

Behavioral Cross Check method

DATE: 28/10/2022

Summary

This document aims to explicit and to explain the designed method called “Behavior Cross Check” (BCC). This method is introduced by the top document framed by the S2C project.

This method is based on the concept of “Cross Check of observations” done on the simulation results of two distinct models. The document includes the description of the proposed method, two proof of concept (PoC) based upon SE and SA models to experiment and validate the method and the conclusions of the method.




<i>Author(s)</i>	<i>Function(s) & name(s)</i>	<i>Research engineers IRT Saint Exupéry</i>	<i>S. Guilmeau</i> 
<i>Checker(s)</i>	<i>Function(s) & name(s)</i>	<i>Head of Systems Engineering Centre of Competence IRT Saint Exupéry</i>	<i>J. Baclet</i> 
<i>Approver</i>	<i>Function & name</i>	<i>Project leader IRT Saint Exupéry</i>	<i>J. Perrin</i> 

Table of Contents

Evolutions

Table of figures

Table of tables

1 Introduction

1.1 Purpose of document

6

1.2 Referenced documents

1.2.1 S2C reference documents

Title	Reference

1.2.2 External reference documents

Title	Reference

2 Objectives, constraints and Context

2.1 Objectives

-
-
-

2.2 Constraints

2.2.1 Common constraints

PROJECT CONFIDENTIAL

2.3 Context

2.3.1 Common Context

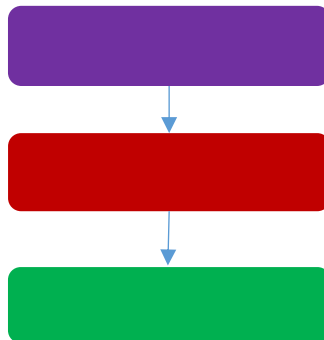
2.3.2 Specific Context

3 Developed method

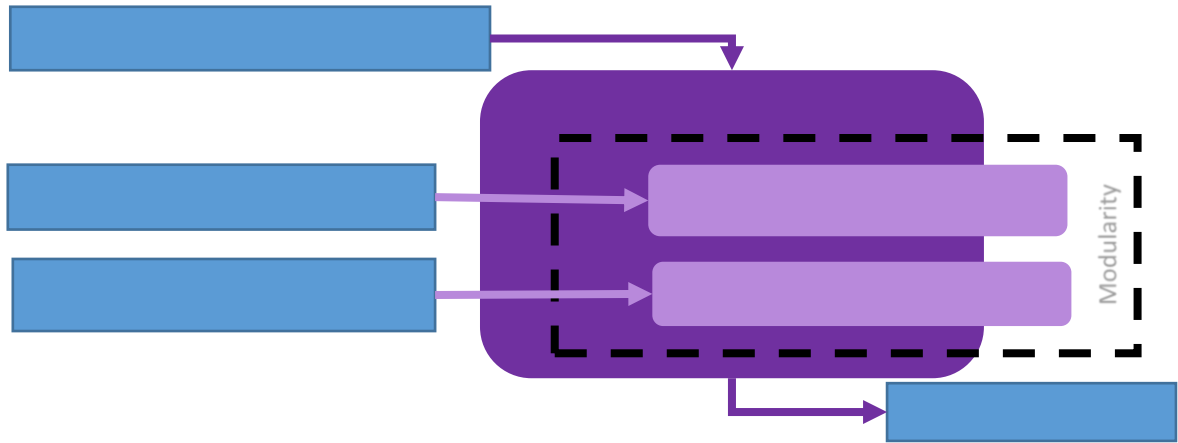
3.1 BCC Process

-
-

-
-
-



3.1.1 Scenario initialization



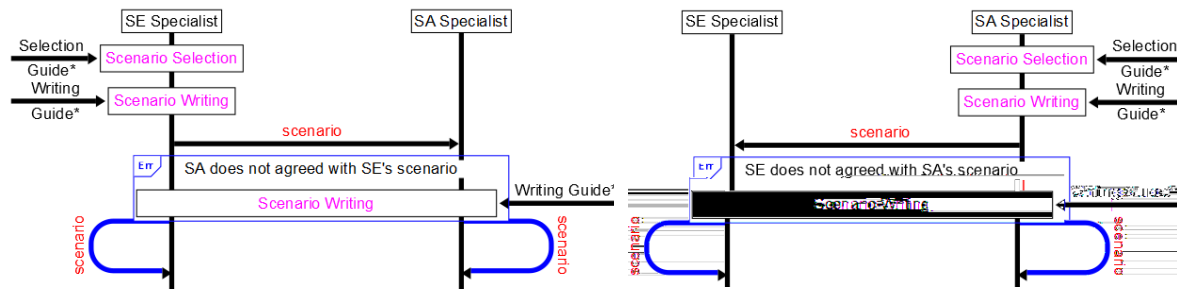
(a) Scenario selection

				#id-d	#id-e	#id-f	#id-g			#id-j	#id-k	#id-l	#id-m	#id-n			
	✓	█					✓	✓	✓	✓	✓	✓	✓	✓	✓		█

