National Aeronautics and Space Administration

Automating the Generation of Heterogeneous Aviation Safety Cases

Ewen Denney
Senior Computer Scientist

2011 Annual Technical Meeting
May 10–12, 2011
St. Louis, MO

Safety Cases

Safety Case*

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 operating environment.

- Contrast with process-based approaches

- Safety is interpreted as the freedom from those hazards considered as presenting unacceptable mishap risk.
  - Uses a process for hazard identification and analysis
  - Requires a process for defining/ characterizing “unacceptable risk”

- Safety cases
  - Justify the measures taken for hazard mitigation
  - Can be represented in various ways
    - Graphically
    - Structured text

* UK Ministry of Defence, Defence Standard 00-56, Part 1, Issue 4, June 2007.

Automation

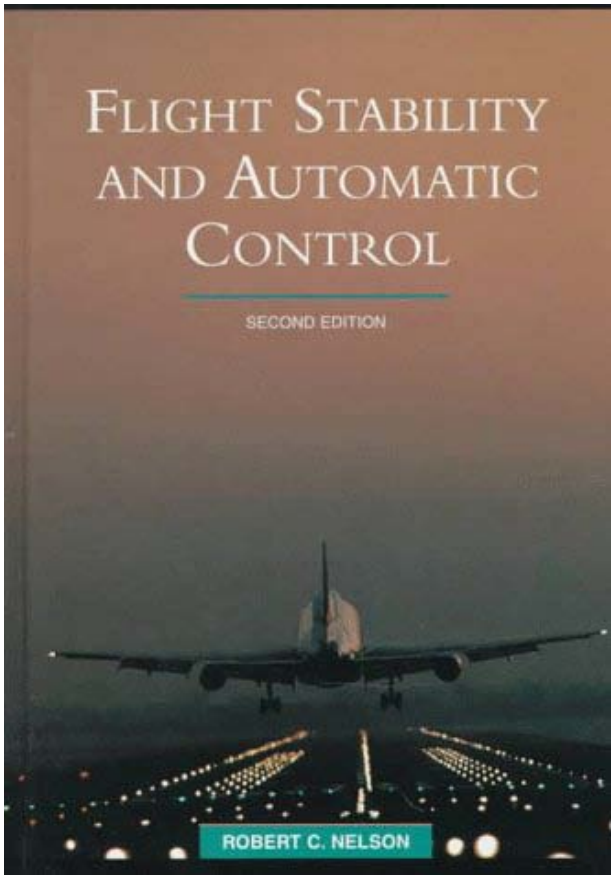
- Safety cases are typically constructed manually
  - Laborious
  - Static
  - Top-down
  - Susceptible to confirmation bias
- We aim to automatically assemble (fragments of) safety cases from
  - Engineering data
  - Safety analyses
  - Tool output
    - Formal verification
    - Simulation
    - Testing
  - Compliance with regulations
- Automation can support
  - Bottom-up construction
  - Tracing to large amounts of data
  - More explicit reasoning, less gaps
  - Iterative Development
  - Queries and abstractions: multiple views

Heterogeneity

- A safety argument requires the integration of diverse sources of information

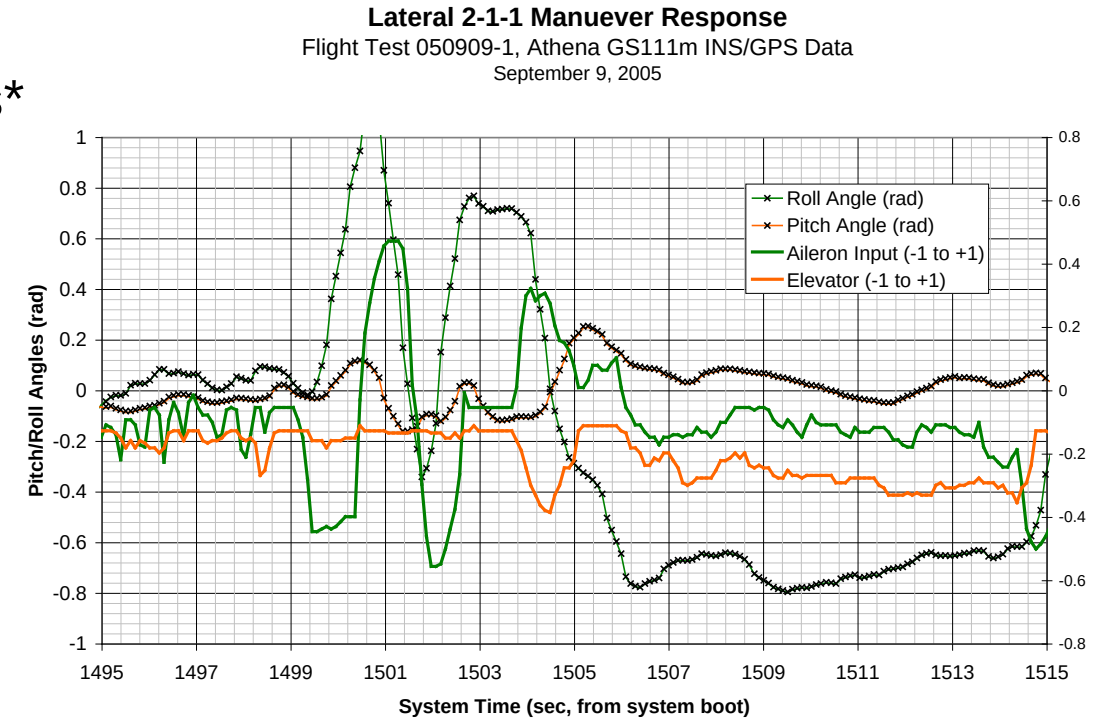
Heterogeneity

- A safety argument requires the integration of diverse sources of information
  - Mathematical theory



Heterogeneity

- A safety argument requires the integration of diverse sources of information
  - Mathematical theory
  - Flight test maneuvers*



* Corey Ippolito, Design of an Autonomous Control System for a Small-Scale UAV.

Heterogeneity

- A safety argument requires the integration of diverse sources of information
  - Mathematical theory
  - Flight test maneuvers
  - Calibration experiments

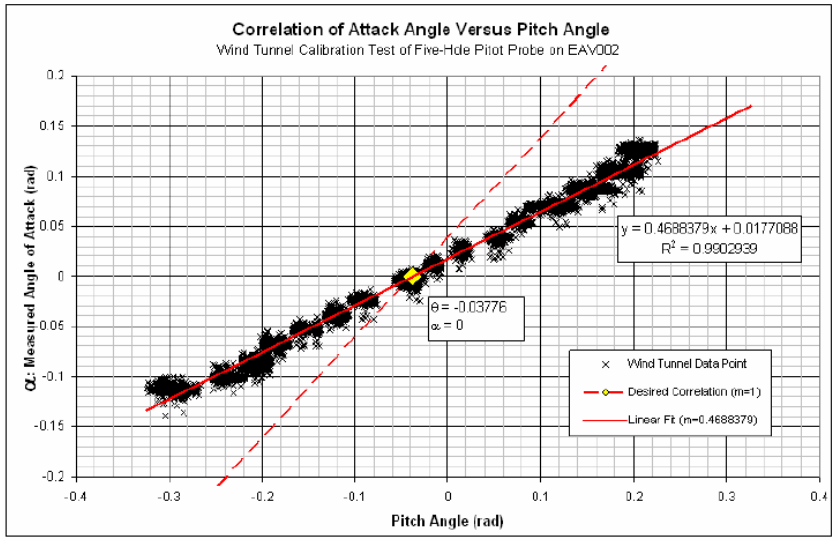
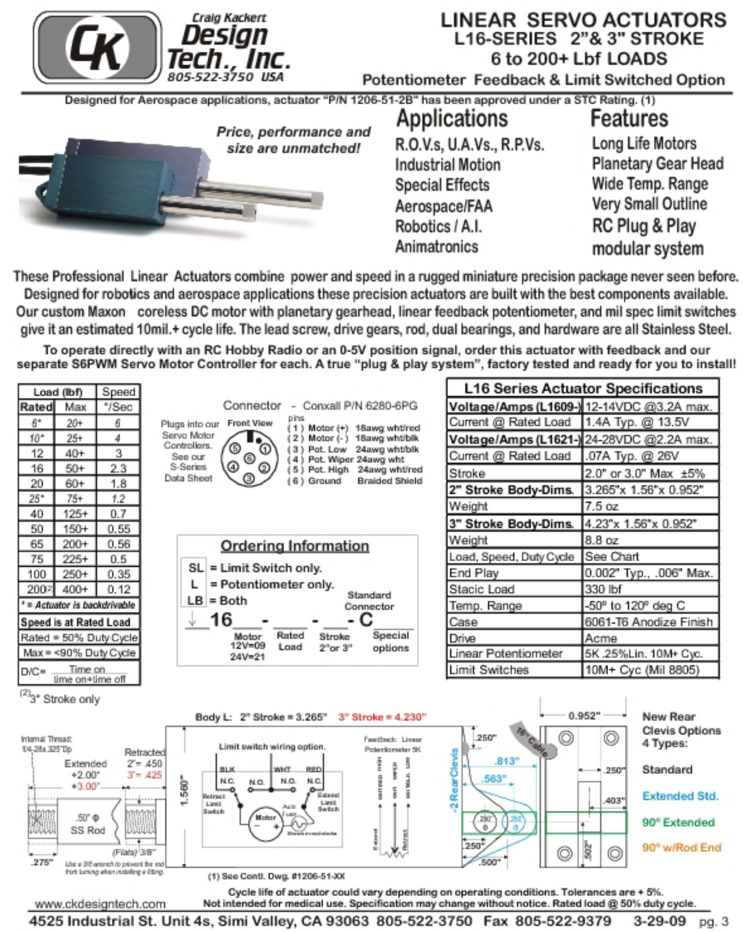


Table 2. Resulting Calibration Coefficients

$\theta_{\alpha=0}$	=	-0.03776	rad
		-2.163	deg
K'	=	4.526366	rad^{-1}
m	=	0.4688379	
K	**=**	**2.122132**	**rad^{-1}**

Heterogeneity

- A safety argument requires the integration of diverse sources of information
  - Mathematical theory
  - Flight test maneuvers
  - Calibration experiments
  - Manufacturer datasheets



Craig Kackert Design Tech., Inc. 805-522-3750 USA

LINEAR SERVO ACTUATORS
L16-SERIES 2"& 3" STROKE
6 to 200+ Lbf LOADS
Potentiometer Feedback & Limit Switched Option

Designed for Aerospace applications, actuator "P/N 1206-51-2B" has been approved under a STC Rating. (1)

Price, performance and size are unmatched!

Applications
R.O.V.s, U.A.Vs., R.P.Vs.
Industrial Motion
Special Effects
Aerospace/FAA
Robotics / A.I.
Animatronics

Features
Long Life Motors
Planetary Gear Head
Wide Temp. Range
Very Small Outline
RC Plug & Play
modular system

These Professional Linear Actuators combine power and speed in a rugged miniature precision package never seen before. Designed for robotics and aerospace applications these precision actuators are built with the best components available. Our custom Maxon coreless DC motor with planetary gearhead, linear feedback potentiometer, and mil spec limit switches give it an estimated 10mil.+ cycle life. The lead screw, drive gears, rod, dual bearings, and hardware are all Stainless Steel.

To operate directly with an RC Hobby Radio or an 0-5V position signal, order this actuator with feedback and our separate S6PWM Servo Motor Controller for each. A true "plug & play system", factory tested and ready for you to install!

Load (lbf)		Speed
Rated	Max	"/Sec
6*	20+	6
10*	25+	4
12	40+	3
16	50+	2.3
20	60+	1.8
25*	75+	1.2
40	125+	0.7
50	150+	0.55
65	200+	0.56
75	225+	0.5
100	250+	0.35
200[2]	400+	0.12

* = Actuator is backdrivable
Speed is at Rated Load
Rated = 50% Duty Cycle
Max = <90% Duty Cycle
D/C = Time on / time on+time off

[2] 3" Stroke only

Connector - Conxall P/N 6280-6PG
Plugs into our Servo Motor Controllers. See our S-Series Data Sheet
Front View
pins
(1) Motor (+) 18awg wht/red
(2) Motor (-) 18awg wht/blk
(3) Pot. Low 24awg wht/blk
(4) Pot. Wiper 24awg wht
(5) Pot. High 24awg wht/red
(6) Ground Braided Shield

Ordering Information
SL = Limit Switch only.
L = Potentiometer only.
LB = Both
16__ - ___ - __ - C ___
Motor 12V=09 24V=21; Rated Load; Stroke 2" or 3"; Standard Connector; Special options

L16 Series Actuator Specifications	
Voltage/Amps (L1609-)	12-14VDC @3.2A max.
Current @ Rated Load	1.4A Typ. @ 13.5V
Voltage/Amps (L1621-)	24-28VDC @2.2A max.
Current @ Rated Load	.07A Typ. @ 26V
Stroke	2.0" or 3.0" Max ±5%
2" Stroke Body-Dims.	3.265"x 1.56"x 0.952"
Weight	7.5 oz
3" Stroke Body-Dims.	4.23"x 1.56"x 0.952"
Weight	8.8 oz
Load, Speed, Duty Cycle	See Chart
End Play	0.002" Typ., .006" Max.
Stacic Load	330 lbf
Temp. Range	-50º to 120º deg C
Case	6061-T6 Anodize Finish
Drive	Acme
Linear Potentiometer	5K .25%Lin. 10M+ Cyc.
Limit Switches	10M+ Cyc (Mil 8805)

