ewen Denney and Ganesh Pai
SGT / NASA Ames Research Center

ewen.denney@nasa.gov

Safety Risk Management & Assurance



Instantiated Methodology for SRM&A



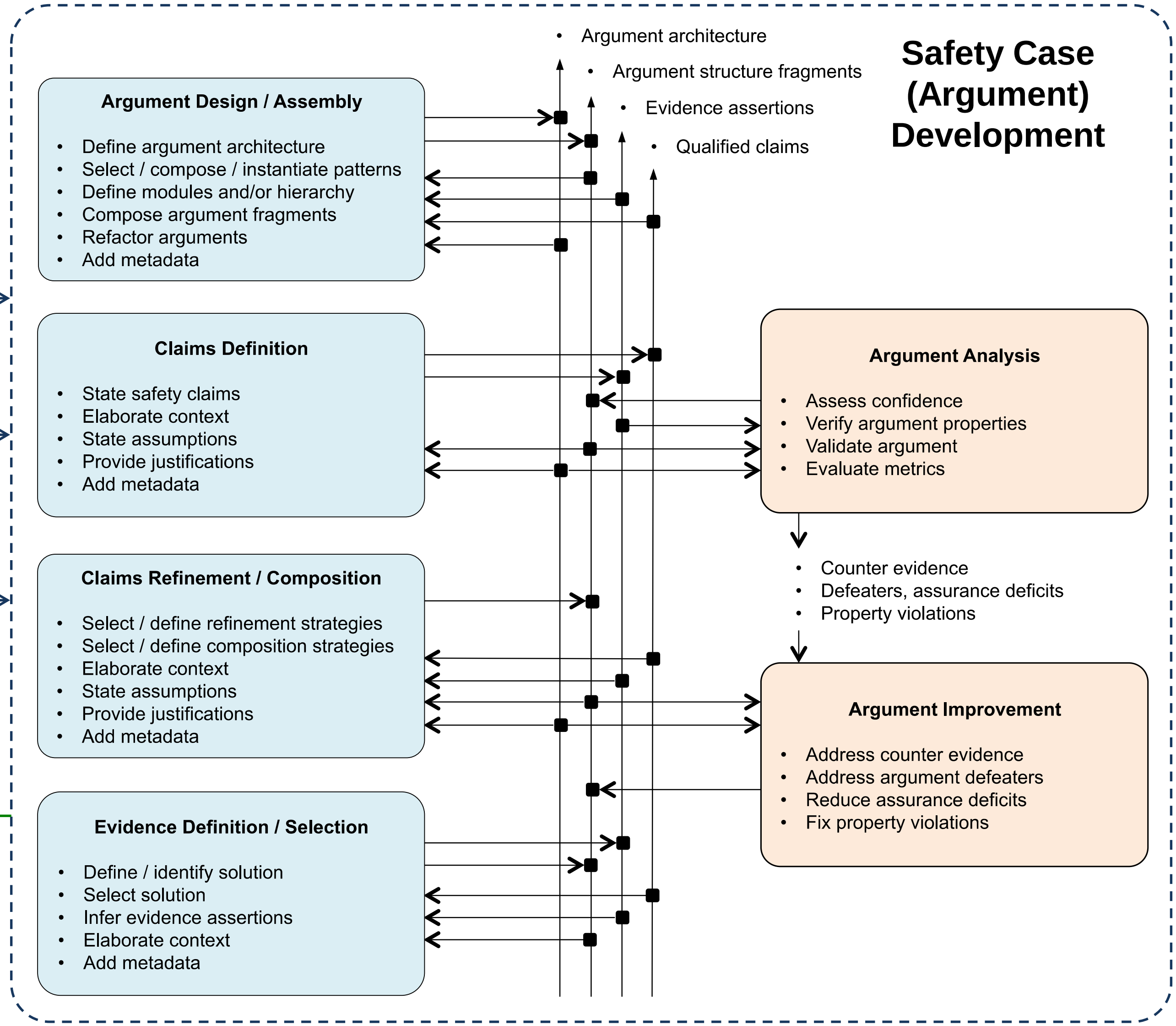
Argument Development

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

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

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

Safety arguments
- Preliminary
- Intermediate / architectural
- Implementation
- Operational



Motivating Automation

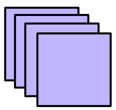
- Maintaining consistency and supporting evolution
  - Systems and safety cases evolve
  - Keep consistent during development / in operation
- Structuring large arguments
  - Modularization
  - Hierarchisation
- Aiding stakeholder comprehension
  - Diverse stakeholders care about different things
- Supporting analysis and review
  - Assess progress, coverage, confidence
- Supporting reuse
  - Extract reusable safety artifacts

Motivating Formalization

Two distinct notions of formalization

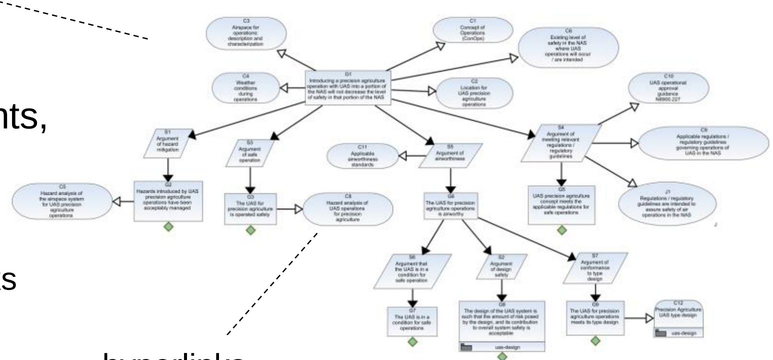
- Formal languages
  - Natural language
  - Controlled natural language
  - Formal assurance language
- Formal structures
  - Formalize the “scaffolding” to support automation
  - Support range of languages
  - Support range of reasoning structures

Argument Structures and Safety Cases

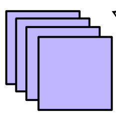


hyperlinks

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

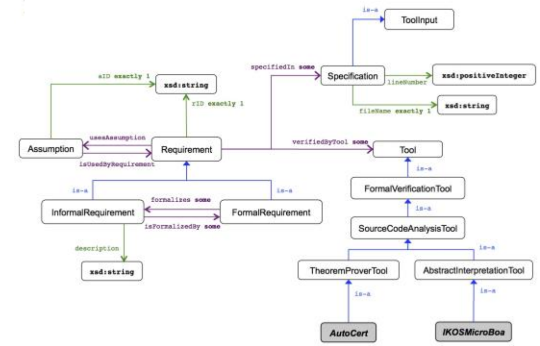


External Documents
e.g., hazard logs, requirements, etc.