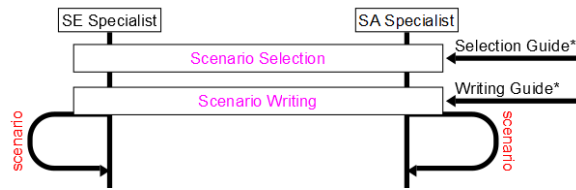
PROJECT CONFIDENTIAL

(c) Modularity

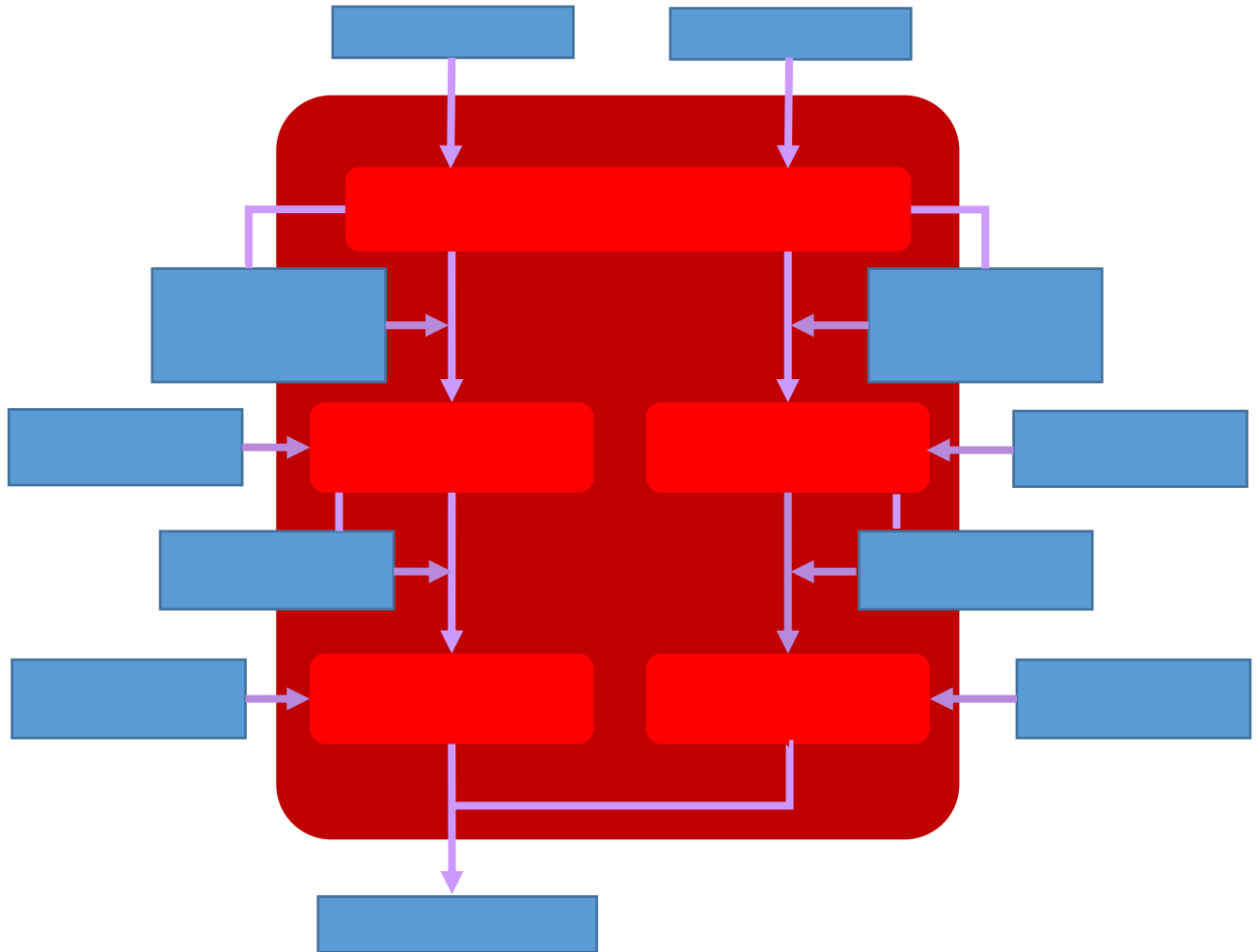
-
-
-



-



3.1.2 Scenario exploitation



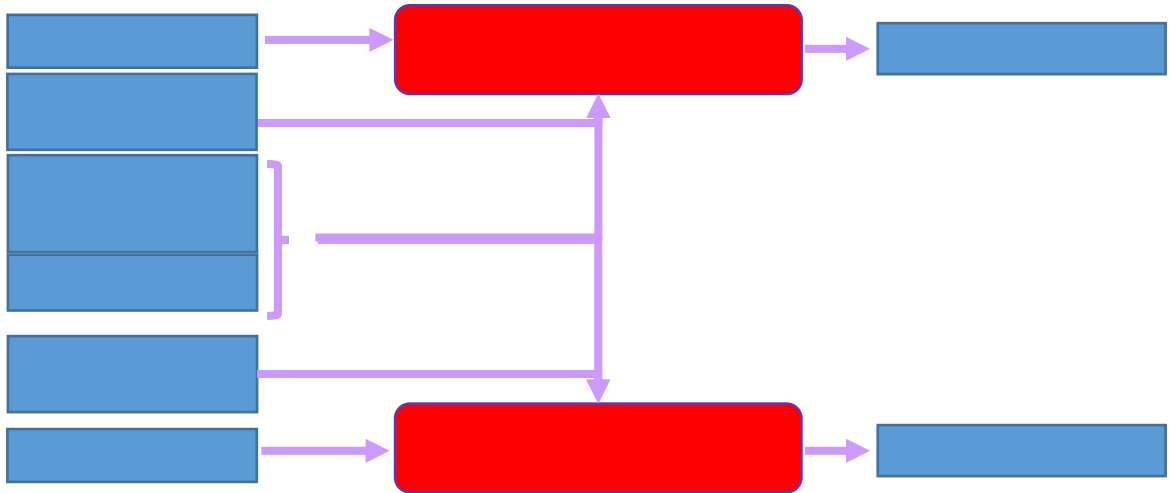
(a) Upstream coordination between Scenario Concepts vs. Domain Variables