Body L: 2" Stroke = 3.265" 3" Stroke = 4.230"
Internal Thread: 1/4-28x.325"Dp
Extended +2.00" +3.00"
Retracted 2" = .450 3" = .425
.50" Φ SS Rod
(Flats) 3/8"
.275"
1.560"
Limit switch wiring option.
BLK WHT RED
N.C. N.O. N.O. N.C.
Retract Limit Switch
Extend Limit Switch
Motor
Auto Fuse
Feedback: Linear Potentiometer 5K
Extend Retract
-2 Rear Clevis
.250" .813" .563" .250" .500"
18" Cable
0.952"
.250" .403" .502"
New Rear Clevis Options 4 Types:
Standard
Extended Std.
90º Extended
90º w/Rod End

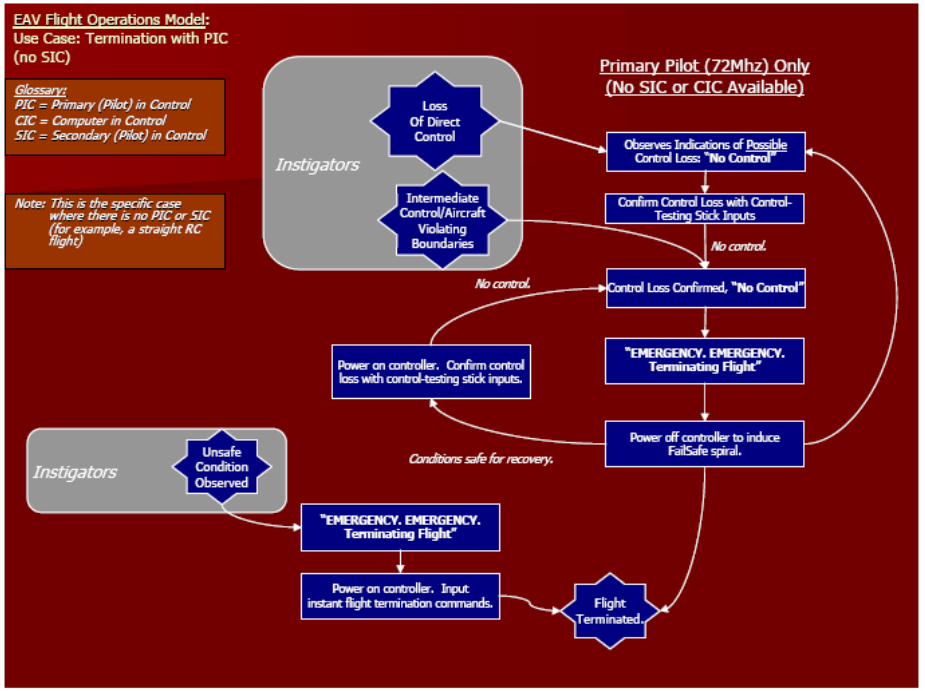
(1) See Contl. Dwg. #1206-51-XX

Cycle life of actuator could vary depending on operating conditions. Tolerances are + 5%.
Not intended for medical use. Specification may change without notice. Rated load @ 50% duty cycle.

www.ckdesigntech.com
4525 Industrial St. Unit 4s, Simi Valley, CA 93063 805-522-3750 Fax 805-522-9379 3-29-09 pg. 3

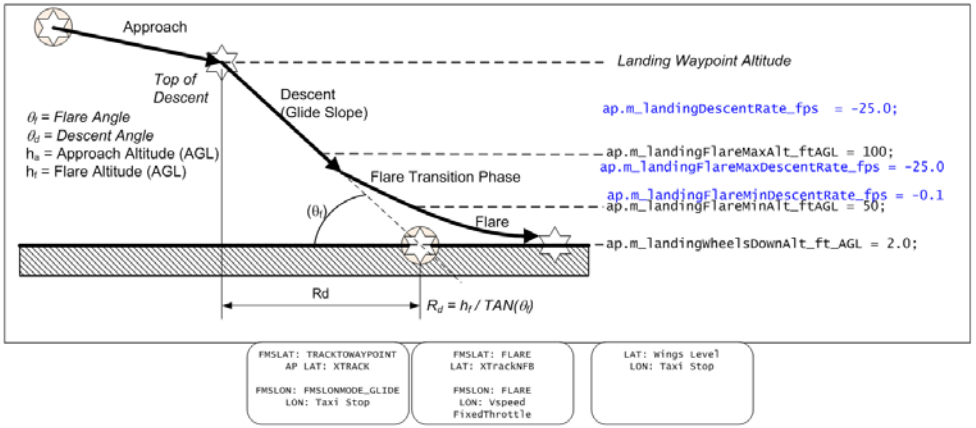
Heterogeneity

- A safety argument requires the integration of diverse sources of information
  - Mathematical theory
  - Flight test maneuvers
  - Calibration experiments
  - Manufacturer datasheets
  - Flight operations procedures



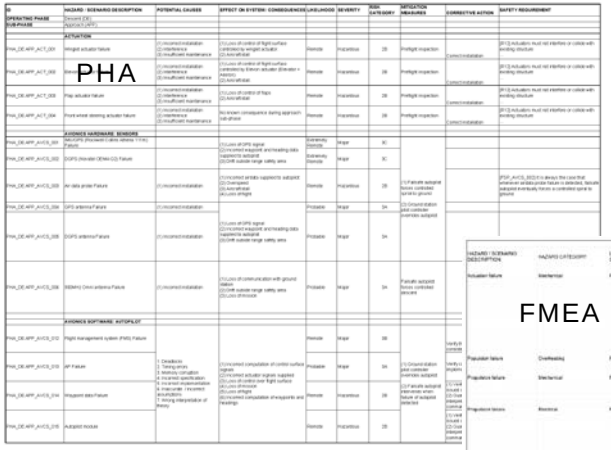
Heterogeneity

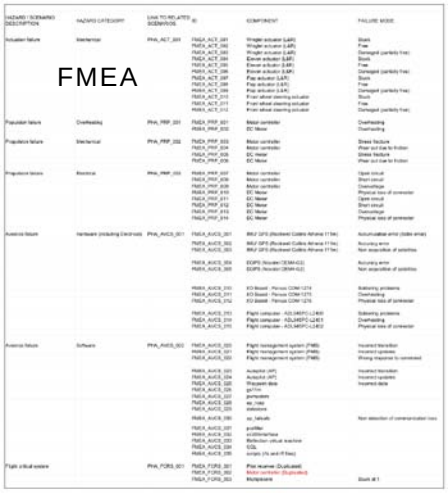
- A safety argument requires the integration of diverse sources of information
  - Mathematical theory
  - Flight test maneuvers
  - Calibration experiments
  - Manufacturer datasheets
  - Flight operations procedures
  - Software verification



Heterogeneity

- A safety argument requires the integration of diverse sources of information
  - Mathematical theory
  - Flight test maneuvers
  - Calibration experiments
  - Manufacturer datasheets
  - Flight operations procedures
  - Software verification
  - Systems and software safety analyses



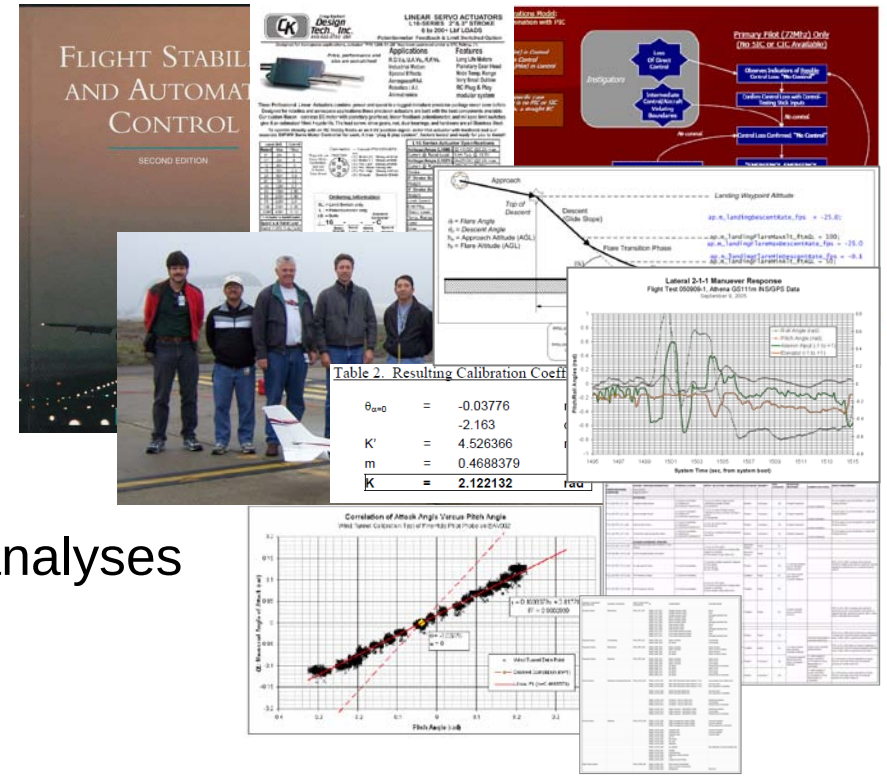


Heterogeneity

- A safety argument requires the integration of diverse sources of information
  - Mathematical theory
  - Flight test maneuvers
  - Calibration experiments
  - Manufacturer datasheets
  - Flight operations procedures
  - Software verification
  - Systems and software safety analyses
  - Expert opinion

Heterogeneity

- A safety argument requires the integration of diverse sources of information
  - Mathematical theory
  - Flight test maneuvers
  - Calibration experiments
  - Manufacturer datasheets
  - Flight operations procedures
  - Software verification
  - Systems and software safety analyses
  - Expert opinion

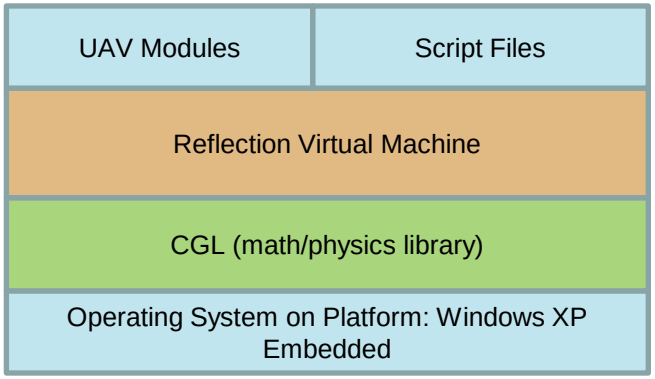


Challenge:

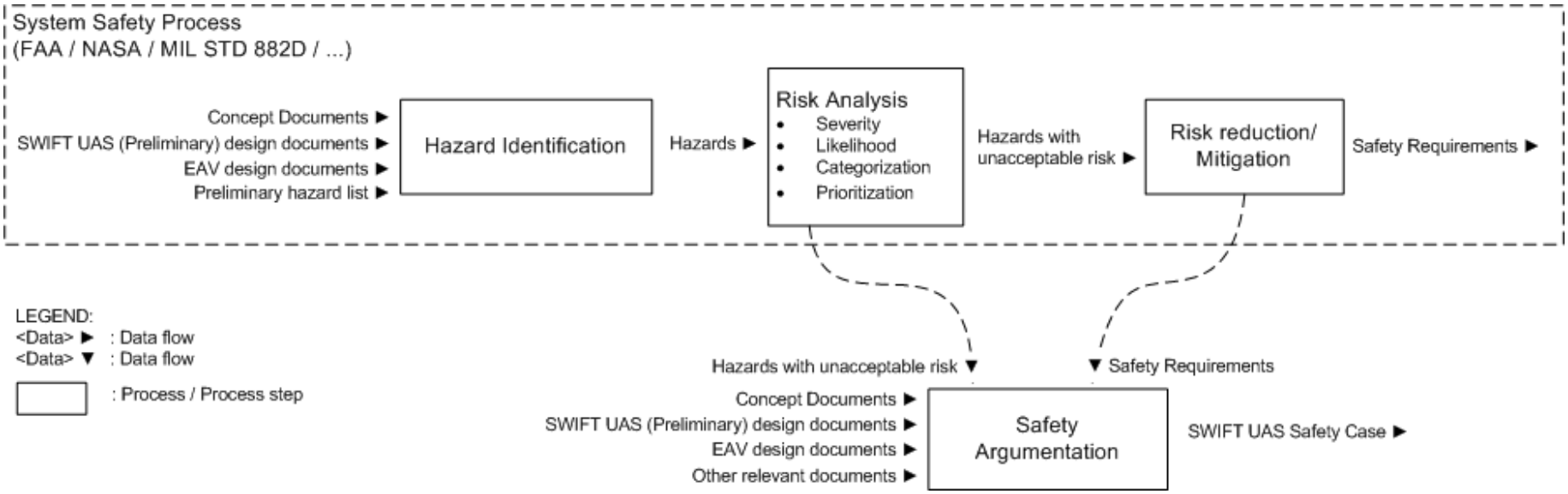
A methodology and framework that allows integrated reasoning about disparate forms of evidence

Target System: Swift UAS

- Experimental Autonomous Vehicle Program
  - Research vehicle program, Code TI, NASA Ames
- Electric motor, lithium batteries, high glide ratio, all composite wing structure, steel/aluminum fuselage frame
- Swift configuration
  - Very low latency, computer controlled with multiple onboard full-power CPUs
  - UAS consists of ground system (GSC) and flight system (UAV)
- Reflection architecture
  - C and C++ component-based plug-and-play infrastructure
  - Real-time embedded avionics system architecture
- Commands in Reflection script, uploaded from ground system, and interpreted by onboard VM



Safety Methodology



- Philosophy: Safety argumentation process driven by system safety process

- Safety argumentation process
  - Argue safety over all identified hazards having unacceptable risk
  - Automation and heterogeneity
- System safety process
  - Hazard identification and risk analysis
  - Safety requirements to eliminate risk or reduce to acceptable levels
  - Iterative and successive refinement
- Phased development instead of isolated development
  - Safety argumentation influences, and is influenced by, system development and safety analysis