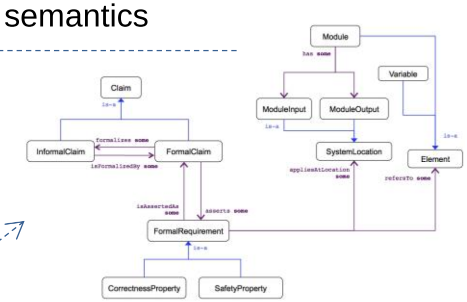


hyperlinks

hyperlinks



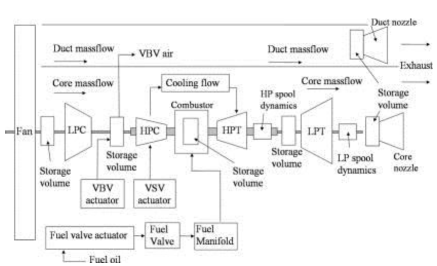
semantics

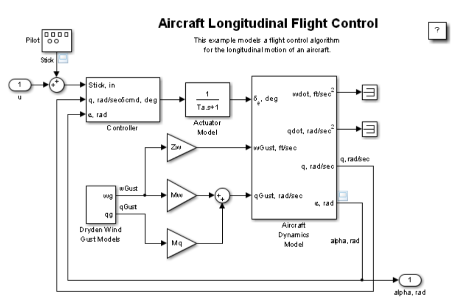


Domain model

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

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

attribute ::= attributeName param*
param ::= String | Int | Nat | nodeID | sameNodeTypeID | goalNodeId | strategyNodeId | evidenceNodeId | assumptionNodeId | contextNodeId | justificationNodeId | contextNodeId | userDefinedEnum

- userDefinedEnum

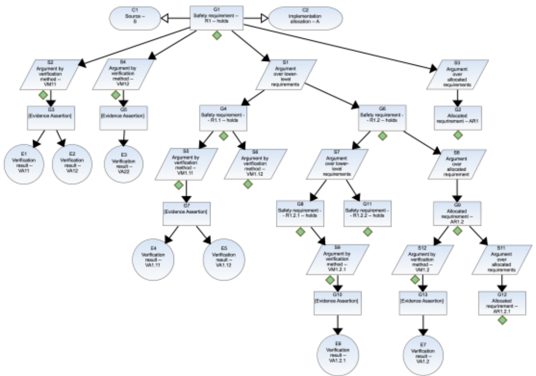
  severity ::= catastrophic | hazardous | major | minor | noSafetyEffect
  likelihood ::= frequent | probable | remote | extremelyRemote | extremelyImprobable

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

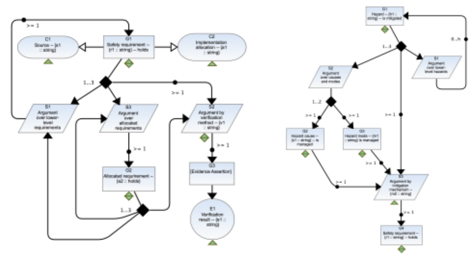
Example

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

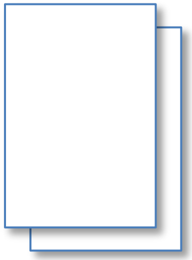


Pattern Library

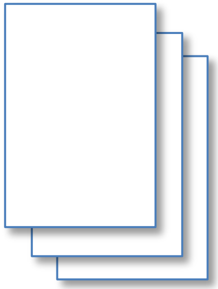


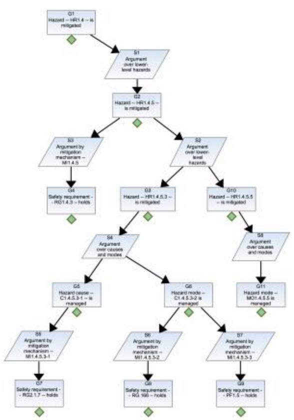
Artifacts

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



Bidirectional Mapping





Argument Fragments

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

Tabular Requirements Specifications

Hazards Table

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

System Requirements Table

ID	Requirement	Source	Allocation	Verification Method	Verification Allocation
RS.1.4.3	Critical systems must be redundant	AFSRB	RF.1.1.1.1.3		
RS.1.4.3.1	The system shall provide independent and redundant channels to the pilot	AFSRB			

Functional Requirements Table

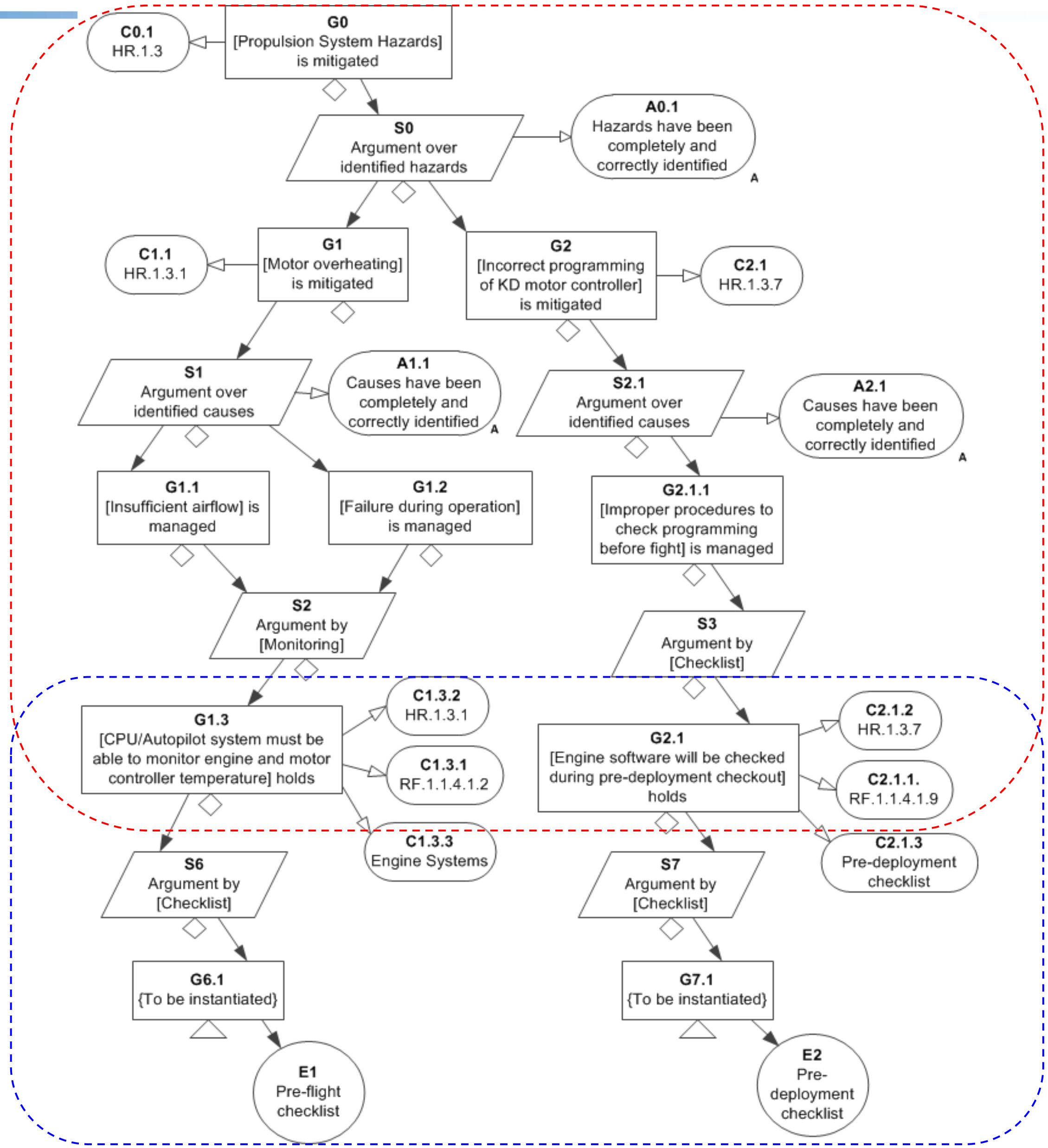
ID	Requirement	Source	Allocation	Verification Method	Verification Allocation
RF.1.1.1.1.3	FCS must be dually redundant	RS.1.4.3	FCS	Visual Inspection	FCS-CDR-20110701, TR20110826
RF.1.1.4.1.2	CPU/autopilot system must be able to monitor engine and motor controller temperature.	HR.1.3.1	Engine systems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