-
-
-
-
-

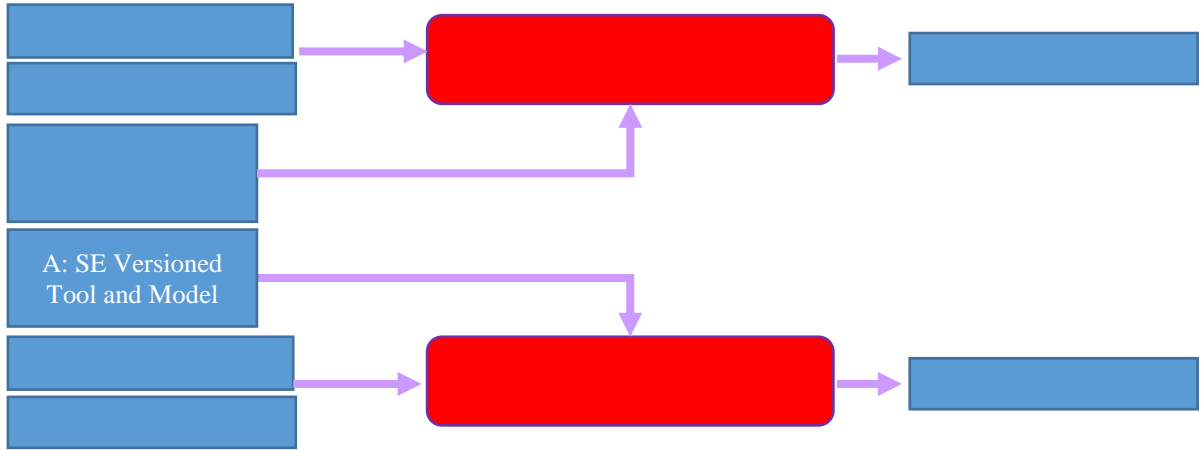
	✓		✓	✓	✓		✓	✓		✓	✓	✓		✓	✓	✓

(b) Transformation from Scenario to Procedure



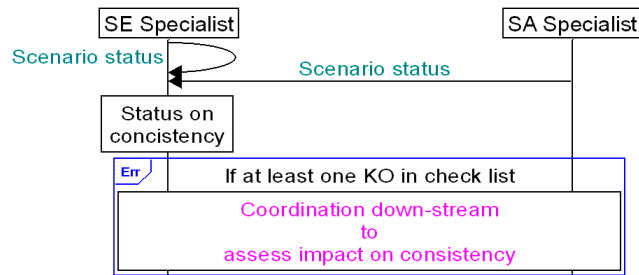
	✓		✓		✓			✓		✓	✓	✓		✓	✓	

(c) Procedure execution



	✓	✓	✓	■	✓	■	✓	✓	■	✓	✓	✓	■	✓	✓	■

3.1.3 Scenario conclusion



	✓		✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	

3.2 Intra-process validation activities

3.3 Iterations

-

-

4 Deductible facts before PoCs are done

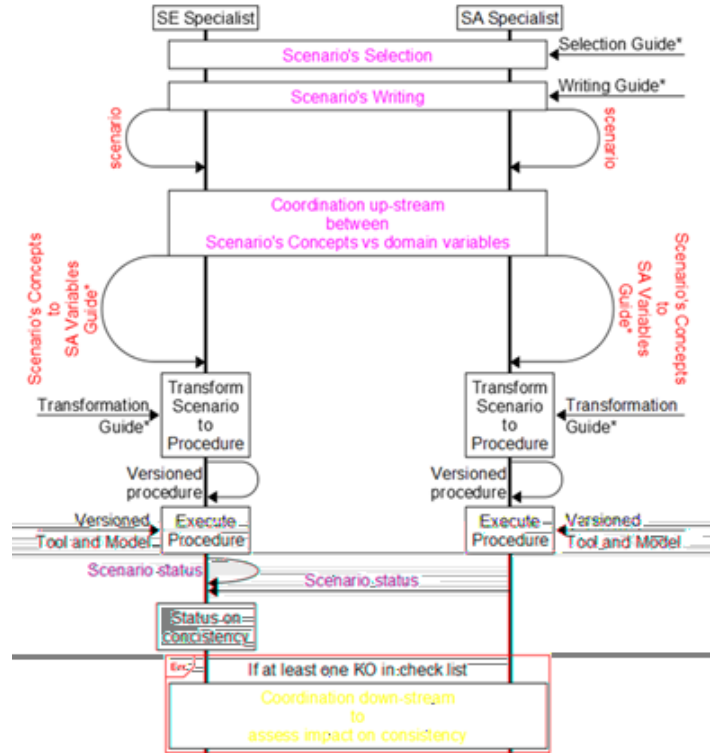
5 PoC activities

•
•

#sq-0

SE&SA propose together...