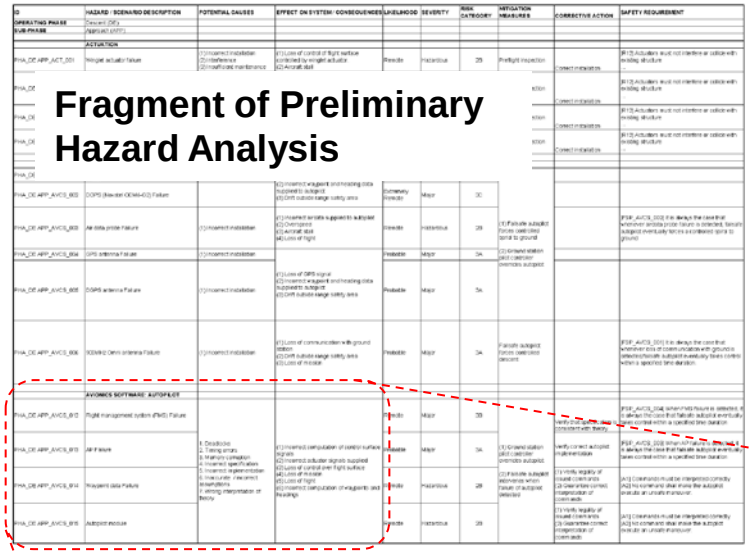
Safety Methodology

- Hazard identification for the Swift UAS (Ongoing)
  - Failure hazards
  - Situations in the operating environment
  - Interactions, ConOps

Fragment of identified hazards (PHL)

NO.	SYSTEM	SUB SYSTEM	COMPONENT / LOCATION	STATE / SITUATION	IS HAZARD?	RATIONALE
1	AIRCRAFT					
1.1		**Actuation**		Failure	Yes	Actuation failure may result in an aircraft which cannot be predictably manoeuvred
1.1.1		*Control surface actuators*		Failure	Yes	Control surface actuator failures can result in an unmanoeuvrable aircraft
1.1.1.1			Winglet actuator (L & R)	Failure	Yes	
1.1.1.2			Elevon actuator (L & R)	Failure	Yes	Failure of elevon actuator results in failure to control elevators and ailerons
1.1.1.3			Flap actuator (L & R)	Failure	Yes	Failure of flap actuator results in failure to control flaps
1.1.2		*Steering actuators*		Failure	Yes	Streering actuator failure results in an aircraft that cannot be steered on the ground introducing a potential for runway incursion / excursion
1.1.2.1			Front wheel steering actuator	Failure	Yes	
1.2		**Propulsion**		Failure	Yes	Propulsion failure results in loss of lift and/ or thrust
1.2.1		*Electric motor system*		Failure	Yes	
1.2.1.1			Motor controller	Failure	Yes	
1.2.1.2			DC motor	Failure	Yes	
1.3		**Avionics**		Failure	Yes	Avionics failure results in loss of control
1.3.1		*Avionics hardware*		Failure	Yes	Avionics hardware failure may result in loss of control
1.3.1.1			Flight sensors	Failure	Yes	Sensor failures will result in incorrect computation of control parameters
1.3.1.1.1			IMU/GPS (Rockwell Collins Athena 111m)	Failure	Yes	
1.3.1.1.2			DGPS (Novatel OEM4-G2)	Failure	Yes	
1.3.1.1.3			Air data probe	Failure	Yes	
1.3.1.1.4			GPS antenna	Failure	Yes	
1.3.1.1.5			DGPS antenna	Failure	Yes	
1.3.1.1.6			900MHz Omni antenna	Failure	Yes	
1.3.1.2			I/O Board - Parvus COM-1274	Failure	Yes	
1.3.1.3			Flight computer - ADL945PC-L2400	Failure	Yes	Flight computer failure may result in loss of control
1.3.1.4			Power board - Tri-M HPSC104-SER	Failure	Yes	
1.3.1.5			Pontech motor controller	Failure	Yes	
			Radio modem	Failure	Yes	
1.3.2		*Avionics software*		Failure	Yes	Avionics software failure may result in loss of control
1.3.2.1			Autopilot	Failure	Yes	Autopilot failure may result in loss of control
1.3.2.1.1			Flight management system (FMS)	Failure	Yes	
1.3.2.1.2			Autopilot (AP)	Failure	Yes	
1.3.2.1.3			Waypoint	Failure	Yes	
1.3.2.2			gs11m	Failure	Yes	
1.3.2.3			javmodem	Failure	Yes	
1.3.2.4			ap_rcap	Failure	Yes	
1.3.2.5			datastore	Failure	Yes	
1.3.2.6			ap_failsafe	Failure	Yes	
1.3.2.7			pcsfilter	Failure	Yes	
1.3.2.8			sv203interface	Failure	Yes	
1.3.2.9			Reflection virtual machine	Failure	Yes	
1.3.2.10			CGL	Failure	Yes	
1.3.2.11			Scripts (rff and rfs files)	Failure	Yes	

Safety Methodology

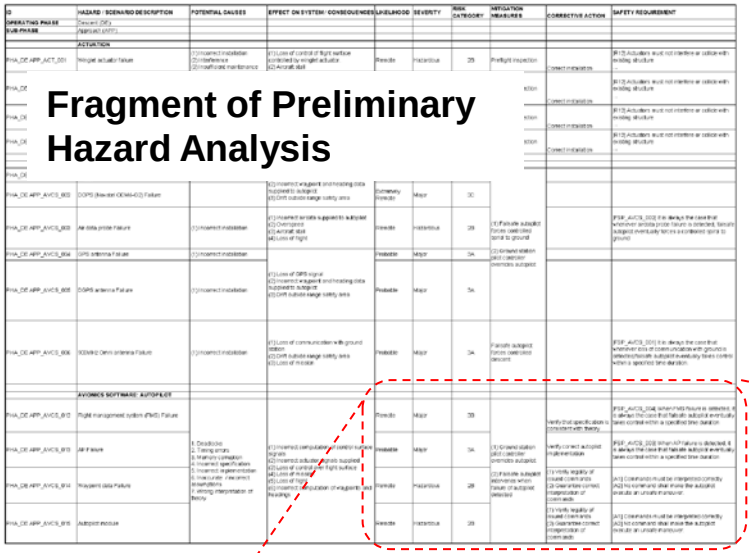


Fragment of Preliminary Hazard Analysis

PHA is ongoing

ID	HAZARD / SCENARIO DESCRIPTION	POTENTIAL CAUSES	EFFECT ON SYSTEM / CONSEQUENCES
	AVIONICS SOFTWARE: AUTOPILOT		
PHA_DE.APP_AVCS_012	Flight management system (FMS) Failure	1. Deadlocks 2. Timing errors 3. Memory corruption 4. Incorrect specification 5. Incorrect implementation 6. Inaccurate / incorrect assumptions 7. Wrong interpretation of theory	(1) Incorrect computation of control surface signals (2) Incorrect actuator signals supplied (3) Loss of control over flight surface (4) Loss of mission (5) Loss of flight (6) Incorrect computation of waypoints and headings
PHA_DE.APP_AVCS_013	AP Failure		
PHA_DE.APP_AVCS_014	Waypoint data Failure		
PHA_DE.APP_AVCS_015	Autopilot module		

Safety Methodology



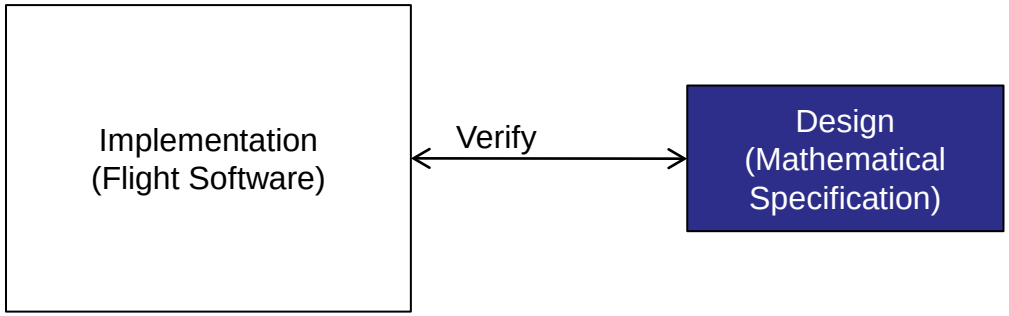
Fragment of Preliminary Hazard Analysis

PHA is ongoing

LIKELIHOOD	SEVERITY	RISK CATEGORY	MITIGATION MEASURES	CORRECTIVE ACTION	SAFETY REQUIREMENT
Remote	Major	3B	(1) Ground station pilot controller overrides autopilot (2) Failsafe autopilot intervenes when failure of autopilot detected	Verify that specification is consistent with theory	[FSP_AVCS_004] When FMS failure is detected, it is always the case that failsafe autopilot eventually takes control within a specified time duration
Probable	Major	3A		Verify correct autopilot implementation	[FSP_AVCS_003] When AP failure is detected, it is always the case that failsafe autopilot eventually takes control within a specified time duration
Remote	Hazardous	2B		(1) Verify legality of issued commands (2) Guarantee correct interpretation of commands	[A1] Commands must be interpreted correctly [A2] No command shall make the autopilot execute an unsafe maneuver.
Remote	Hazardous	2B		(1) Verify legality of issued commands (2) Guarantee correct interpretation of commands	[A1] Commands must be interpreted correctly [A2] No command shall make the autopilot execute an unsafe maneuver.

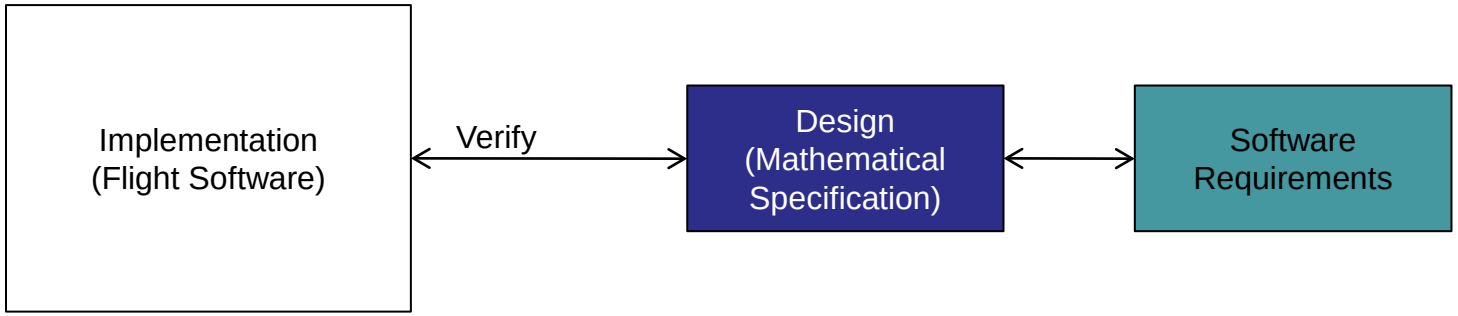
Software Verification Methodology

- Key verification component:
  - Check implementation against design



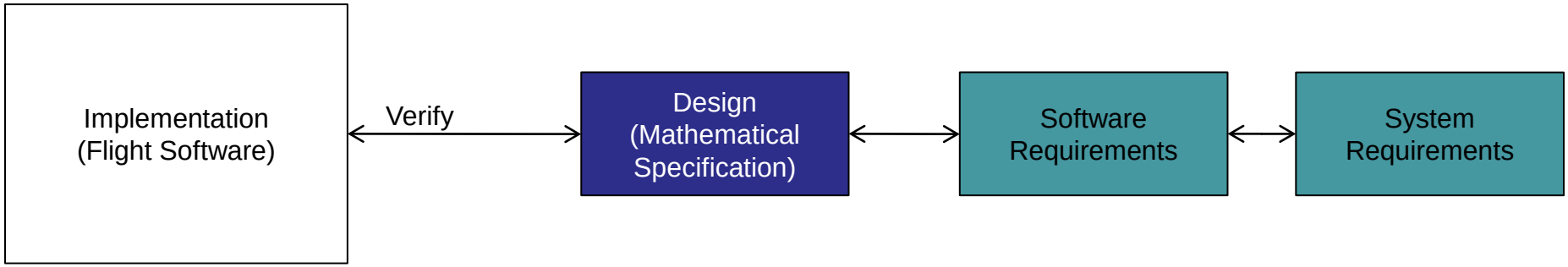
Software Verification Methodology

- Software requirements
  - Arise, in part, from software safety analysis
    - e.g., requirements on software to mitigate identified hazards
  - Constrain flight software design