Mapping Multiple Tables

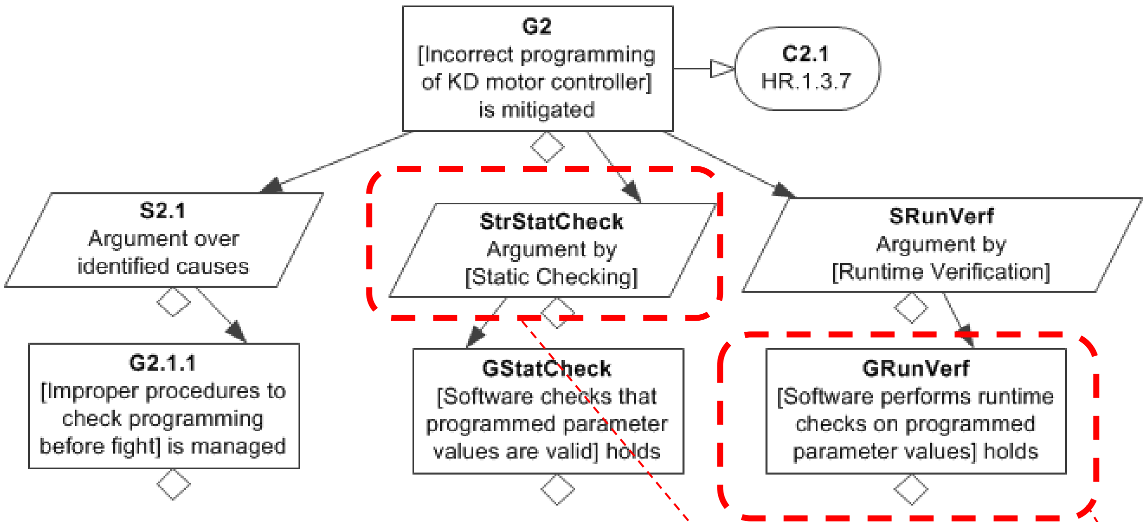
From hazards table

Linking tables using common content

From functional requirements table



Mapping Modifications



Evidence linking
(Strategy definition)

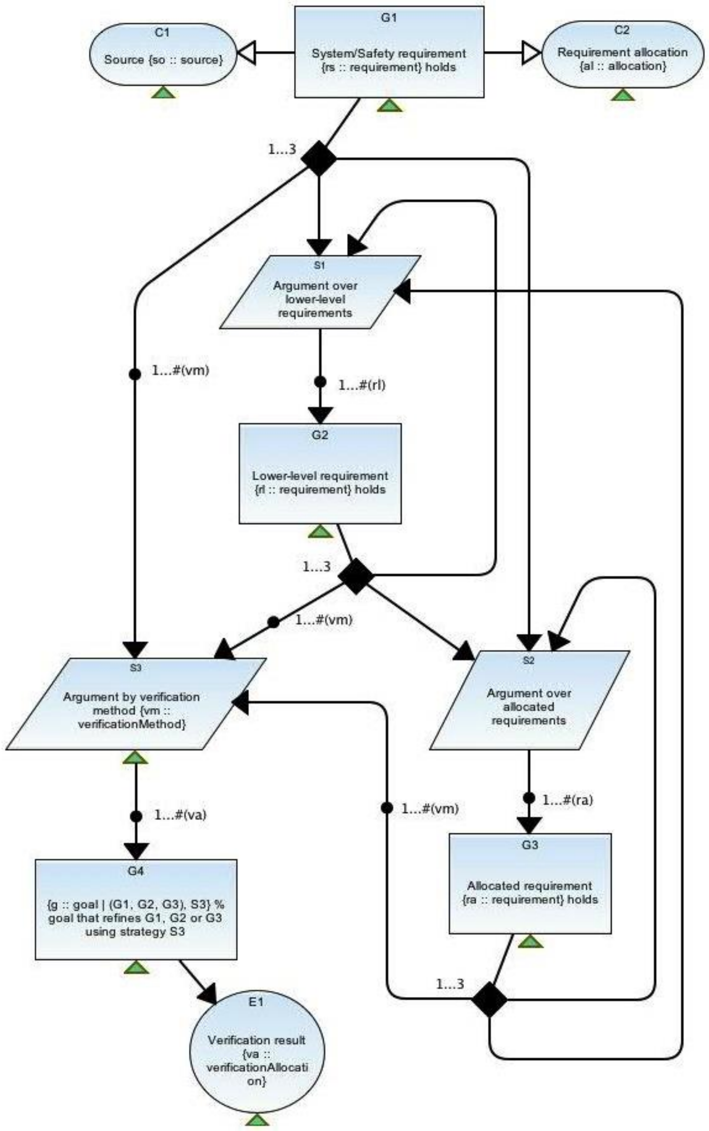
Claims definition

Hazards Table

ID	Hazard	Cause / Mode	Mitigation	Safety Requirement
HR.1.3	Propulsion system hazards			
HR.1.3.1	Motor overheating	Insufficient airflow	Monitoring	RF.1.1.4.1.2
		Failure during operation		
HR.1.3.7	Incorrect programming of KD motor controller	Improper procedures to check programming before flight	Checklist	RF.1.1.4.1.9
		-	*Static checking*	*GStatCheck*
		-	*Runtime Verification*	*GRunVerf*

Patterns

- Patterns represent classes of arguments
  - Typed variables
  - Labels
  - Constraints on data
- Well-formedness constraints
  - Well-founded recursion
  - Interaction between multiplicity and boilerplate
  - Restrictions on multiple parentage
- Can auto-instantiate from compatible dataset
- Semantics
  - Hypergraphs
  - Structure-preserving embeddings