... then specialists work



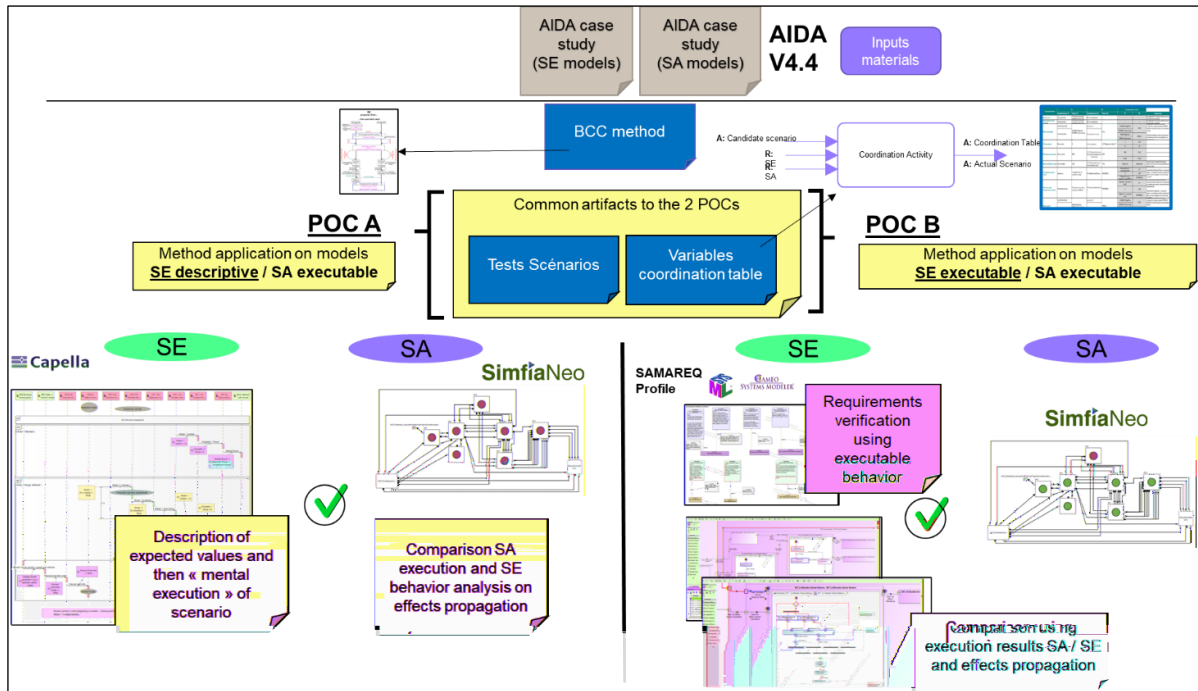
5.1 Proof-of-concept

5.1.1 Preliminary input material

5.1.2 PoCs overview

PoC A

PoC B



5.1.3 Common activity 1– Scenario Initialization

Scenario initialization

(a) Scenario 1: Loss of motor

(b) Scenario 2: Loss of Attitude Information

(c) Scenario 3: Position Information Erroneous

(d) Scenario 4: Switch From Auto to Manual Mode

5.1.4 Common activity 2 - Upstream coordination between Scenario Concepts vs. Domain Variables

Scenario Concept	SE		SA		Value's domain correspondance	
	Variable/observer SE	SE Values	Variable/observer SA	SA Values	SE	SA
Drone in flight	Drone real position	XYZ position vector value included in specific shape	N/A: Voir commentaires	-	-	-
Drone insied authorized area	Drone real position	XYZ position vector value included in specific shape	N/A: Voir commentaires	-	-	-
Automatic Mode	Control Mode Internal	AUTOMATIC Flight Plan AUTOMATIC Speed Consign	Sortie SF4.3.2	AUTO	AUTOMATIC Flight Plan	AUTO
	Control Mode Pilote		EXT_PilotDetection.PilotSelectedMode		MANUAL	
Loss of motor	Motor x motion	0	SF12_CreateMotion	LOST (failure mode "fail_loss")	0	LOST
					!=0	OK / ERRONEOUS
Motor loss detection	Motor x runaway	TRUE	SF7.SF3_MonitorParameters.SF734_MonitorDroneMotors.Motro1Runaway	TRUE	TRUE	TRUE
					FALSE	FALSE
Shut off motor lost	Motor x disabled	TRUE	SF7.SF73.MonitorParameters.MotorsDisabled_MotorX	TRUE	TRUE/FALSE	TRUE/FALSE
Consequence motor loss	Global thrust	!= Required Thrust +/- acceptance margin	SF1GlobalThrustAndTorque	ERRONEOUS	= Required Thrust +/- acceptance margin	OK
					0	LOST
Pilot detection abnormal behavior	Sensed drone position	XYZ position vector value à comparer avec flight plan	SF2.DetectControlDronePosition	ERRONEOUS	!= Required Thrust +/- acceptance margin AND !=0	ERRONEOUS
					= flight plan +/- acceptance margin	OK
					0	LOST
Manual Mode	Control Mode Internal	MANUAL Flight Plan MANUAL Speed Consign	Sortie SF4.3.2	MANUEL	AUTOMATIC Flight Plan	AUTO
	Control Mode Pilote		EXT_PilotDetection.PilotSelectedMode		MANUAL Flight Plan	
Loss of drone control	Global thrust	!= Required Thrust +/- acceptance margin	SF1GlobalThrustAndTorque	ERRONEOUS	= Required Thrust +/- acceptance margin	OK
					0	LOST
Propeller off	Propeller x thrust	0	SF1SF11_ControlHelix1.SF13_CreateThrustAndTorque.output	LOST	= Required Propeller Thrust +/- acceptance margin	OK
					0	LOST
					!= Required Propeller Thrust +/- acceptance margin AND !=0	ERRONEOUS
Motor shutdown power	Motor Shutdown	TRUE	SF1SF11_ControlHelix1.SF16_DepowerMotor	TRUE	TRUE	TRUE
					FALSE	FALSE

SSR method

5.1.5 Common activity 3 – Transformation from Scenario to Procedure

Transformation from Scenario to Procedure

(a) Example of SE Verification procedure

6. Test principle

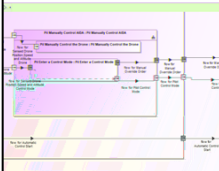
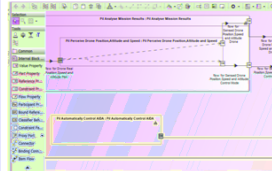
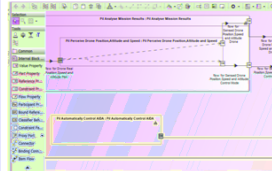
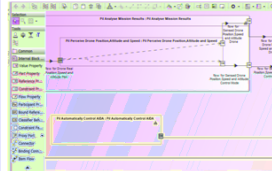
- Launch MBSE simulation environment
- Initialize scenario preconditions (start scenario until UAV is starting inspection and is in the allowed area around the aircraft)
- Execute simulation step by step
- Observe variables and simulation results
- Stop the MBSE Simulation environment

7. Preconditions

Drone is in flight in allowed area (drone real position in range “allowed area”) and has started to follow an inspection plan.
 Drone is in AUTOMATIC mode
All equipments behave normally – modes machines are not in degraded modes

8. Test procedure – steps

a. AIDA-XX YYY-SA-001

<p>on until UAV is around the aircraft</p> <p>ed area</p> 	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%;">Action:</td> <td>Open MBSE tool and start standard simulation in the allowed area</td> </tr> <tr> <td>Expectation:</td> <td>Drone real position is included in the allowed area Drone follows the flight plan Control Mode = AUTOMATIC</td> </tr> <tr> <td>Result:</td> <td>  </td> </tr> </table>	Action:	Open MBSE tool and start standard simulation in the allowed area	Expectation:	Drone real position is included in the allowed area Drone follows the flight plan Control Mode = AUTOMATIC	Result:	
Action:	Open MBSE tool and start standard simulation in the allowed area						
Expectation:	Drone real position is included in the allowed area Drone follows the flight plan Control Mode = AUTOMATIC						
Result:							