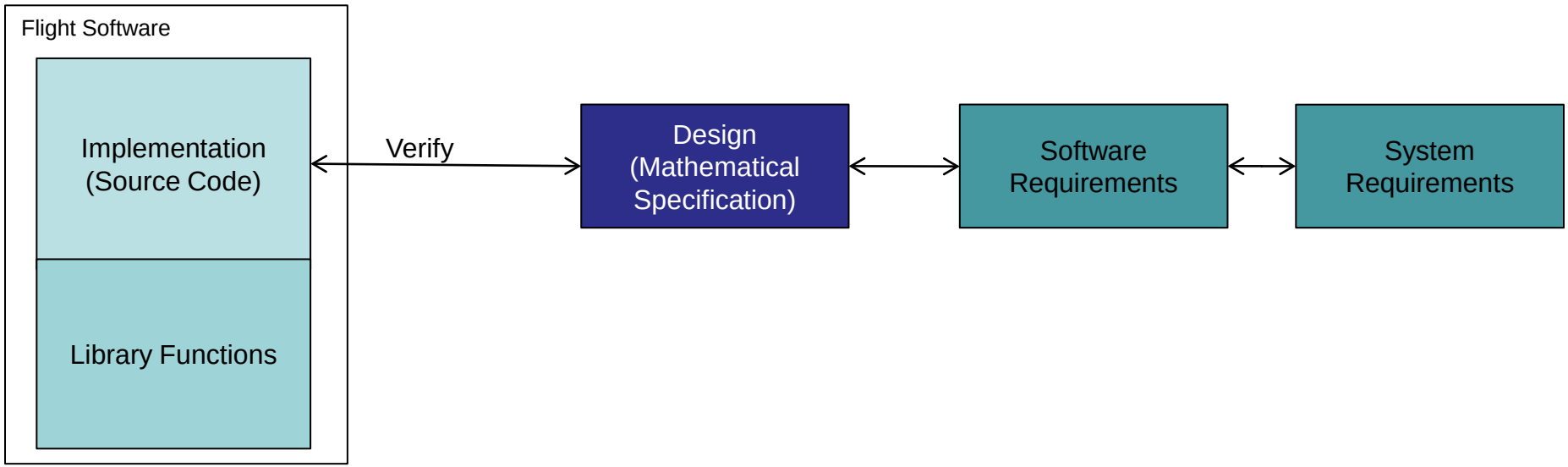
Software Verification Methodology

- System requirements
  - Arise, in part, from system safety analysis
    - e.g., hazard mitigation measures to be implemented by the system
  - Flow down to software
  - Express stakeholder needs



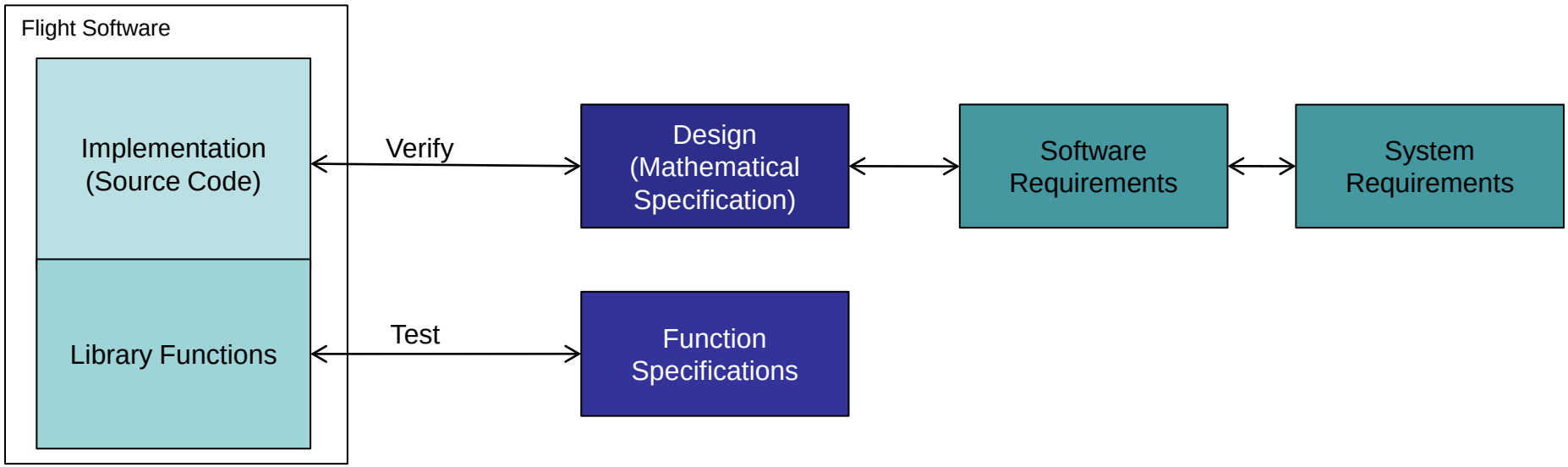
Software Verification Methodology

- Verification of flight software
  - Specifically verification of the source code
  - Against mathematical specification



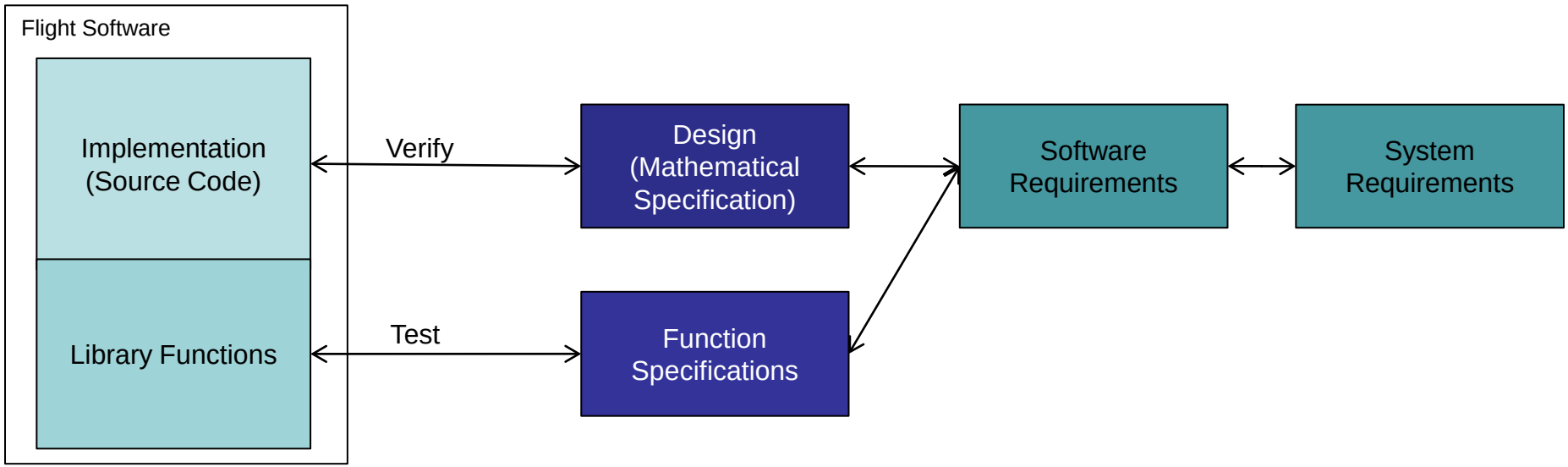
Software Verification Methodology

- Verification of flight software
  - Specifically verification of library functions
  - Testing against function specifications