Requirements Breakdown Pattern

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

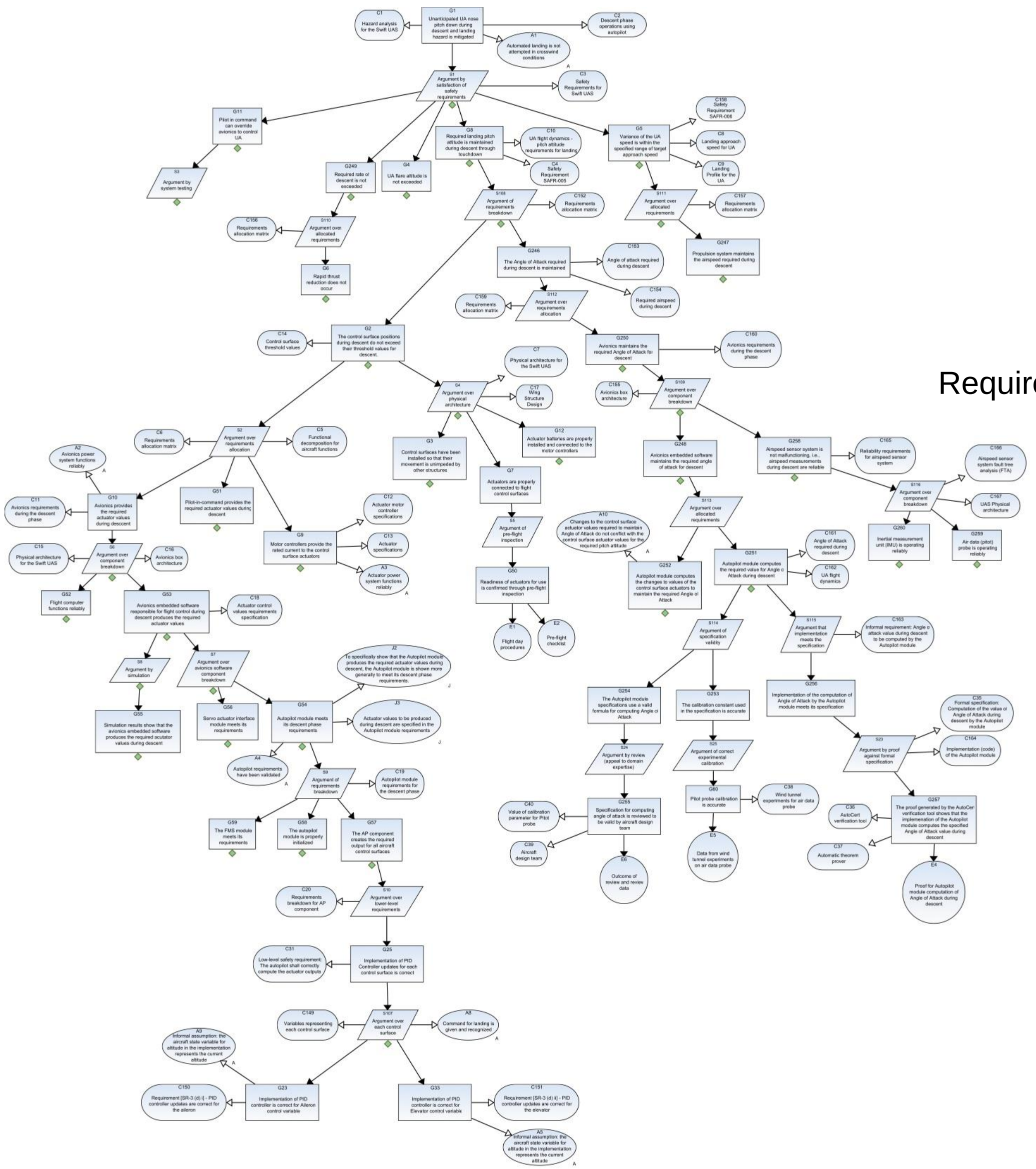
applied to

produces

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

- 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
- …

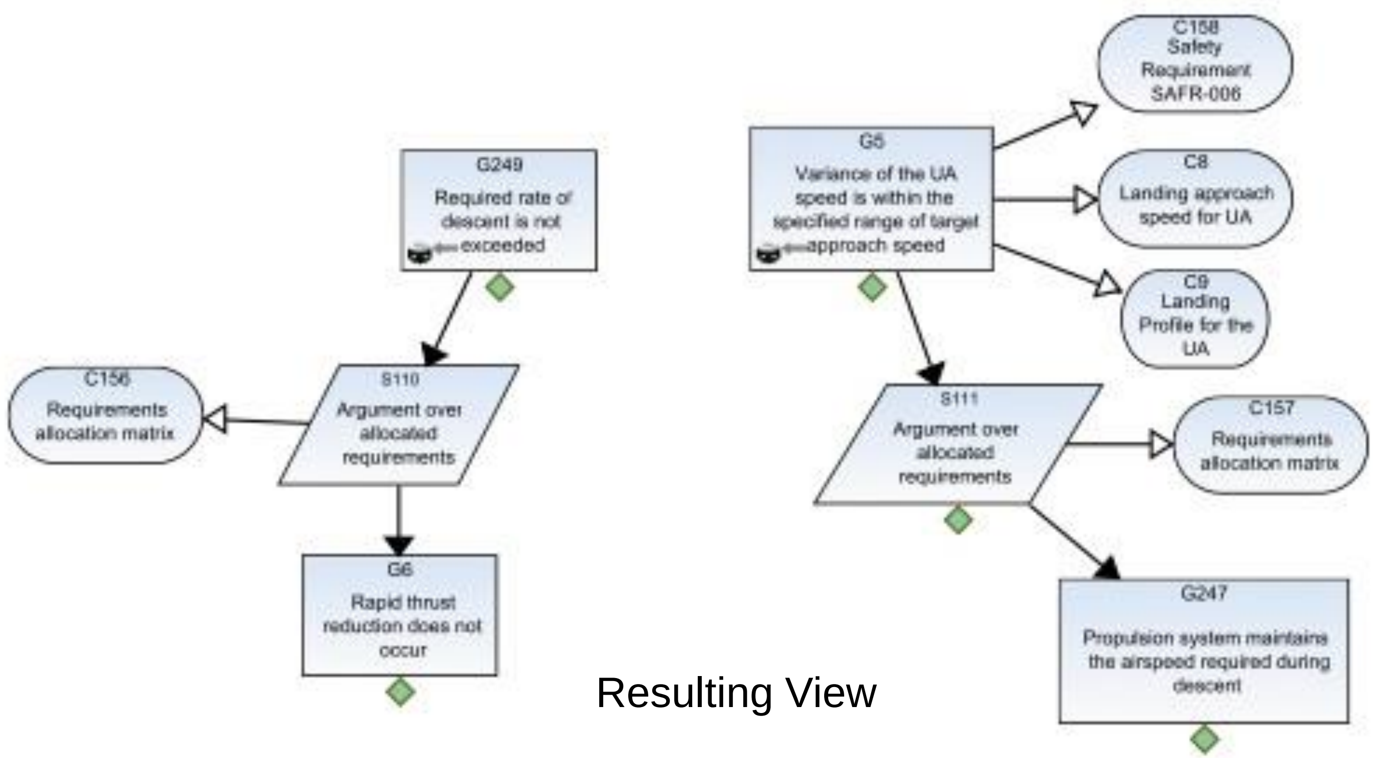
AQL Queries and Views: Example

- 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

Evaluation: Metrics

- Quantitative basis for evaluation
  - Internal measures of “quality” e.g.,
    - To what extent are claims developed – fully? partially?: **Claims coverage**
    - To what extent are high- / low-level safety requirements covered?: **Requirements coverage**
  - External measures of “quality” e.g.,
    - To what extent are hazards covered? – fully? partially?: **Hazard coverage**
  - Integrating confidence into a measure e.g.,
    - How well are the hazards covered?

- Quantitative basis for decision making
  - Tracking progress of an integrated systems development and safety process e.g.,
    - Coverage of hazards / claims / requirements at a specific milestone
    - Coverage for a specific sub-system / operational mode
  - Resource/Effort allocation e.g.,
    - Low coverage and/or Low confidence = Reallocate effort (contingent on cost-benefit analysis)

Language for Safety Case Metrics

- Build on AQL

- Examples
  - Number of claims that are related to hazards:
    #(type **has** claim and attributes **has** hazard))

  - A generic coverage metric: *Proportion of undeveloped claims to total number of claims*
    #(type **has** claim and status **has** undeveloped) / #(type **has** claim)

  - Specific metrics: *Coverage of claims for hazard H1*
    {#(type **has** claim and
    status **has** undeveloped and
    isBelow(attributes **has** hazard and description **has** H1))} /
    {#(type **has** claim and isBelow(attributes **has** hazard and
    description **has** H1))

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)