(b) Example of SA Verification procedure

This is not a

current SA practise to find such writing test activities

11. → Test-procedure¶

a. → AIDA-V443.001-SA-001¶

Action:¶	Look-for-function-"SF1i2_CreateMotion"-in-your-left-menu-and-trigger-its-failure-mode-"fail_loss".-Double-click-on-"fail_loss"-failure-mode-to-trigger-it-on-the-simulation.-Failure-modes-are-nested-under-the-function-in-the-Model-Explorer-left-menu.-¶
Expected-results:¶	Failure-mode-"fail_loss"-is-triggered-on-the-simulation.-The-failure-mode-once-is-triggered-it-should-have-disappeared-on-the-menu.-On-the-layout,failed-components-are-displayed-in-red-colour.-¶
Observed-result:¶	"fail_loss"-failure-mode-does-not-appear-any-more-for-the-function-"SF1i2_CreateMotion".-Many-variables-have-changed-its-status-after-triggering-the-failure-mode,some-red-colours-are-displayed-in-the-main-view.-¶
Status:¶	OK¶
Note-(optional):¶	N/A¶

b. → AIDA-V443.001-SA-002¶

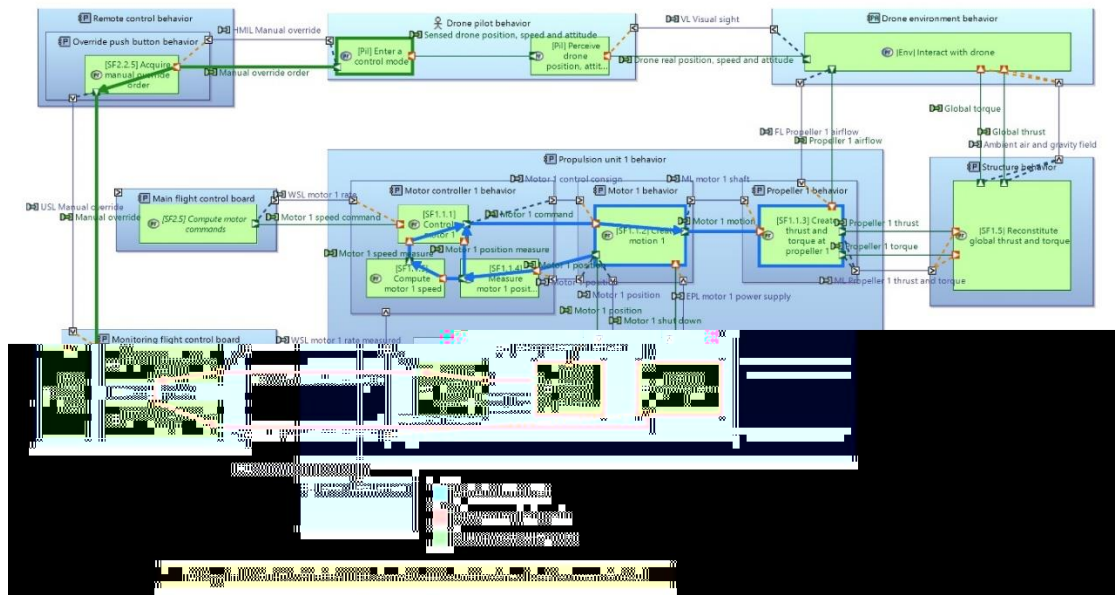
Action:¶	Check-for-observers-output-values-considered-in-this-test¶
Expected-results:¶	<ul style="list-style-type: none"> •→ Motor-1-(failed-motor)-is-disabled:¶ SF7.SF73.MonitorParameters.MotorsDisabled_Motor1=-TRUE¶ •→ Control-mode-is-no-longer-AUTO,it-is-MANUAL-now:¶ EXT_PilotDetection.mode=-AUTO¶ •→ Drone-loss-of-control:¶ FC01_CAT_Uncontrolled_drone_crash_in_unauthorized_area=-TRUE¶ FC02_HAZ_Uncontrolled_drone_crash_in_authorized_area=-TRUE¶
Observed-result:¶	Following-results-are-obtained:¶ SF7.SF73.MonitorParameters.MotorsDisabled_Motor1=-TRUE¶ EXT_PilotDetection.mode=-AUTO¶ FC01_CAT_Uncontrolled_drone_crash_in_unauthorized_area=-TRUE¶ FC02_HAZ_Uncontrolled_drone_crash_in_authorized_area=-TRUE¶
Status:¶	Not-passed¶

5.1.6 PoC A - PoC SE model with scenarios (not executable) and SA executable model

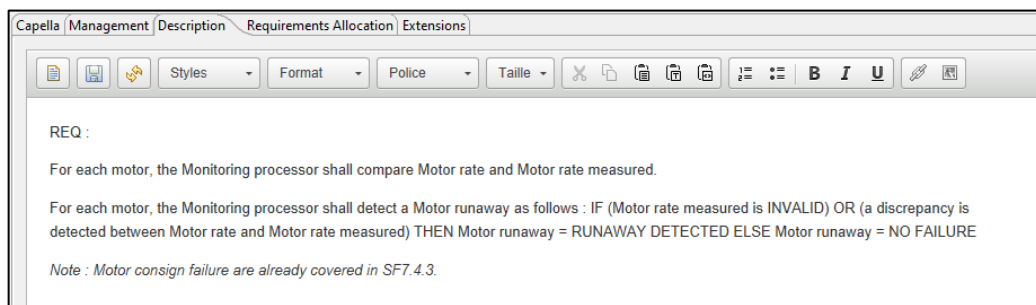
(a) Input Specialization

(i) SE Models:

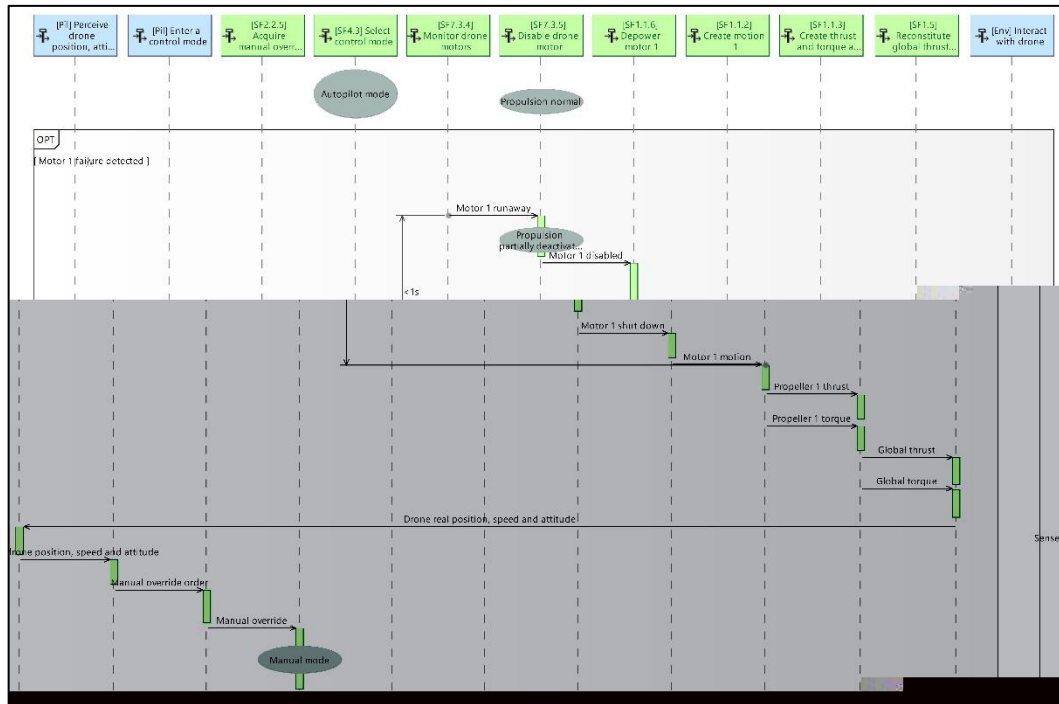
- Behaviors, Functions and associated Data Flows / Functional Chains:



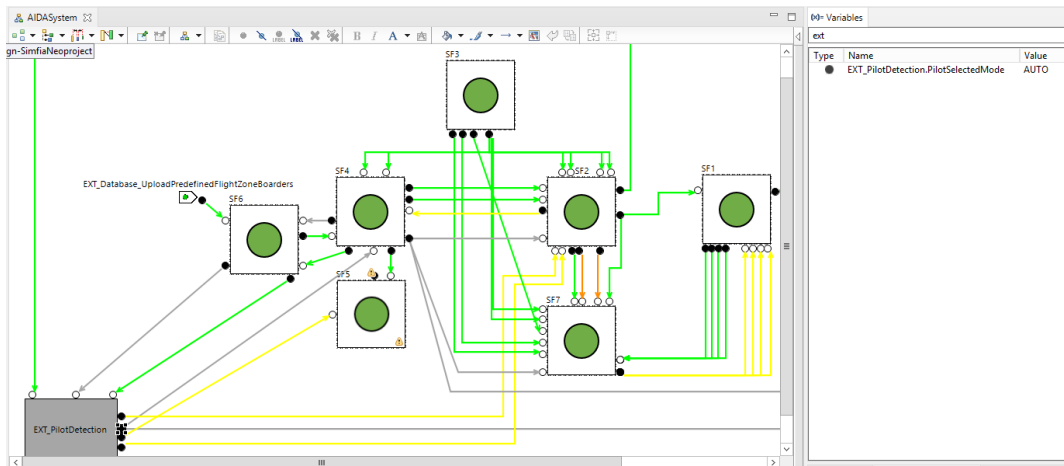
- Behavior Textual Requirements for Functions regarding Validity evaluation:



○ Functional Exchanges Scenarios:



(ii) SA Models :