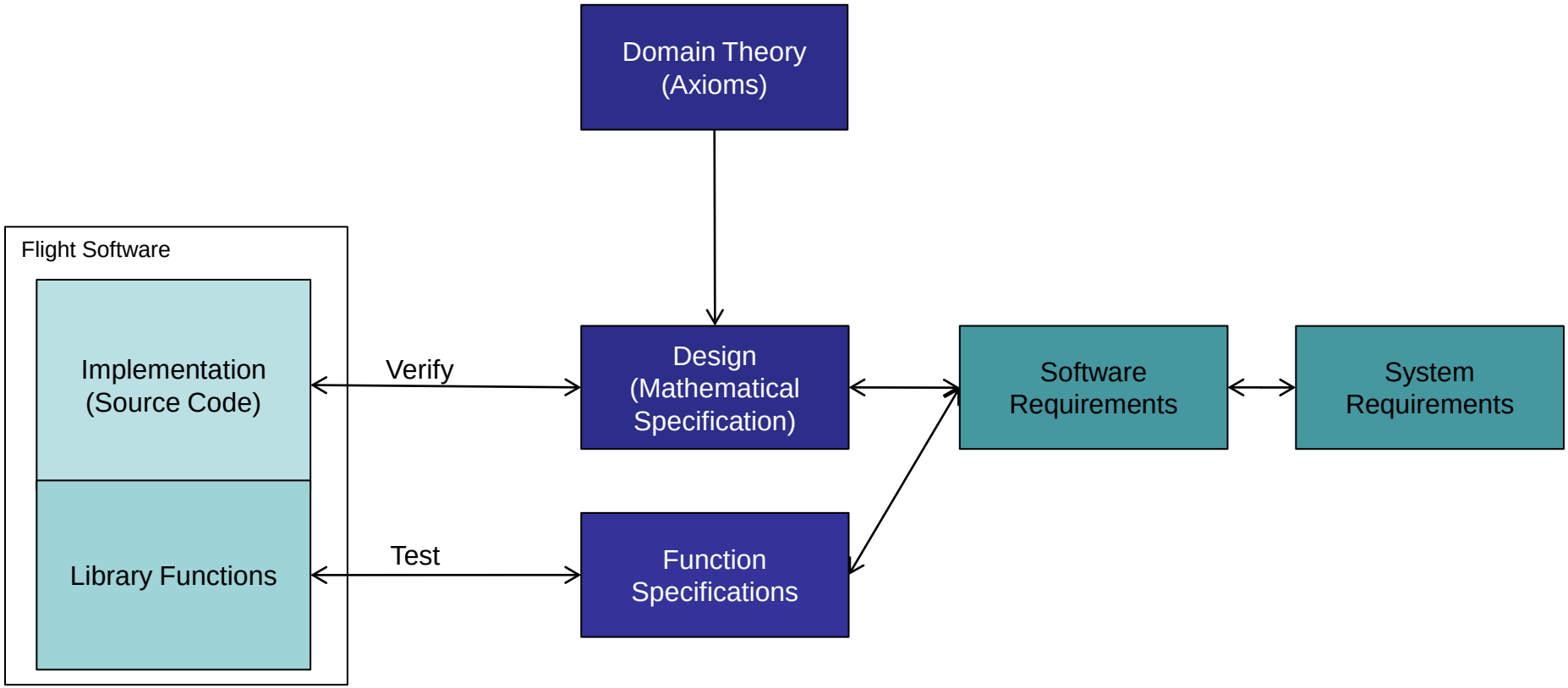
Software Verification Methodology

- Function specifications derived from software requirements



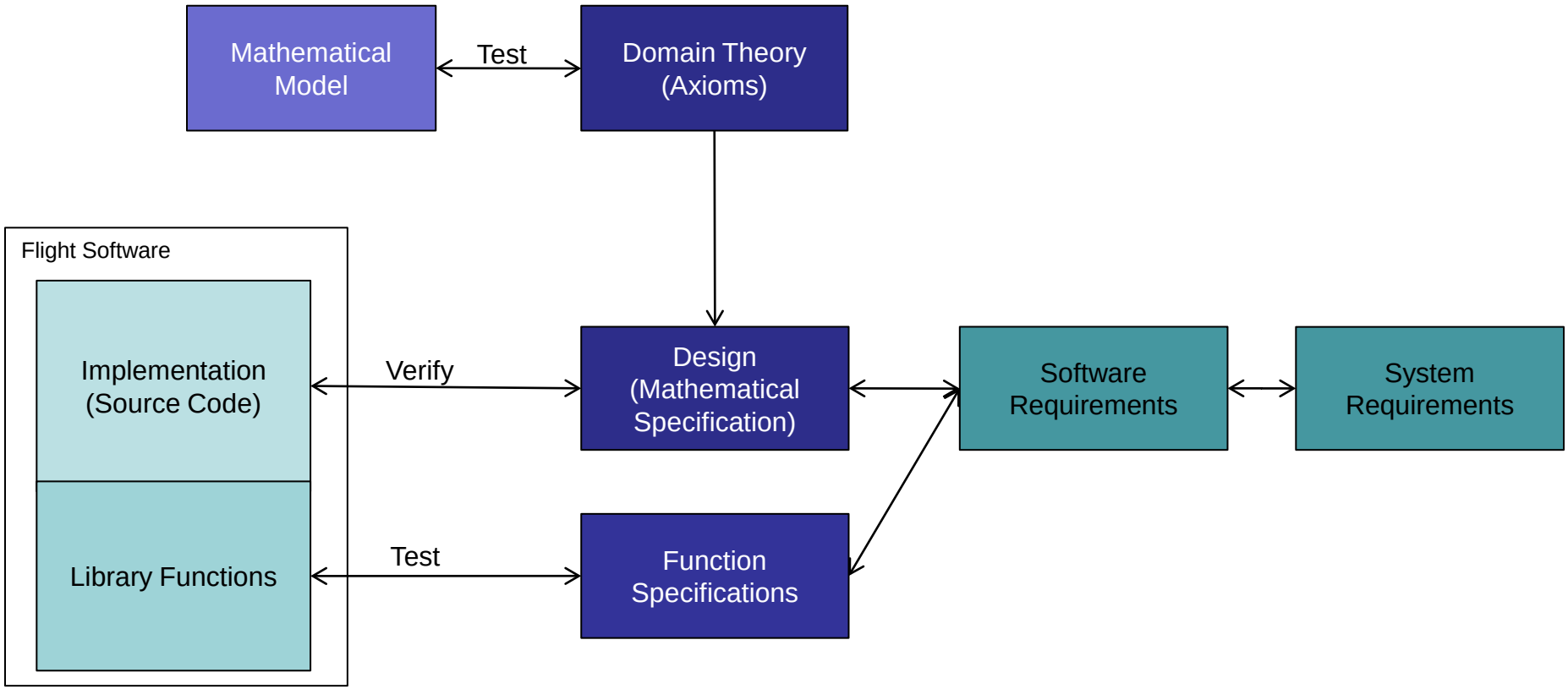
Software Verification Methodology

- Design defined within the context of a domain theory



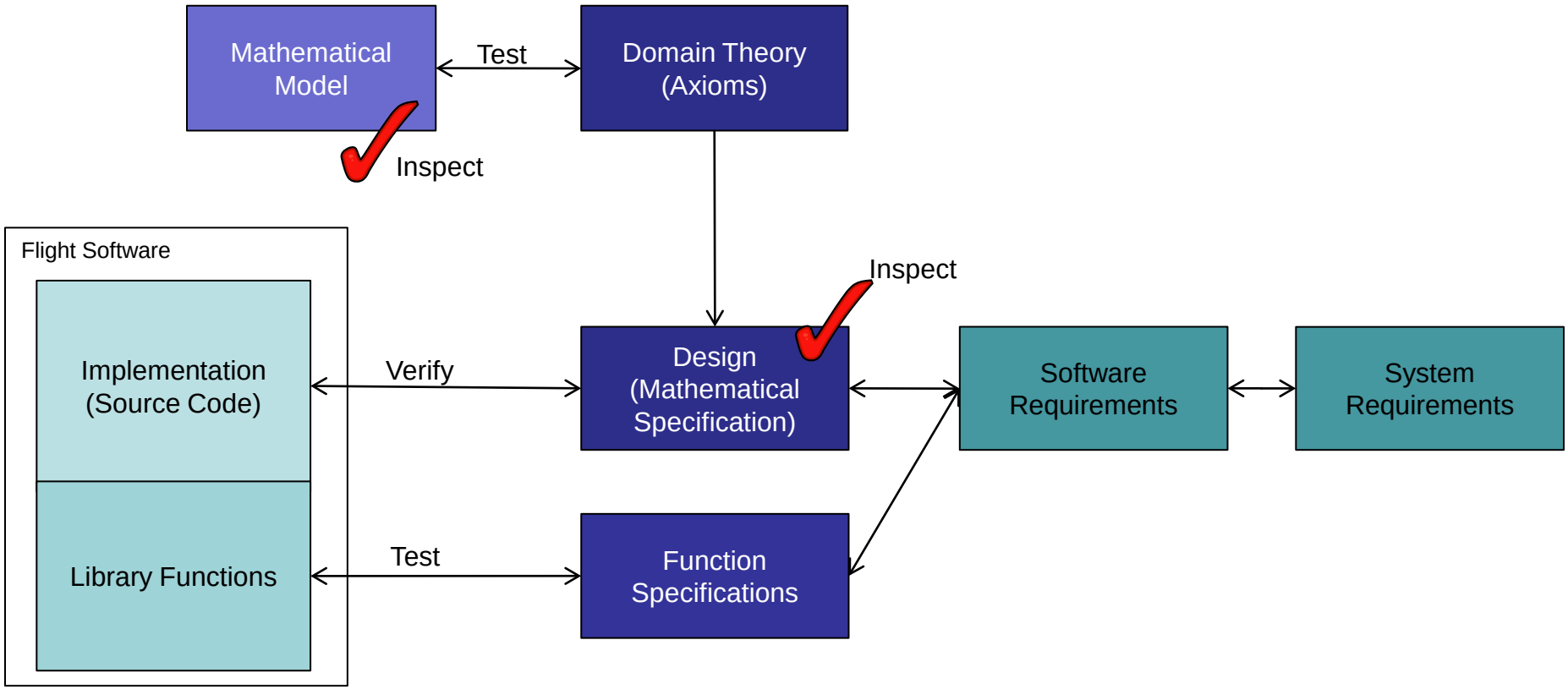
Software Verification Methodology

- Verification of domain theory against mathematical models (physics)
  - Via testing.