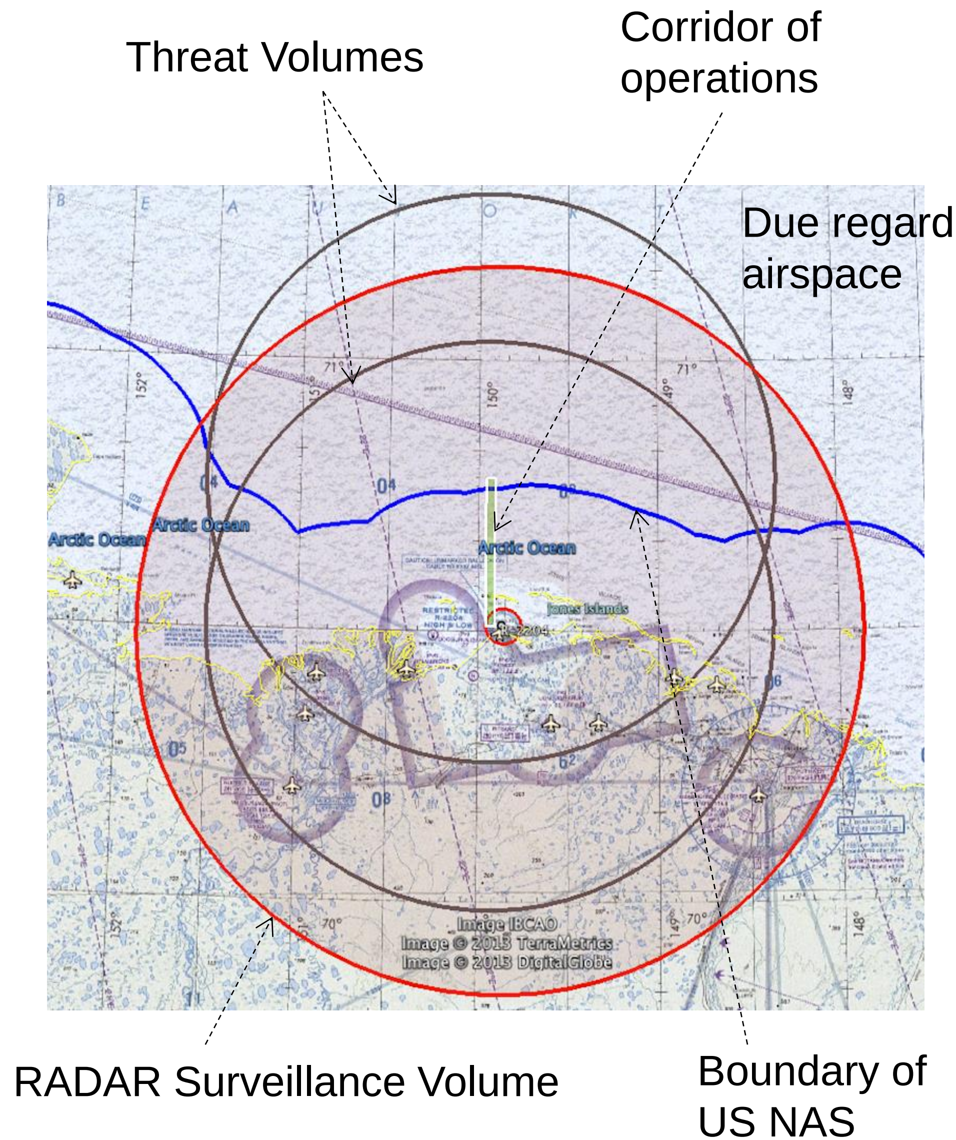
- Performing Earth Science measurements in the Arctic Ice
  - Off the coast of Alaska (Oliktok Point)
  - Satellite-based solution was too expensive
  - Use airborne instruments on UAS
    - Two classes of small UAS
    - NASA SIERRA; University of Alaska's Boeing Insitu ScanEagle
  - Too dangerous for 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

SIERRA UAV

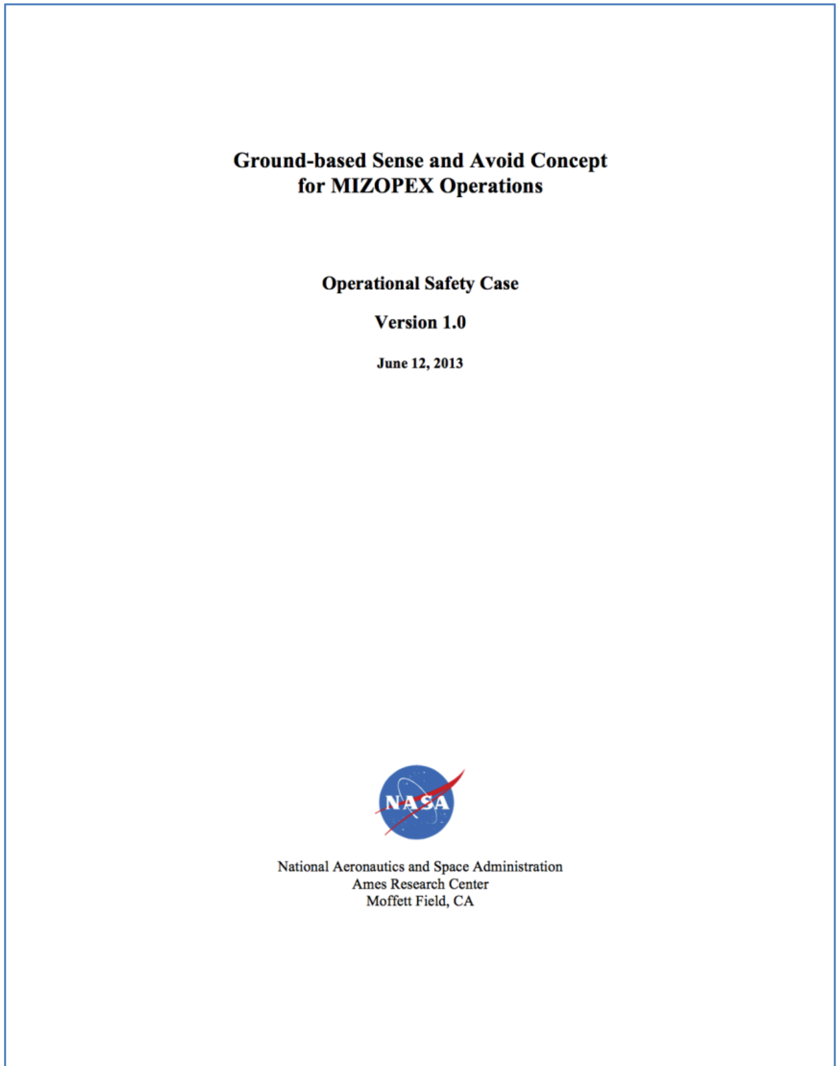
Air Defense RADAR for monitoring and airspace deconfliction



MIZOPEX GBSAA Hazard Analysis

- GBSAA Hazard
  - Known / unknown <u>state of the GBSAA system</u> (which may / may not be a deviation from its required operational state)
  - One or more known / unknown classes of *environmental conditions*
  - Combinations in different <u>*flight phases*</u>
  - Examples
    - <u>Loss of RADAR system to detect air traffic</u> in the surveillance volume, <u>*during outbound transit*</u> when *surveillance volume previously all clear*
    - <u>GBSAA functioning as required</u>, *with non cooperative aircraft in the threat volume not covered by the surveillance volume on an intercept flight path*, when <u>*UA is outbound in the transit corridor*</u>.
  - 5 known states, 8 flight phases, 3 classes of environmental conditions ~ 26 cases leading to potential mid-air collision
  - Collision with terrain managed through range safety