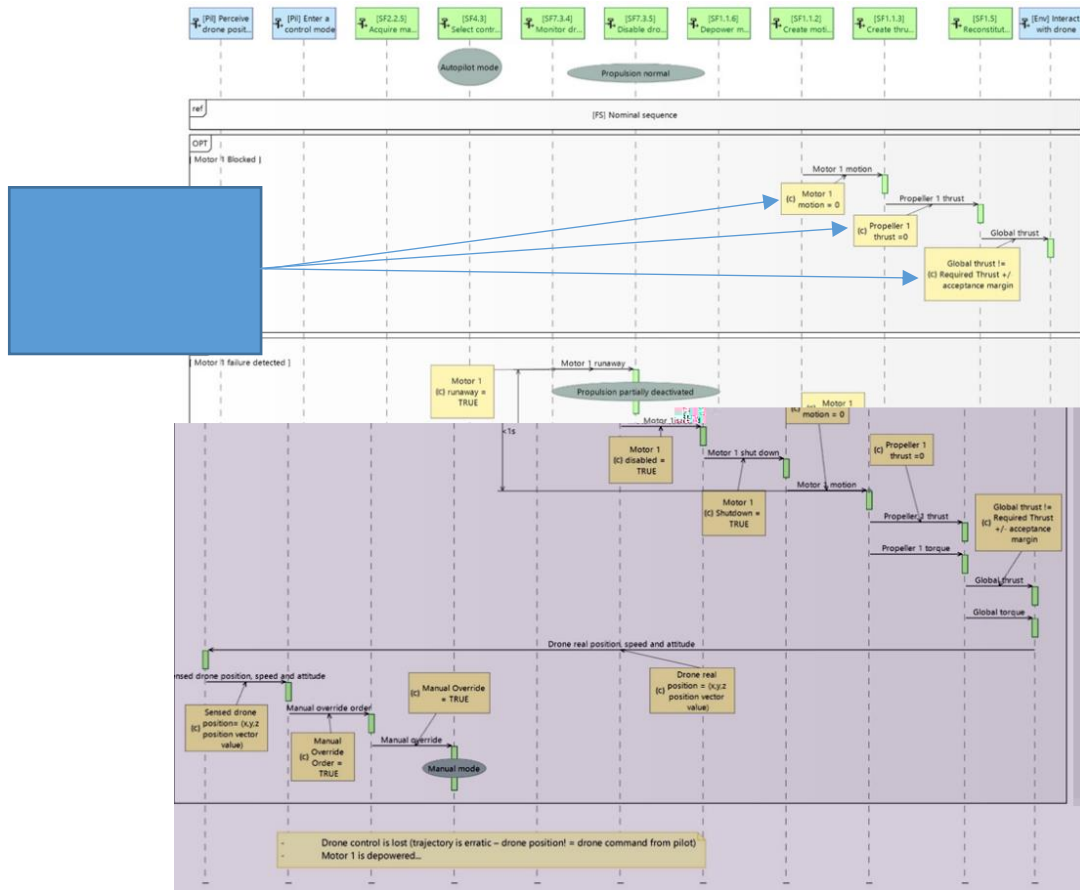


extended

(b) PoC Results

(i) SE

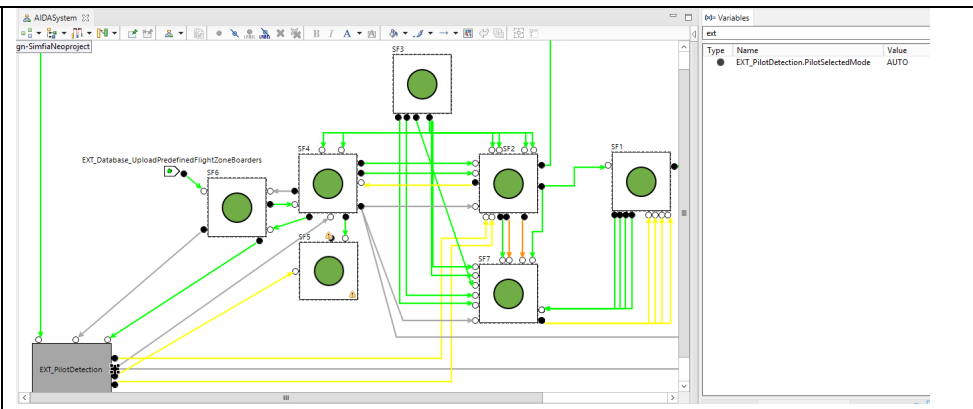
- _____ : Motor Loss



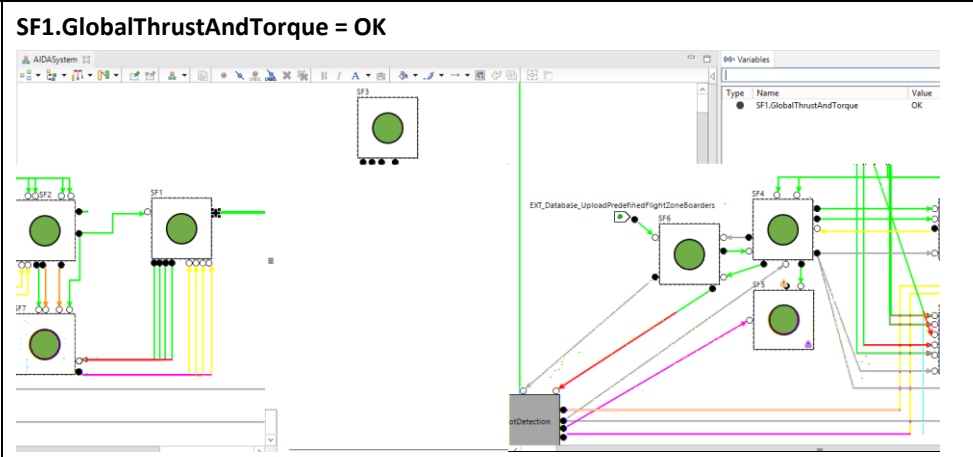
(ii) SA

AIDA-XX.YYY-SA-001

Action:	
Expectation:	EXT_PilotDetection.PilotSelectedMode = AUTO
Result:	EXT_PilotDetection.PilotSelectedMode = AUTO

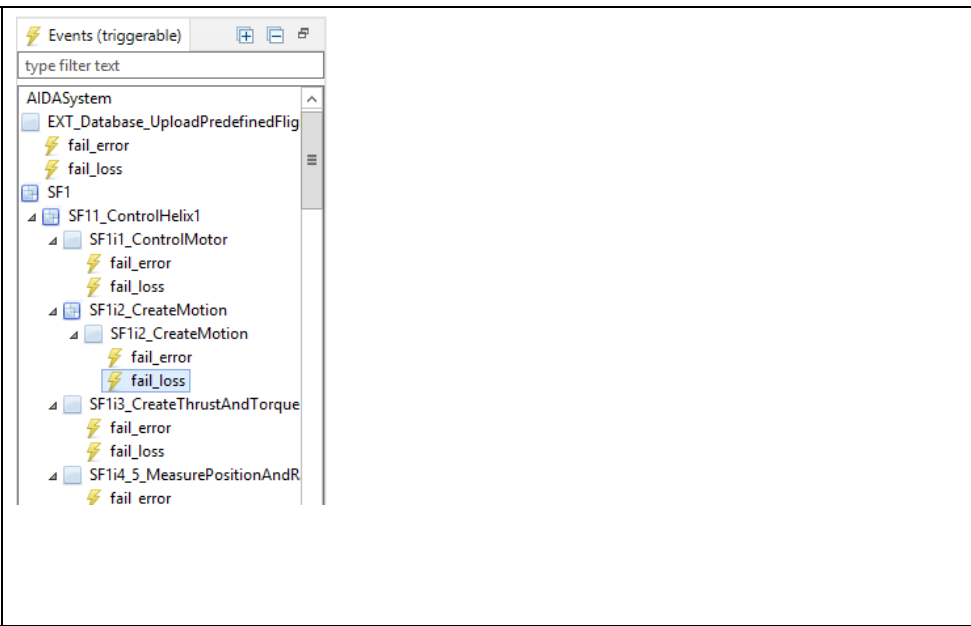
	
<p>Status:</p>	
<p>Note (opt.):</p>	

AIDA-XX.YYY-SA-002

<p>Action:</p>	
<p>Expectation:</p>	<p>SF1.GlobalThrustAndTorque = OK</p>
<p>Result:</p>	<p>SF1.GlobalThrustAndTorque = OK</p> 
<p>Status:</p>	
<p>Note (opt.):</p>	

AIDA-XX.YYY-SA-003

<p>Action:</p>	<p>SF1i2_CreateMotion fail_loss fail_loss</p>
<p>Expectation:</p>	

Result:	
Status:	
Note (opt.):	