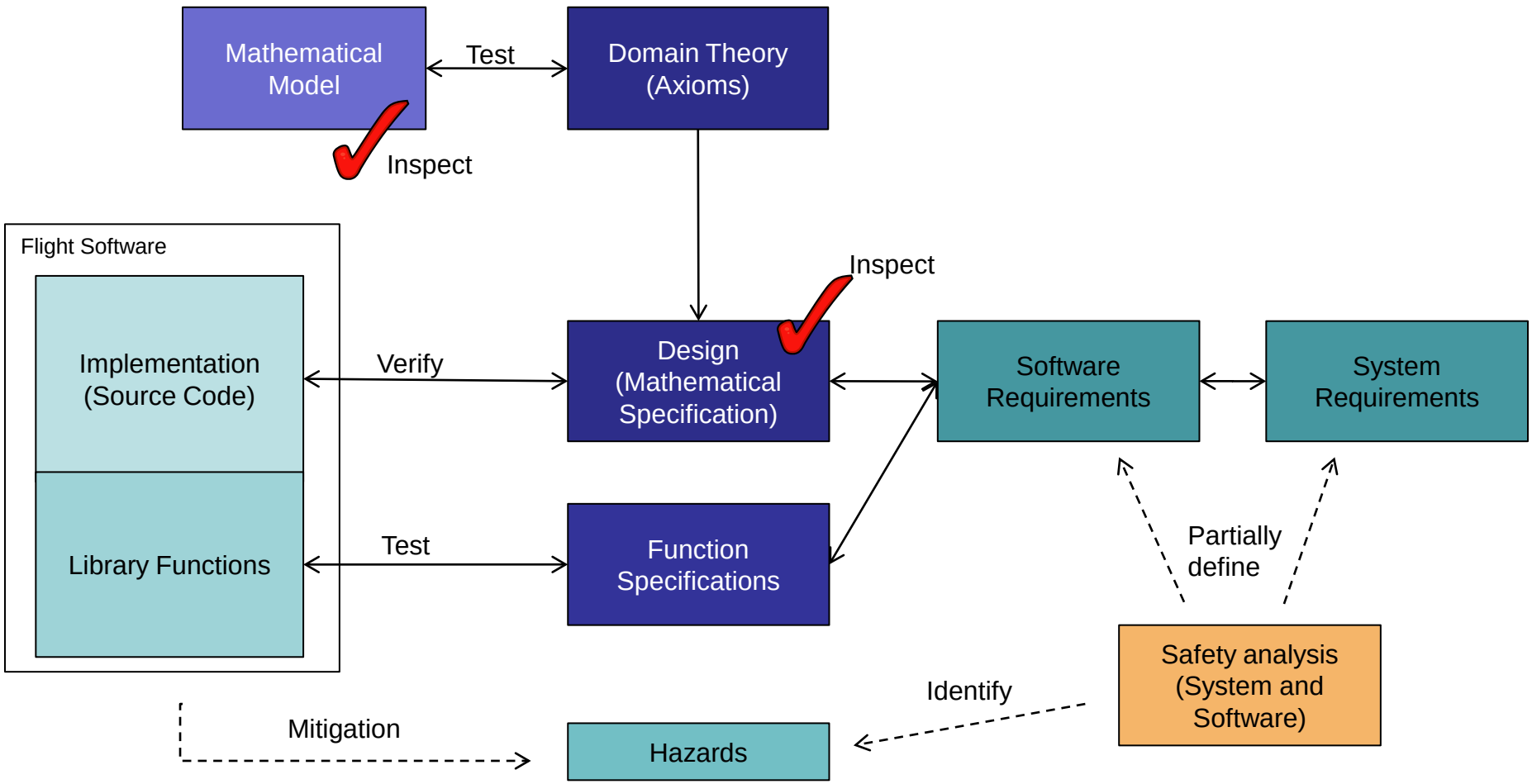
Software Verification Methodology

- Verification of mathematical models / specification via inspection



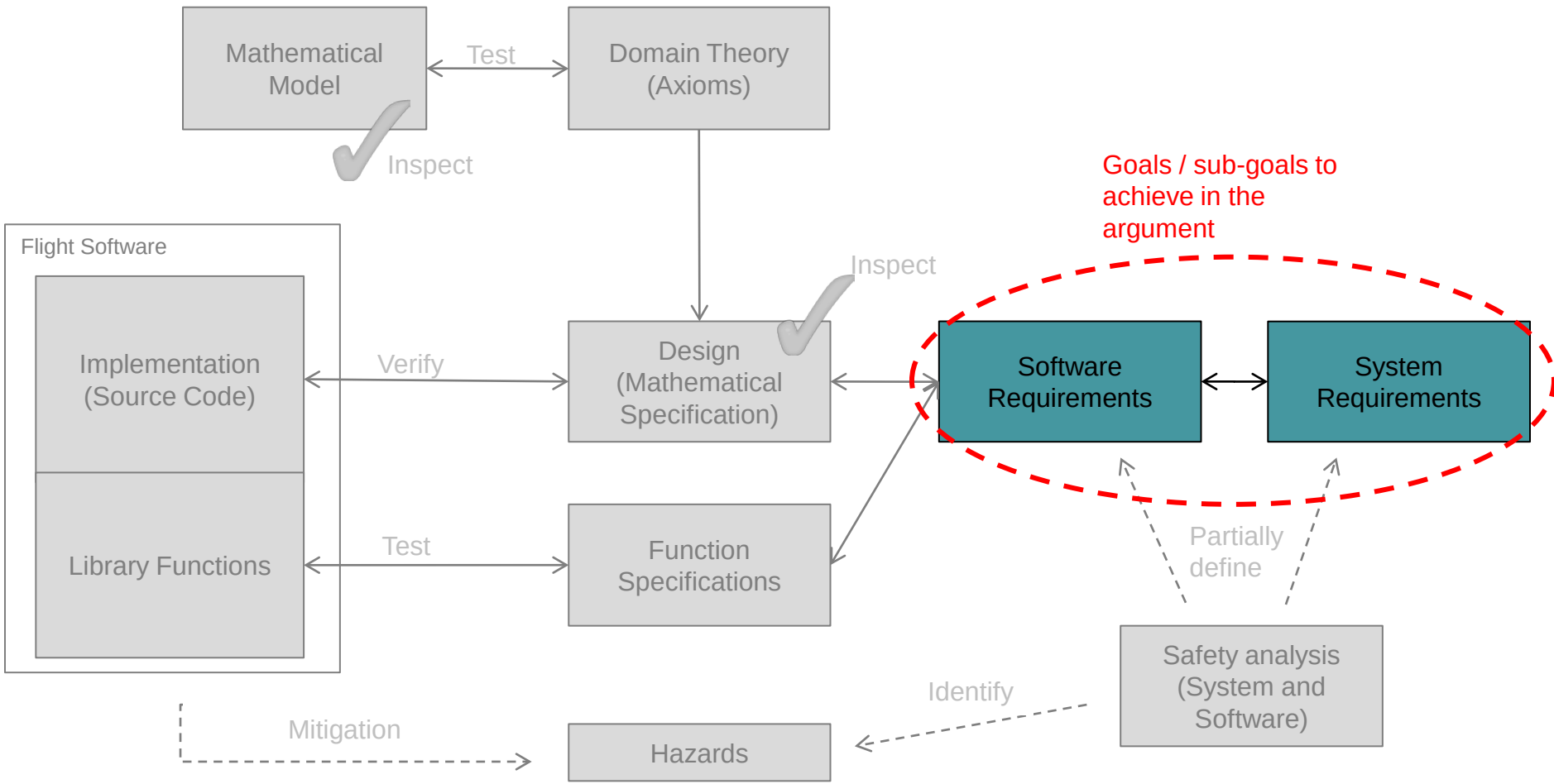
Software Verification Methodology

- The role of verification in the context of safety analysis



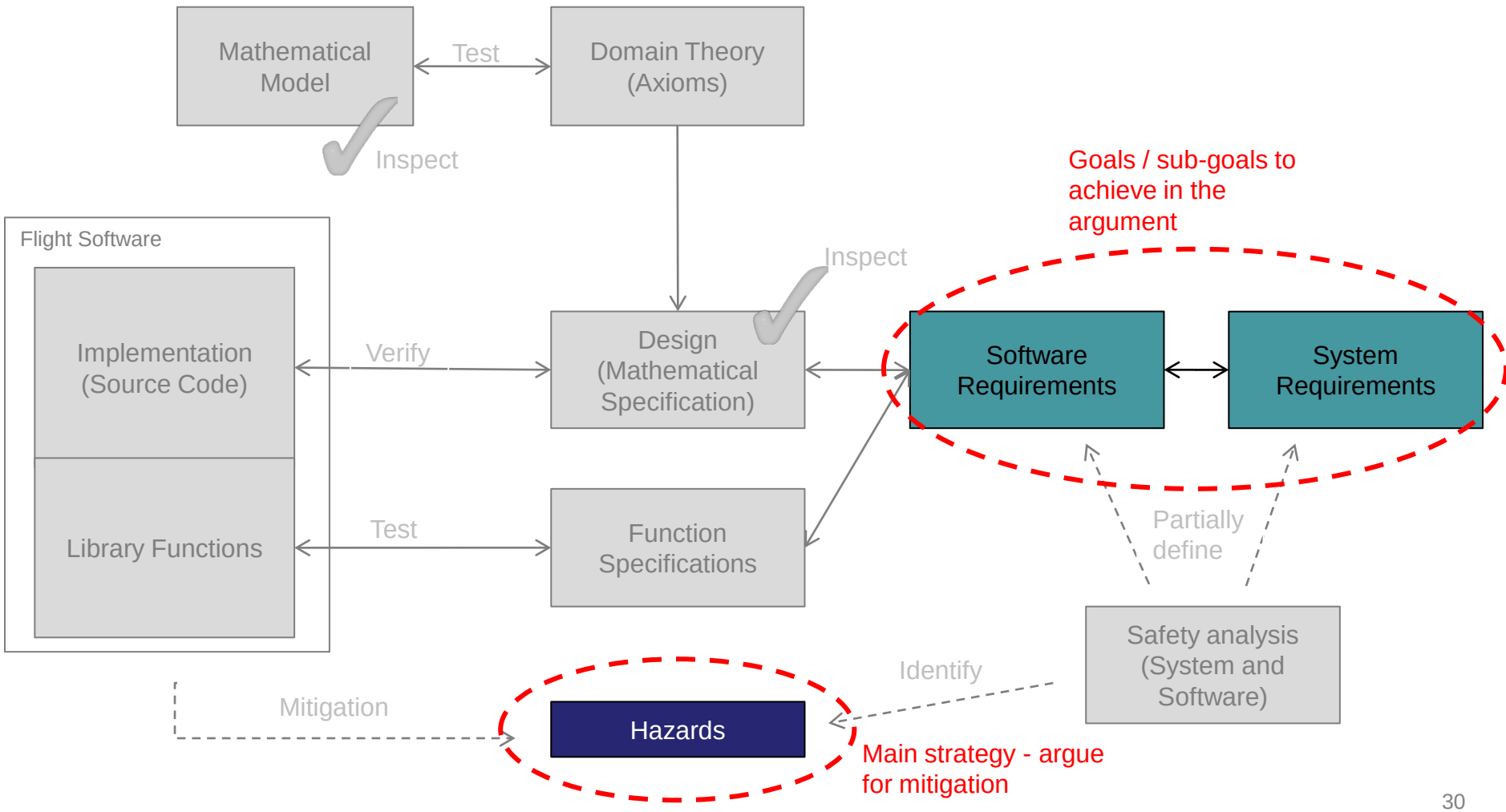
Software Verification Methodology

- The role of verification in the context of safety *arguments*

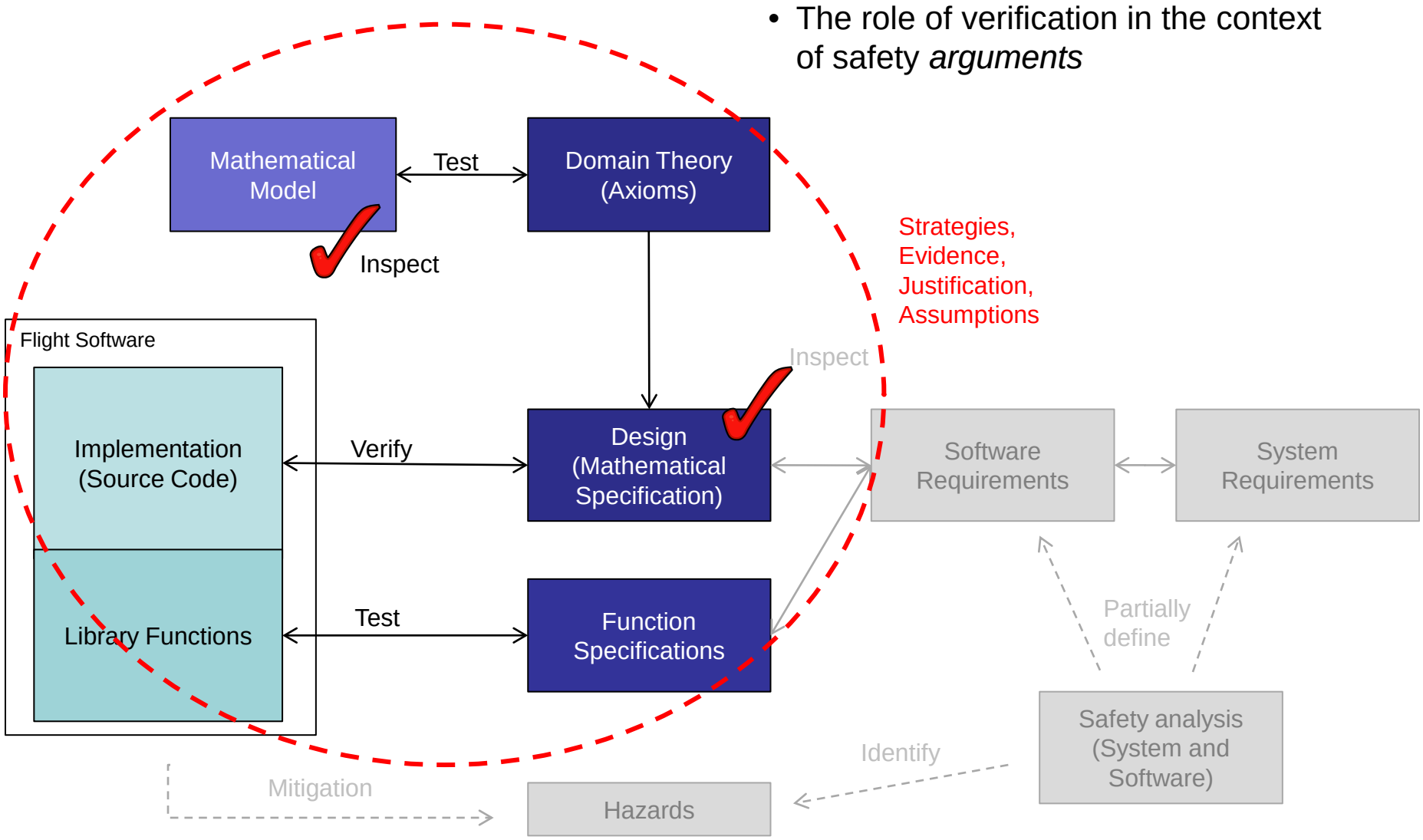


Software Verification Methodology

- The role of verification in the context of safety *arguments*

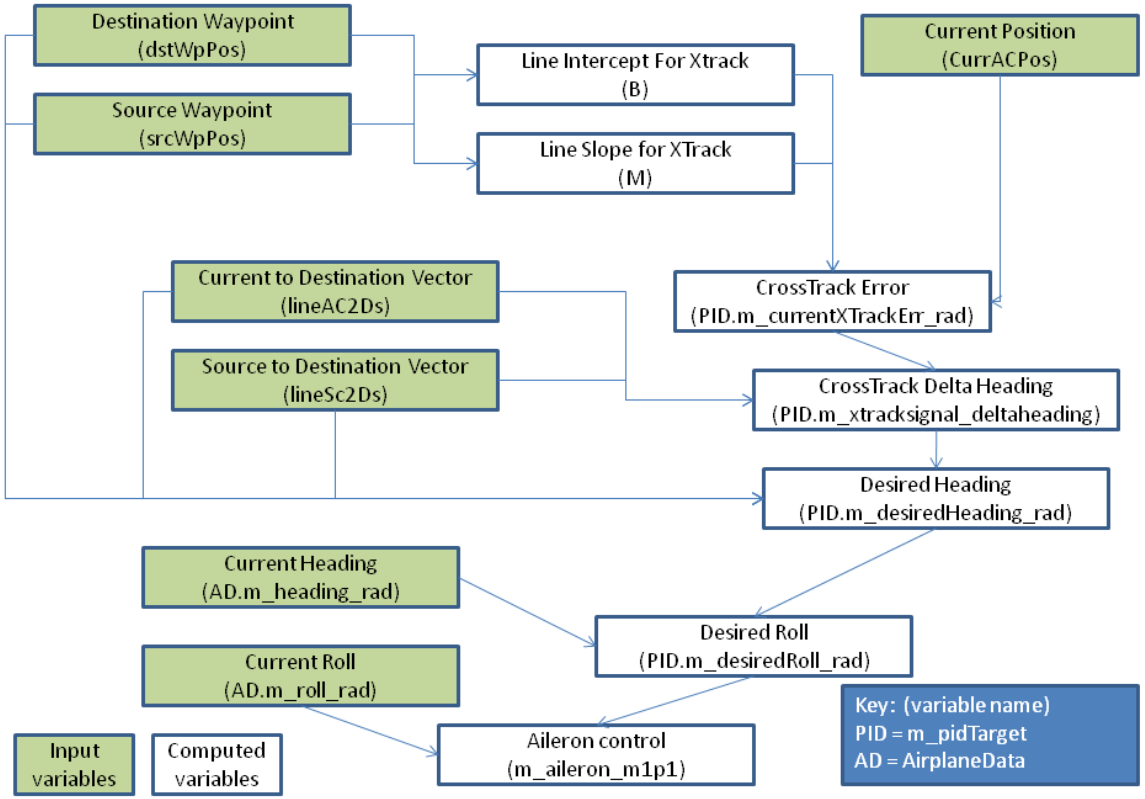


Software Verification Methodology



AutoCert

- Analyses source code using theorem proving
- Domain theory defined using math equations
- Generates chain of reasoning from assumptions to requirements
- Traces between code, documentation and V&V artifacts



Regulatory Framework

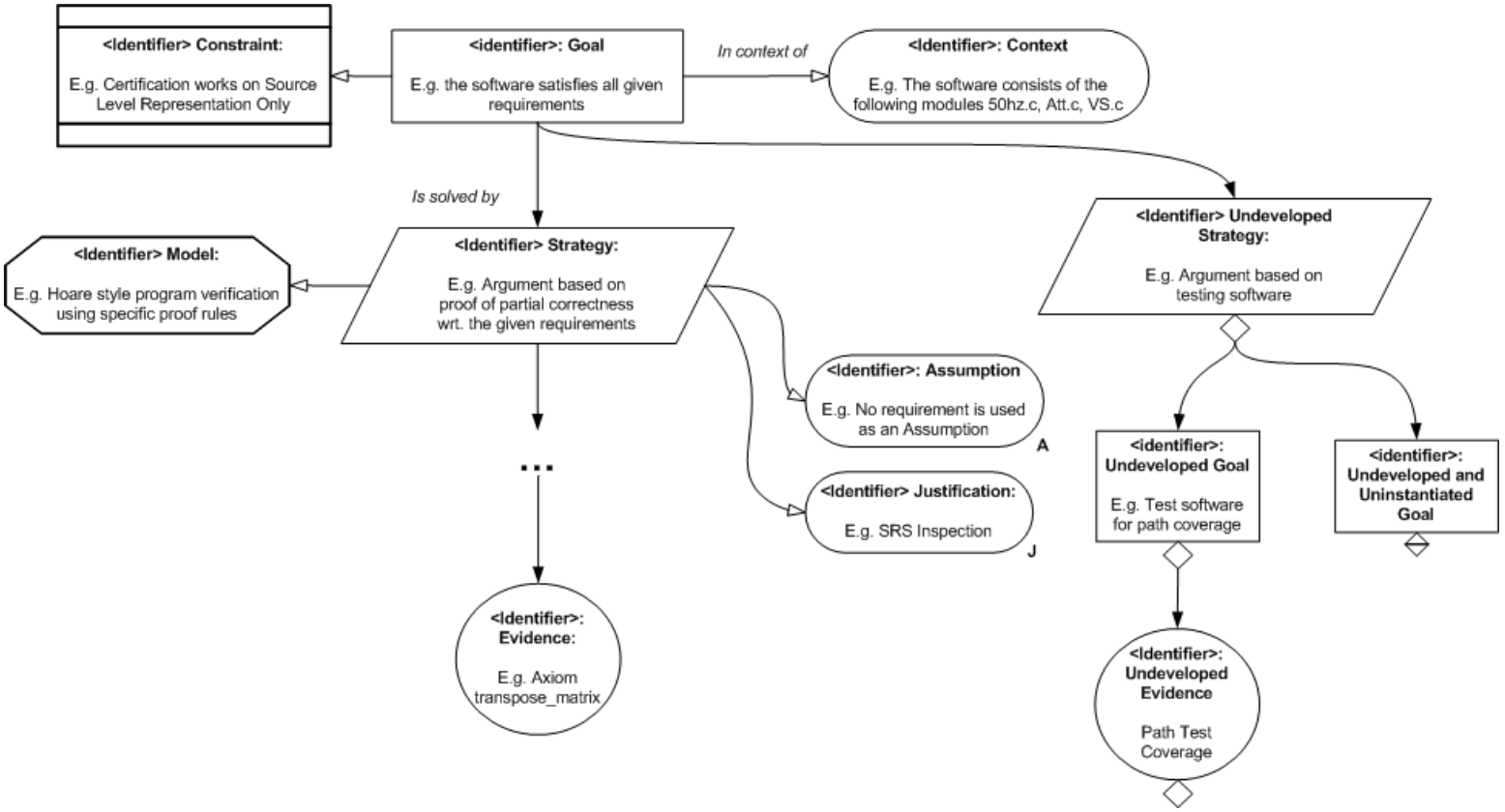
- NPR 7150.2A - NASA Software Engineering Requirements
- NPR 7123.1A - NASA Systems Engineering Processes and Requirements
- NPR 7900.3B - Aircraft Operations Management Manual
- NASA-GB-8719.13 - NASA Software Safety Guidebook
- NASA-STD-8719.13 - NASA Software Safety Standard
- NASA-STD-8739.8 - Software Assurance Standard
- NPR 8715.5A - NASA Range Flight Safety Program
- NPR 8715.3C - NASA General Safety Program Requirements
- NPR 8705.5A - Probabilistic Risk Assessment (PRA) Procedures for Safety and Mission Success for NASA Programs and Projects
- APR 8705.1 - System Safety and Mission Assurance (NASA Ames)

A safety case should be aligned with the relevant regulations

→ Justifies how evidence generated from compliance with regulations supports claims about software and system safety.