MIZOPEX GBSAA Operational Safety Case



Ground-based Sense and Avoid Concept
for MIZOPEX Operations

Operational Safety Case

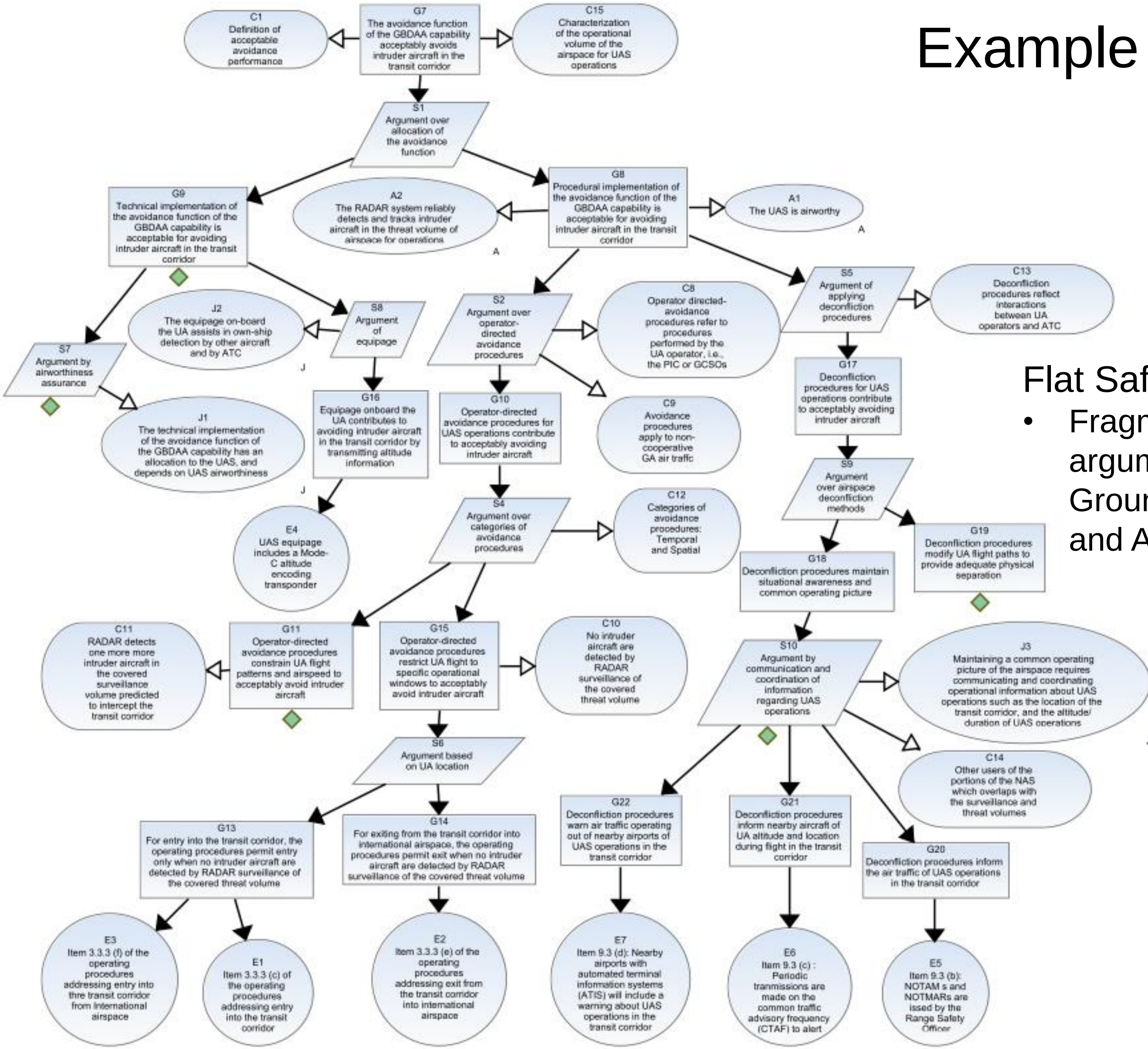
Version 1.0

June 12, 2013

National Aeronautics and Space Administration
Ames Research Center
Moffett Field, CA

- 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

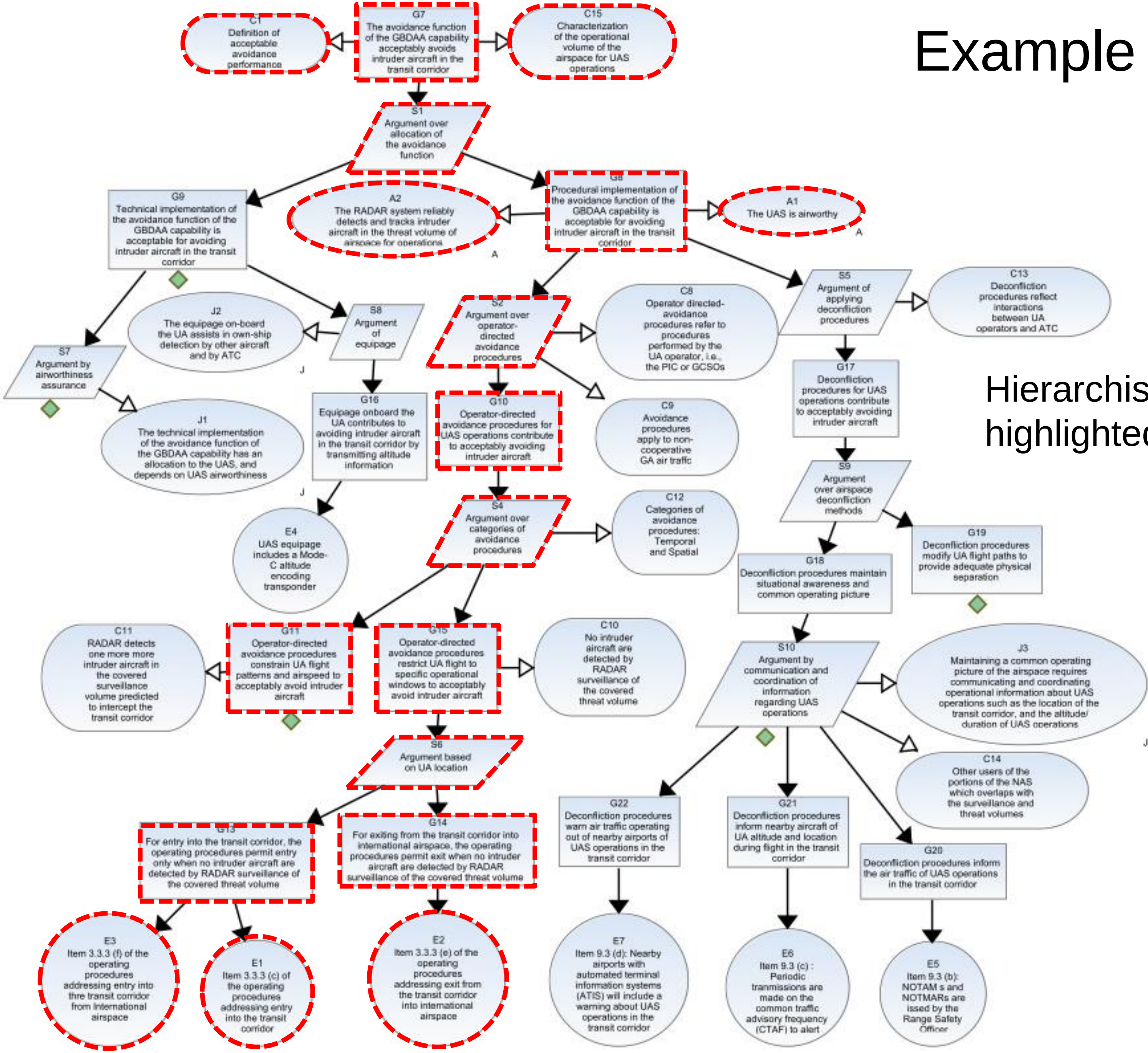
Example



Flat Safety Argument

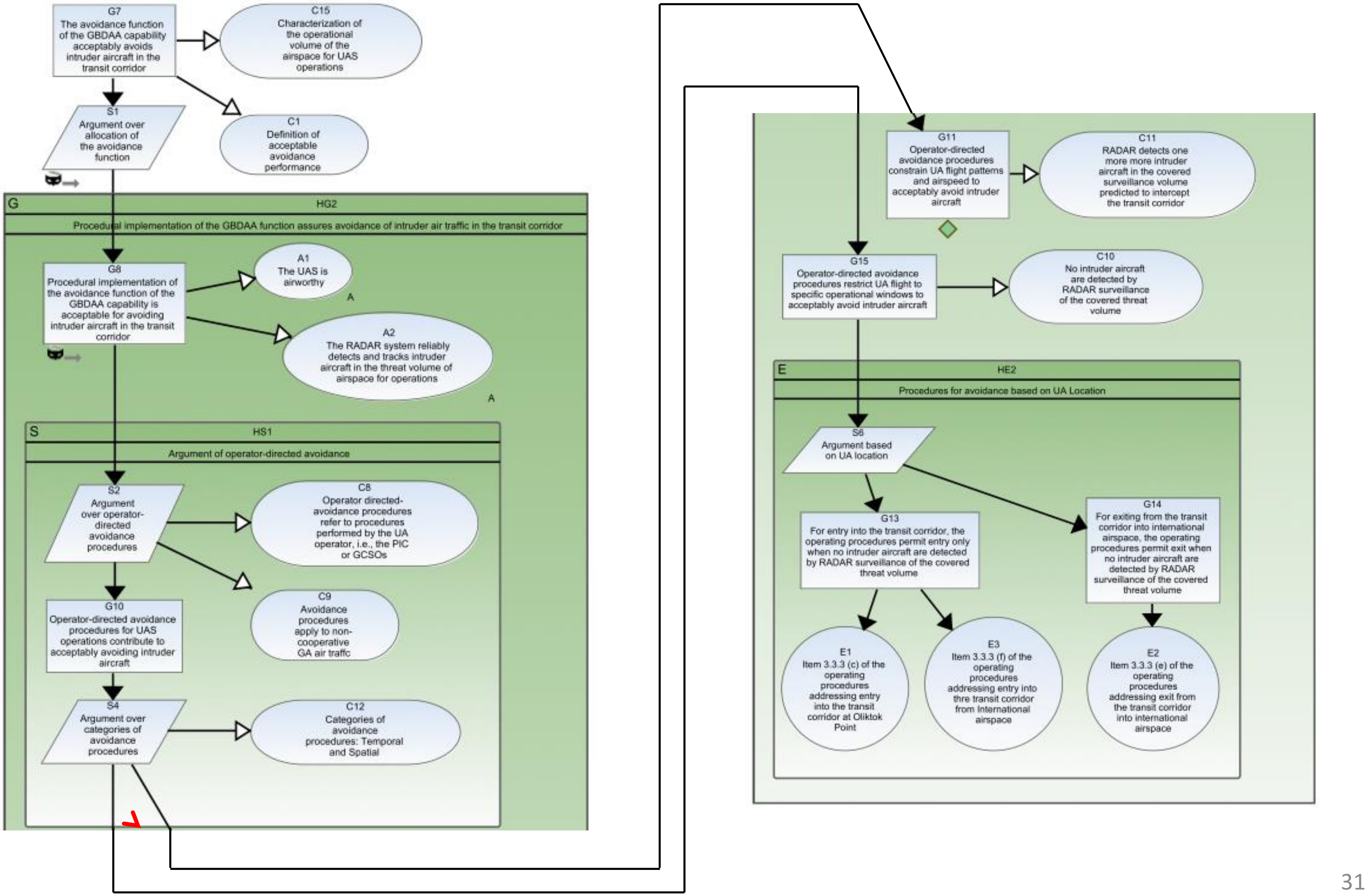
- Fragment of larger argument for Ground-based Detect and Avoid (GBDAA)