Safety Argumentation

- Document safety cases using Goal structuring notation (GSN)*



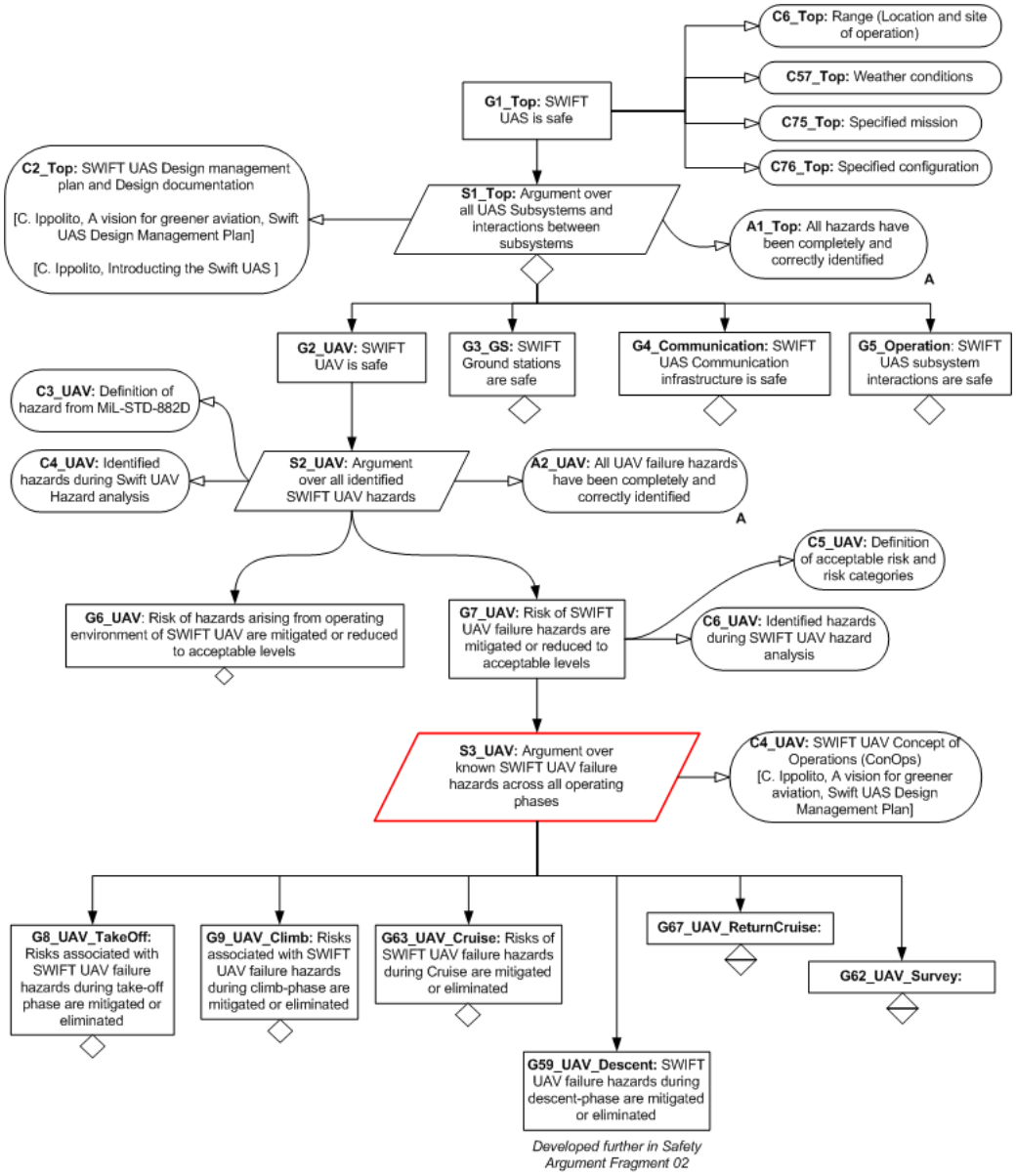
* Tim Kelly. Arguing Safety: A Systematic Approach to Managing Safety Cases. PhD thesis, University of York, 1998

Safety Case Fragments – Swift UAS

- Options
  - (1) Argue (mitigation / elimination of hazards) over phases of operation
    - Take off, Climb, Cruise, Survey, Return-cruise, Descent, Land.
    - Need to address safety of transitions between phases
    + *Addressing hazards which change risk categories across phases*

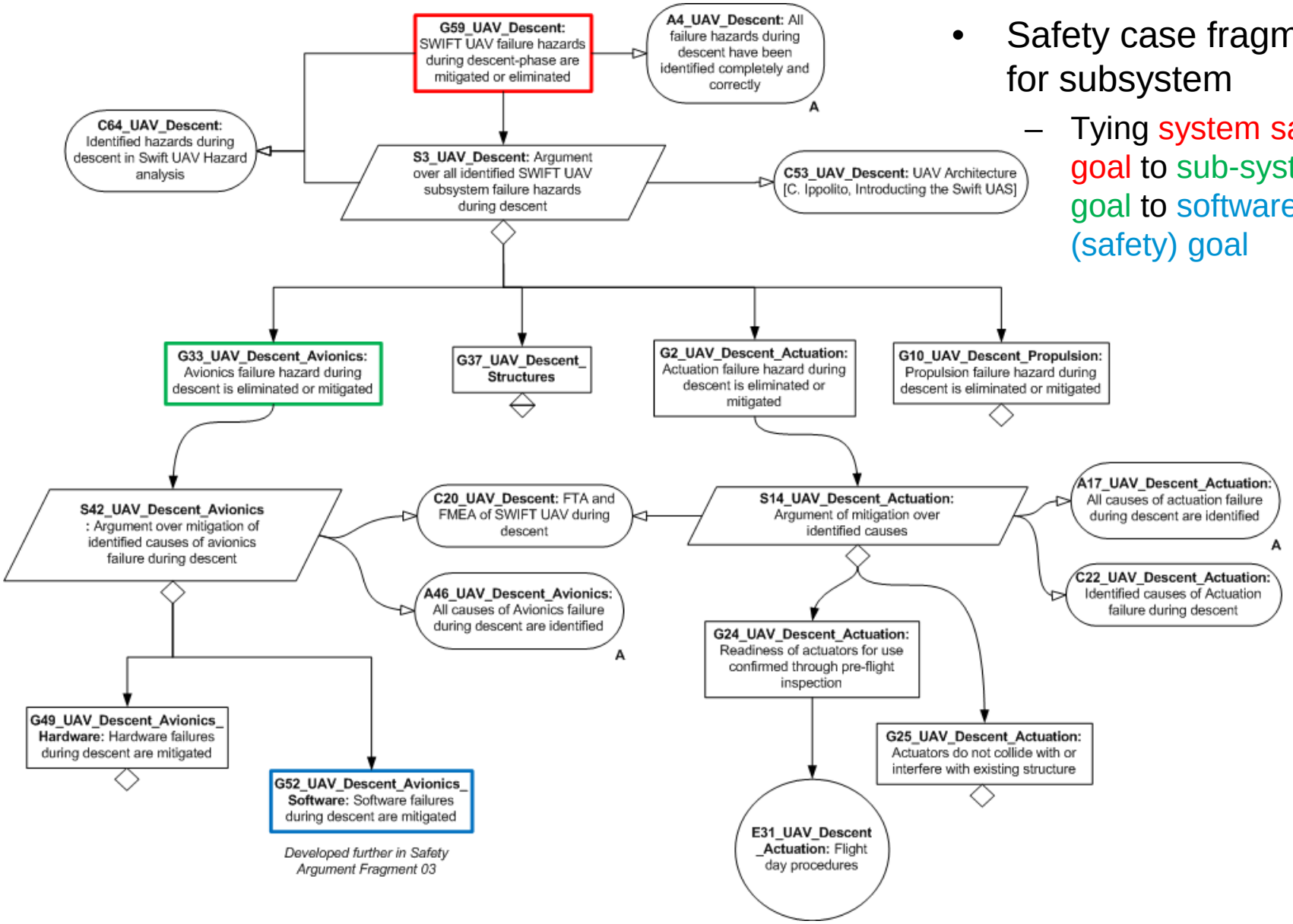
  - (2) Argue (mitigation / elimination of hazards) over system architecture
    - Airborne system (Swift UAV)
      - Actuation
      - Propulsion
      - Avionics (HW / SW)
      - Contingency management
      - Power system
      - Structure
    - Ground system
    - Communication infrastructure
    - Need to address safety of interactions between systems
    + *Maintainable and Modular argument*

Safety Case Fragments – Swift UAS



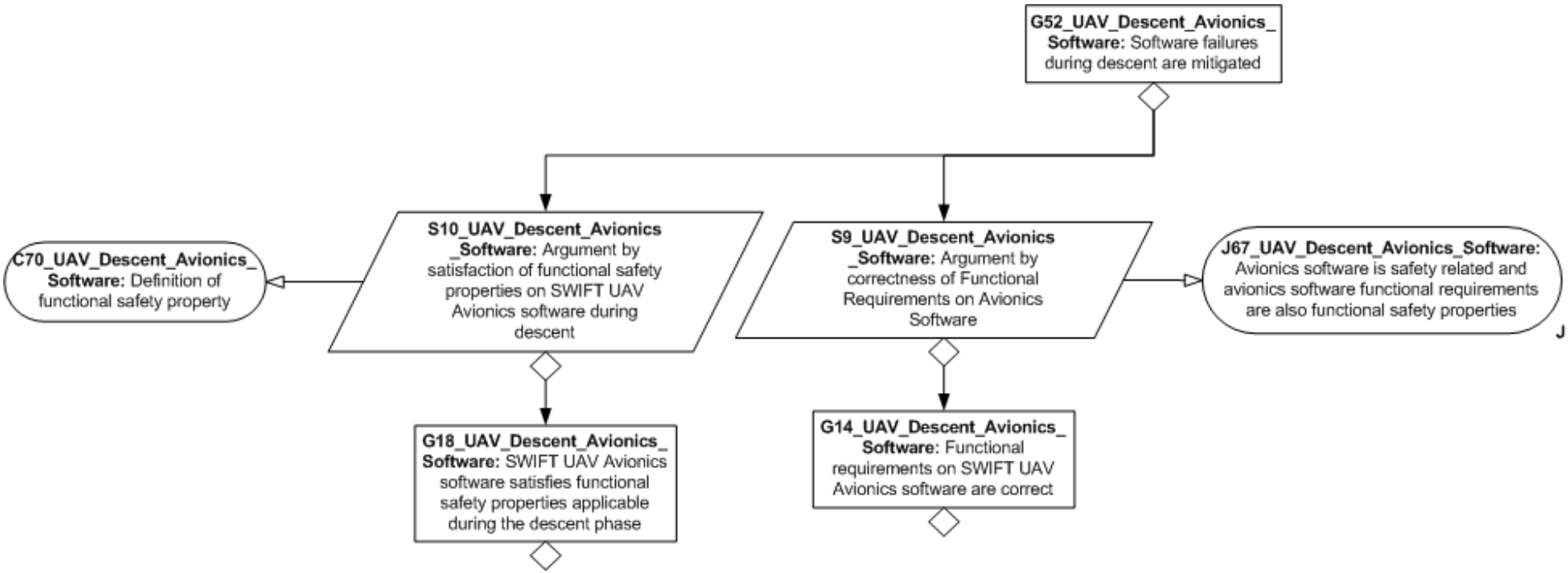
- System-level safety case <u>fragment</u>
  - Using Option (1) for exemplification
  - Slice of overall safety argument

Safety Case Fragments – Swift UAS



- Safety case fragment for subsystem
  - Tying system safety goal to sub-system goal to software (safety) goal

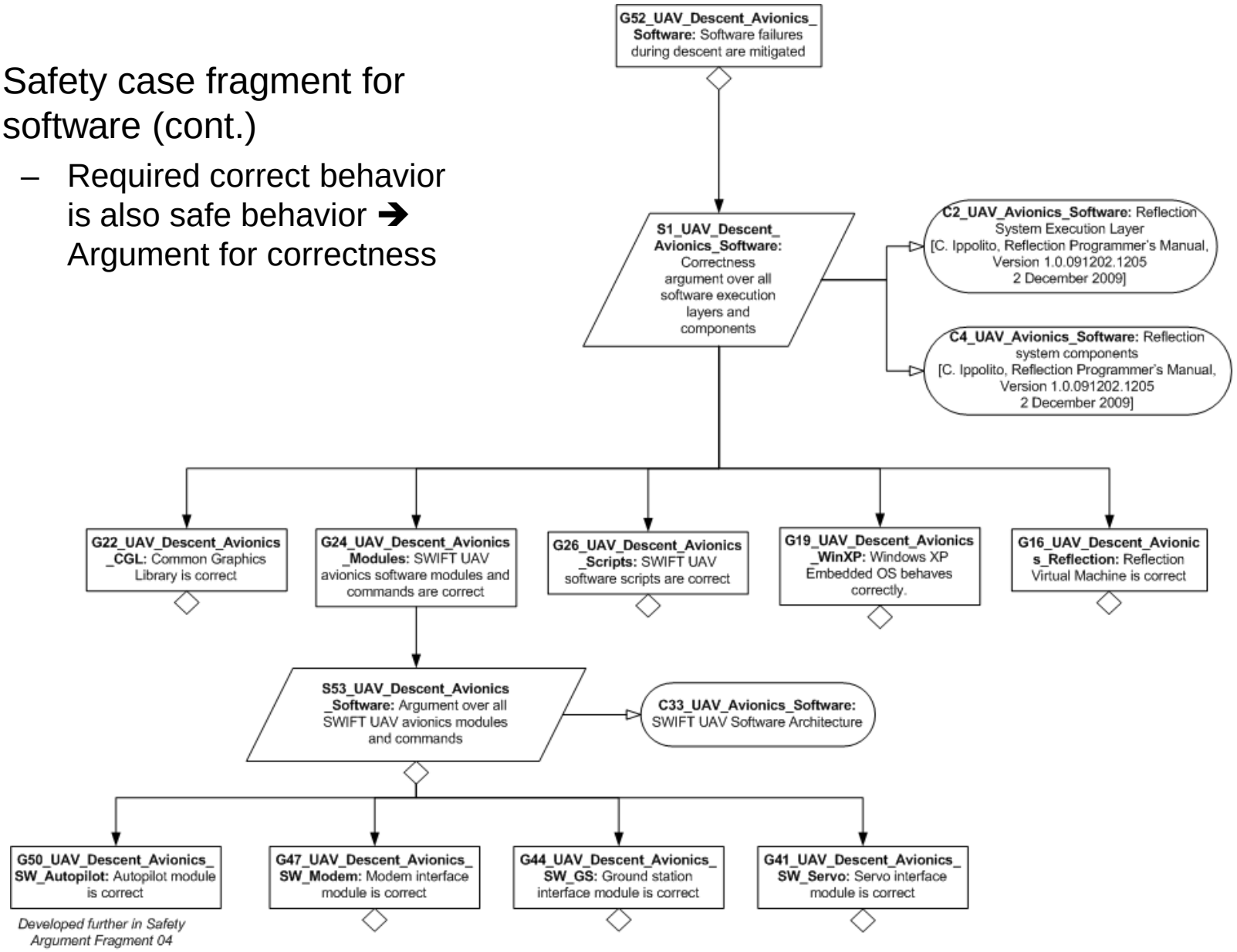
Safety Case Fragments – Swift UAS



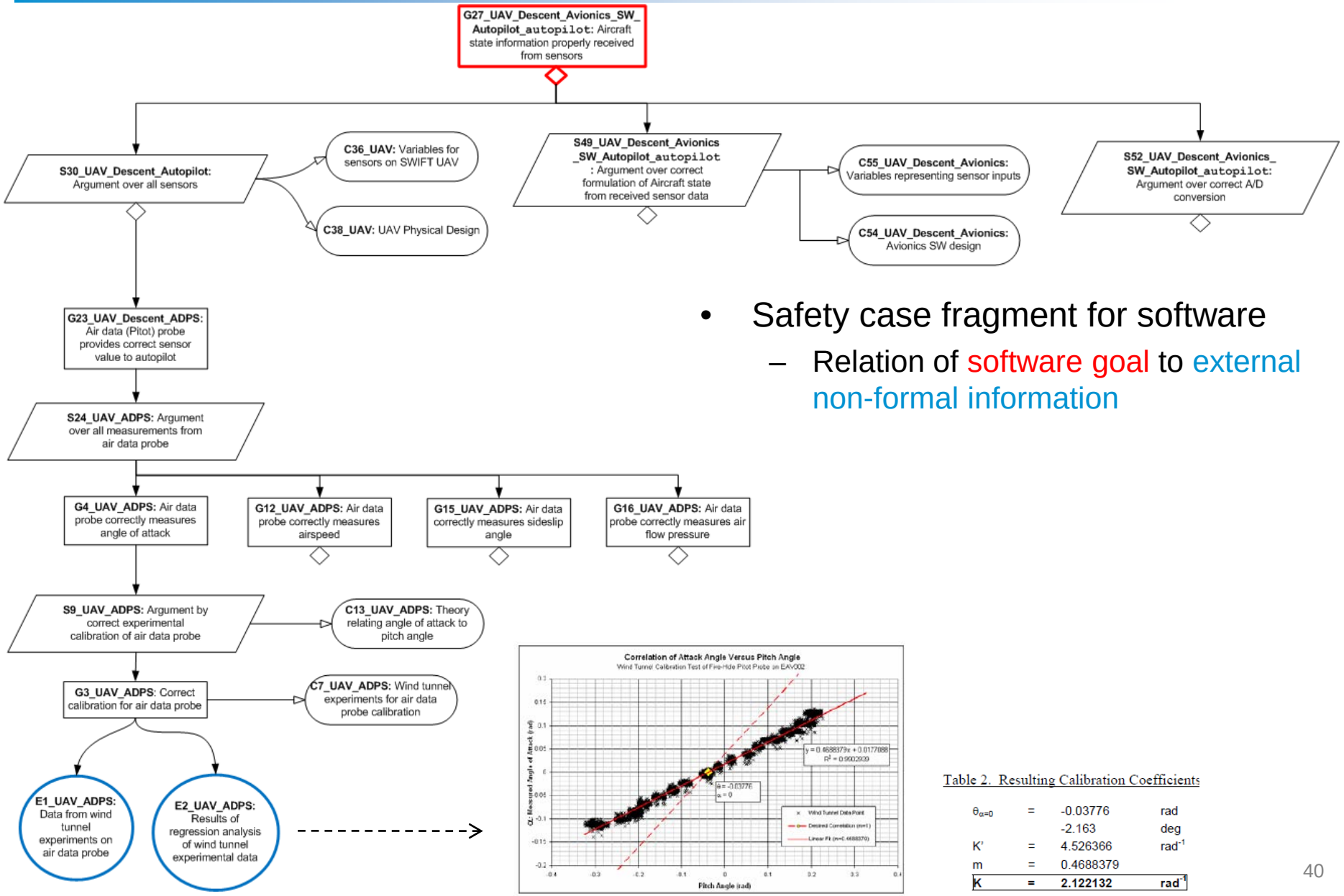
- Safety case fragment for software
  - Autopilot is safety relevant, failsafe autopilot is safety critical (HazAn)
  - Mitigation of autopilot / failsafe autopilot failure hazard → Safe behavior
  - Required correct behavior is also safe behavior
  - Strategy used: argue correct behavior of autopilot / failsafe autopilot

Safety Case Fragments – Swift UAS

- Safety case fragment for software (cont.)
  - Required correct behavior is also safe behavior ➔ Argument for correctness



Safety Case Fragments – Swift UAS

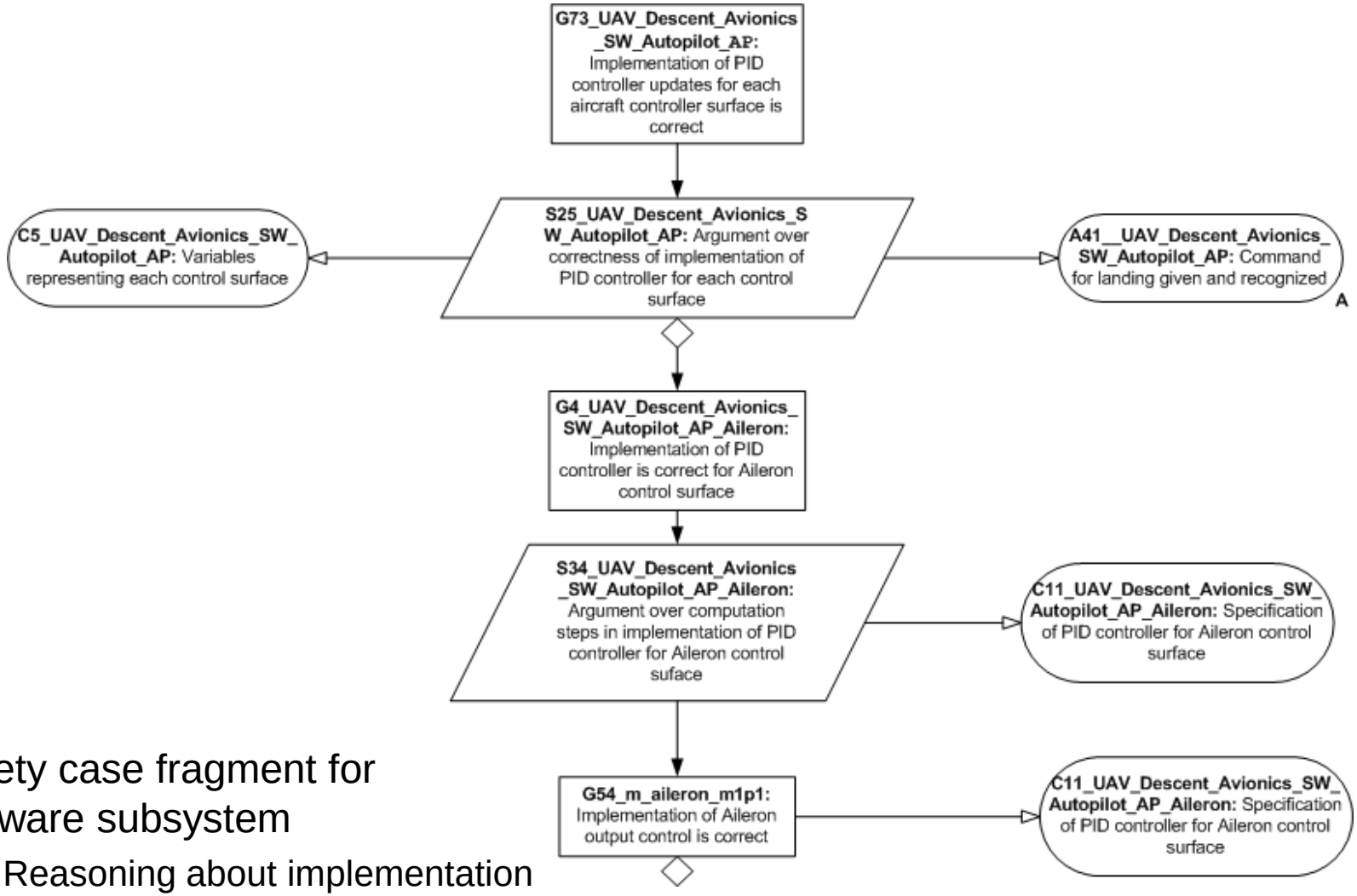


- Safety case fragment for software
  - Relation of software goal to external non-formal information

Table 2. Resulting Calibration Coefficients

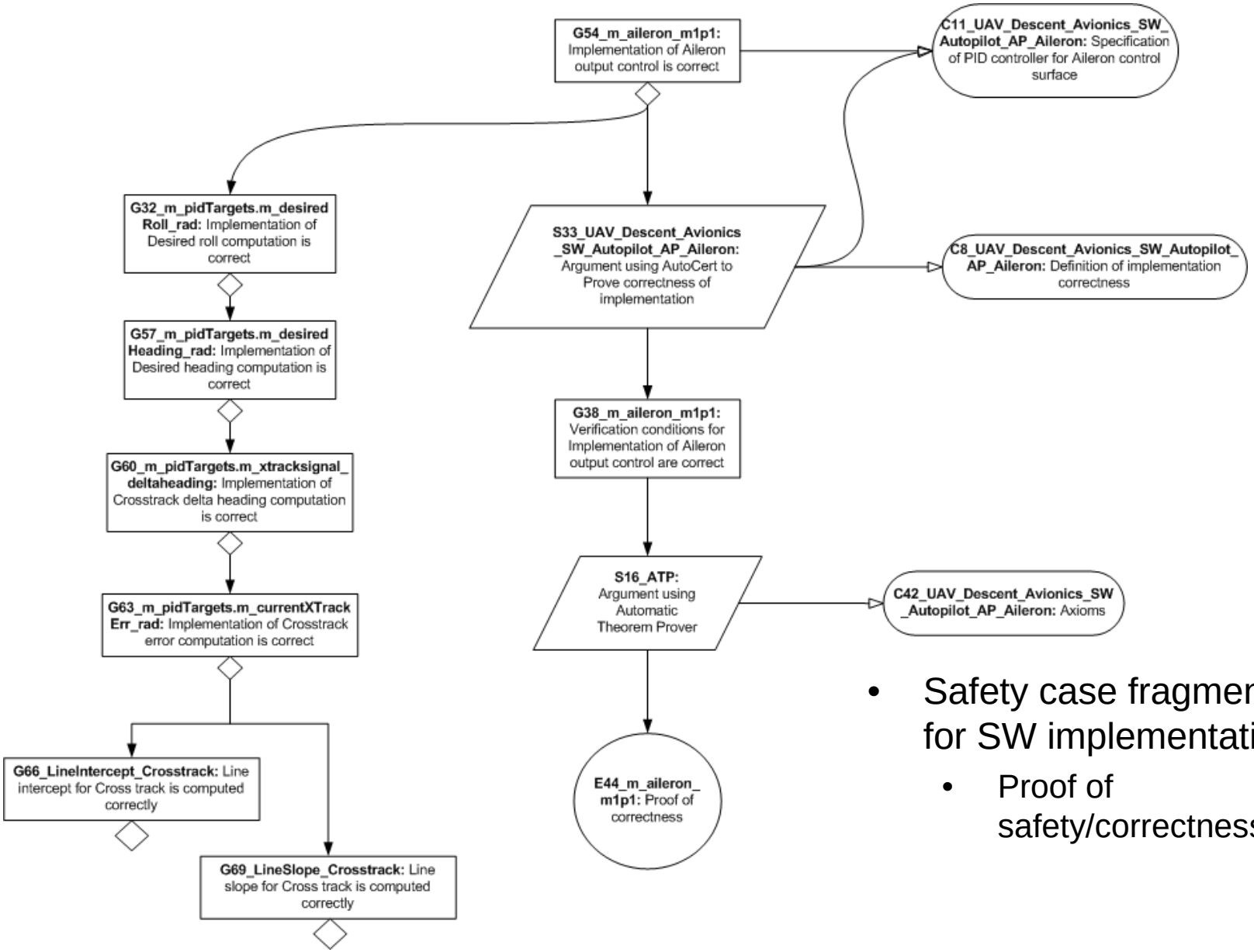
$\theta_{\alpha=0}$	=	-0.03776	rad
		-2.163	deg
K'	=	4.526366	rad^{-1}
m	=	0.4688379	
K	**=**	**2.122132**	**rad^{-1}**

Safety Case Fragments – Swift UAS



- Safety case fragment for software subsystem
  - Reasoning about implementation

Safety Case Fragments – Swift UAS



- Safety case fragment for SW implementation
  - Proof of safety/correctness

Summary

- Safety case provides assurance for safety based on:
  - Hazard identification and mitigation
  - System/software boundary
  - Tracing from high to low-level requirements
  - Integration of formal and non-formal analyses
  - Correspondence of formal analyses to case fragments
  - Combination of top-down and bottom-up analyses
- Concentrate on airworthiness
  - Later: operations, NAS
- Next:
  - Allow control of “design choices” in safety case
  - Mark-up language for evidence
  - Include output of different tools
  - Map to regulatory framework
  - Incorporate tool qualification
  - Queries and views
  - Probabilistic reasoning