Example



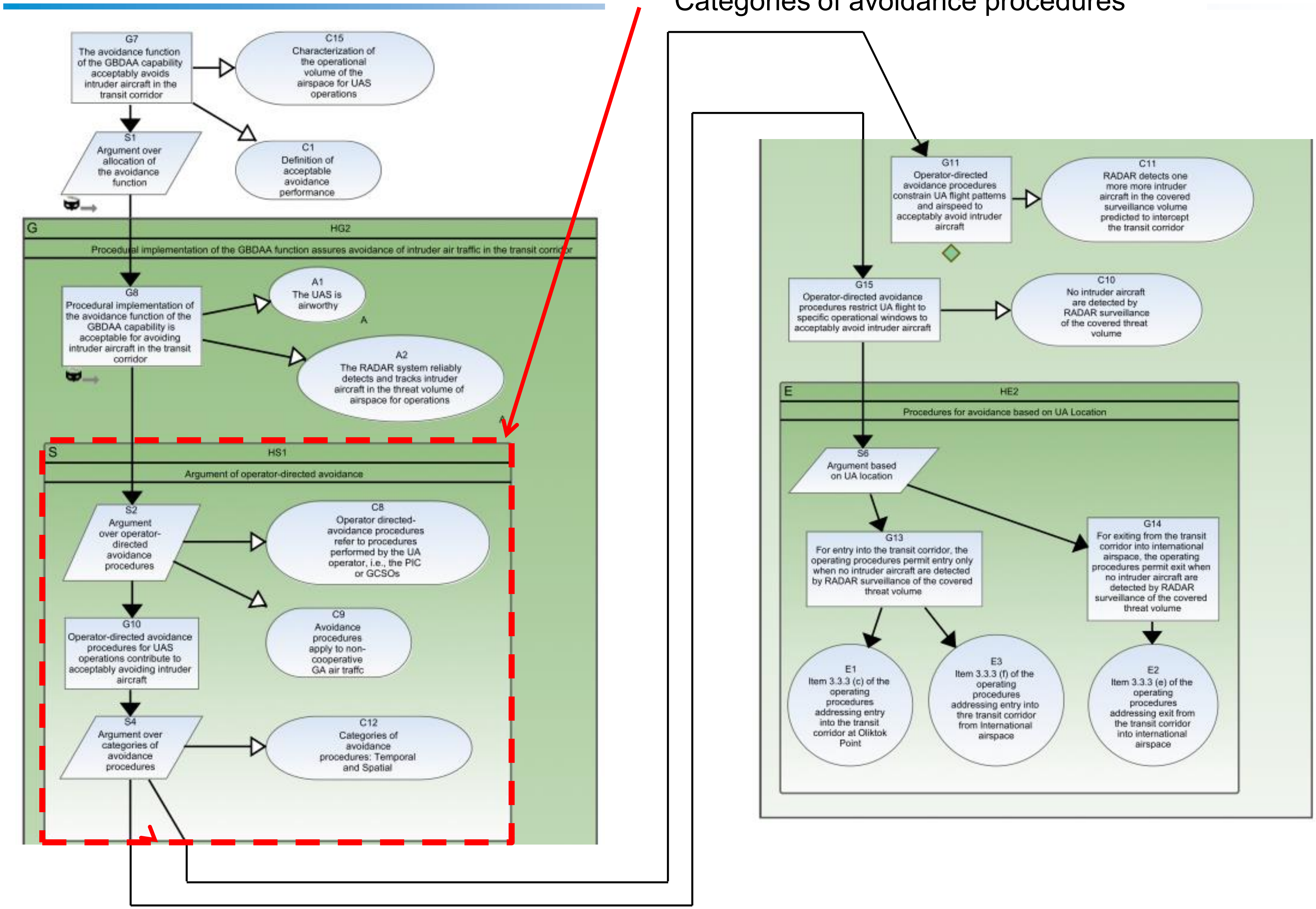
Hierarchisation of highlighted slice

Hierarchised Fragment



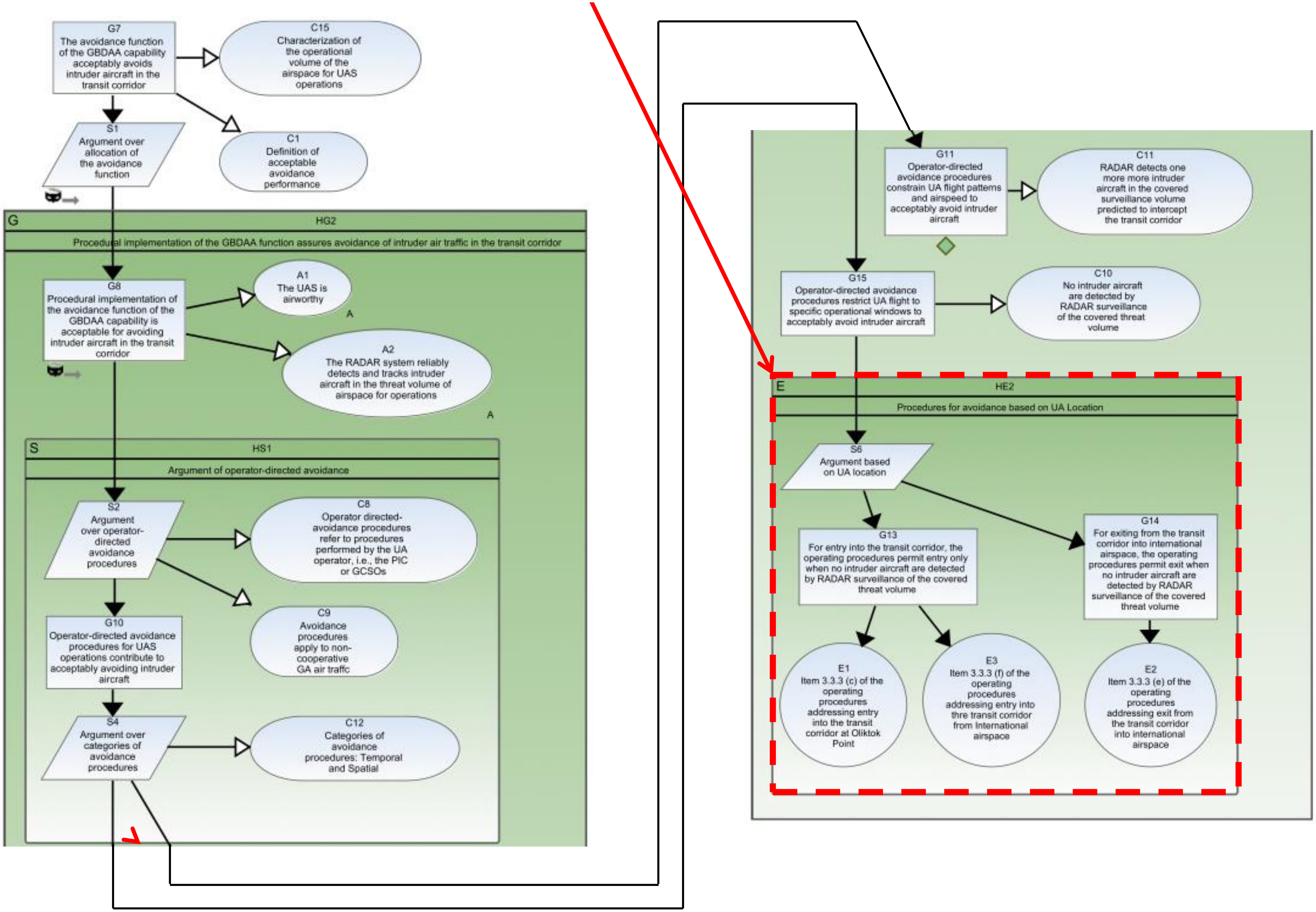
A. Hierarchical Strategy (Open)

- Representing a chain of strategies
- “Operator directed avoidance” followed by “Categories of avoidance procedures”



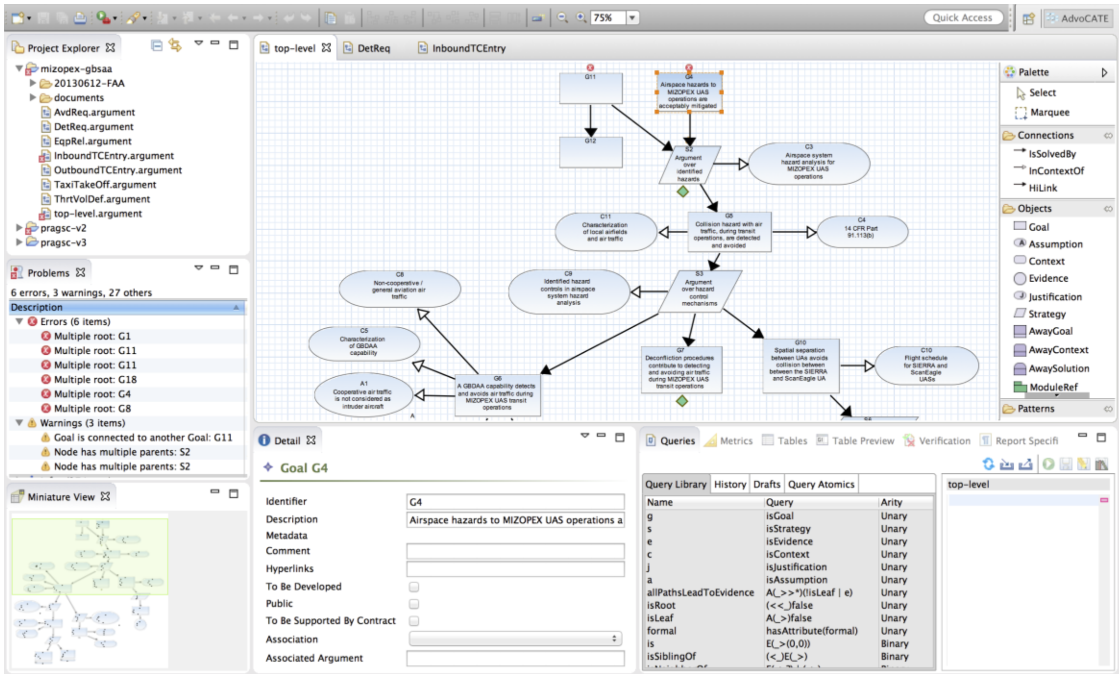
B. Hierarchical Evidence (Open)

- Representing procedures for avoidance based on aircraft location



Tool Support

AdvoCATE: Assurance Case Automation Toolset



- Creation of safety / assurance argument
  - Hyperlinks in nodes to documents, data for evidence, context, etc.
  - Metadata on nodes: hazards, high/low requirements, risk (severity, likelihood), provenance

Vision

Safety information, assurance and risk management (SMART) Dashboard

- 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

Conclusions

- 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 → lightweight → formal

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

Please consider submitting a paper

3rd International Workshop on Assurance Cases for Software-intensive Systems (ASSURE 2015)

September 22, 2015. Delft, The Netherlands.

Collocated with SAFECOMP 2015

Paper submission deadline: May 22, 2015

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