AIDA-XX.YYY-SA-004

Action:	
Expectation:	SF1.SF11_ControlHelix1.SF1i3_CreateThrustAndTorque.output = LOST
Result:	SF1.SF11_ControlHelix1.SF1i3_CreateThrustAndTorque.output = LOST
Status:	
Note (opt.):	

AIDA-XX.YYY-SA-005

Action:	
Expectation:	SF1.GlobalThrustAndTorque = ERRONEOUS
Result:	SF1.GlobalThrustAndTorque = ERRONEOUS
Status:	
Note (opt.):	

AIDA-XX.YYY-SA-006

Action:	
Expectation:	SF7.SF3_MonitorParameters.SF734_MonitorDroneMotors.Motro1Runaway = TRUE
Result:	SF7.SF3_MonitorParameters.SF734_MonitorDroneMotors.Motro1Runaway = TRUE

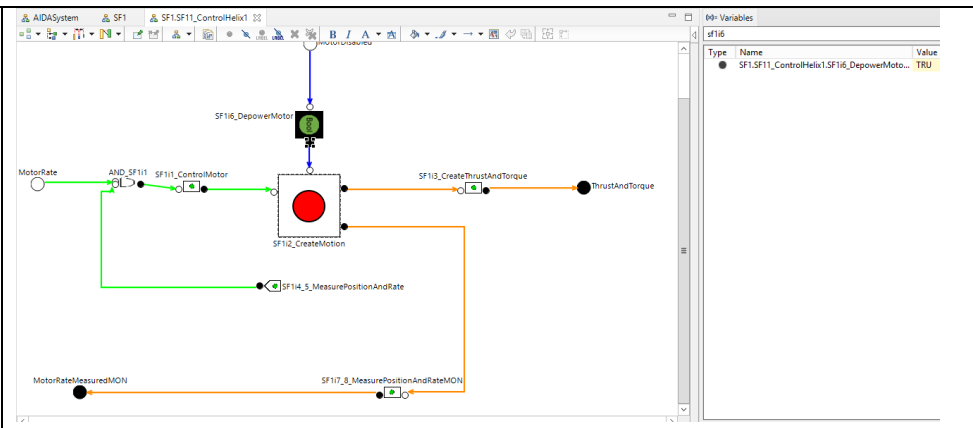
<p>Status:</p>	
<p>Note (opt.):</p>	

AIDA-XX.YYY-SA-007

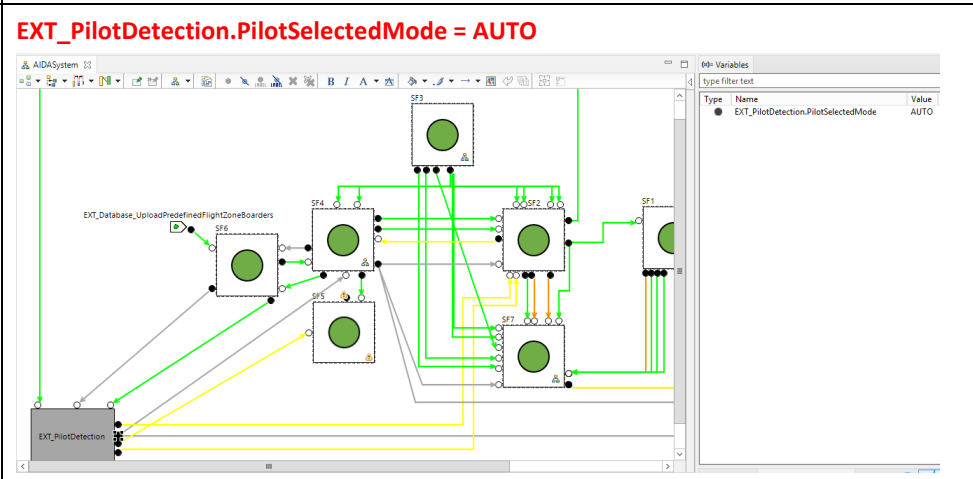
<p>Action:</p>	
<p>Expectation:</p>	<p>SF7.SF73.MonitorParameters.MotorsDisabled_MotorX = TRUE</p>
<p>Result:</p>	
<p>Status:</p>	
<p>Note (opt.):</p>	

AIDA-XX.YYY-SA-008

<p>Action:</p>	
<p>Expectation:</p>	<p>SF1.SF11_ControlHelix1.SF1i6_DepowerMotor = TRUE</p>
<p>Result:</p>	<p>SF1.SF11_ControlHelix1.SF1i6_DepowerMotor = TRUE</p>

	 <p>The screenshot shows a Simulink model for the SF116_DepowerMotor block. It features several interconnected blocks: AND_SF111, SF111_ControlMotor, SF112_CreateMotion, SF113_CreateThrustAndTorque, SF114_5_MeasurePositionAndRate, SF117_8_MeasurePositionAndRateMON, and MotorRateMeasuredMON. The MotorRate input is processed through AND_SF111 and SF111_ControlMotor to SF112_CreateMotion, which then feeds into SF113_CreateThrustAndTorque. Feedback loops from SF114_5 and SF117_8 are used for monitoring and control. The output is ThrustAndTorque.</p>
<p>Status:</p>	
<p>Note (opt.):</p>	

AIDA-XX.YYY-SA-009

	<p>SA</p>
<p>Action:</p>	
<p>Expectation:</p>	<p>EXT_PilotDetection.PilotSelectedMode = MANUAL</p>
<p>Result:</p>	<p>EXT_PilotDetection.PilotSelectedMode = AUTO</p>  <p>The screenshot shows a Simulink model for the EXT_PilotDetection block. It includes blocks for EXT_Database_UploadPredefinedFlightZoneBoards, SF3, SF4, SF5, SF6, SF7, SF8, and SF1. The model is a complex state machine or logic network. The output variable EXT_PilotDetection.PilotSelectedMode is shown as AUTO in the Variables window.</p>
<p>Status:</p>	
<p>Note (opt.):</p>	

(c) **PoC Conclusion**

5.1.7 **PoC B - PoC executable SE model**

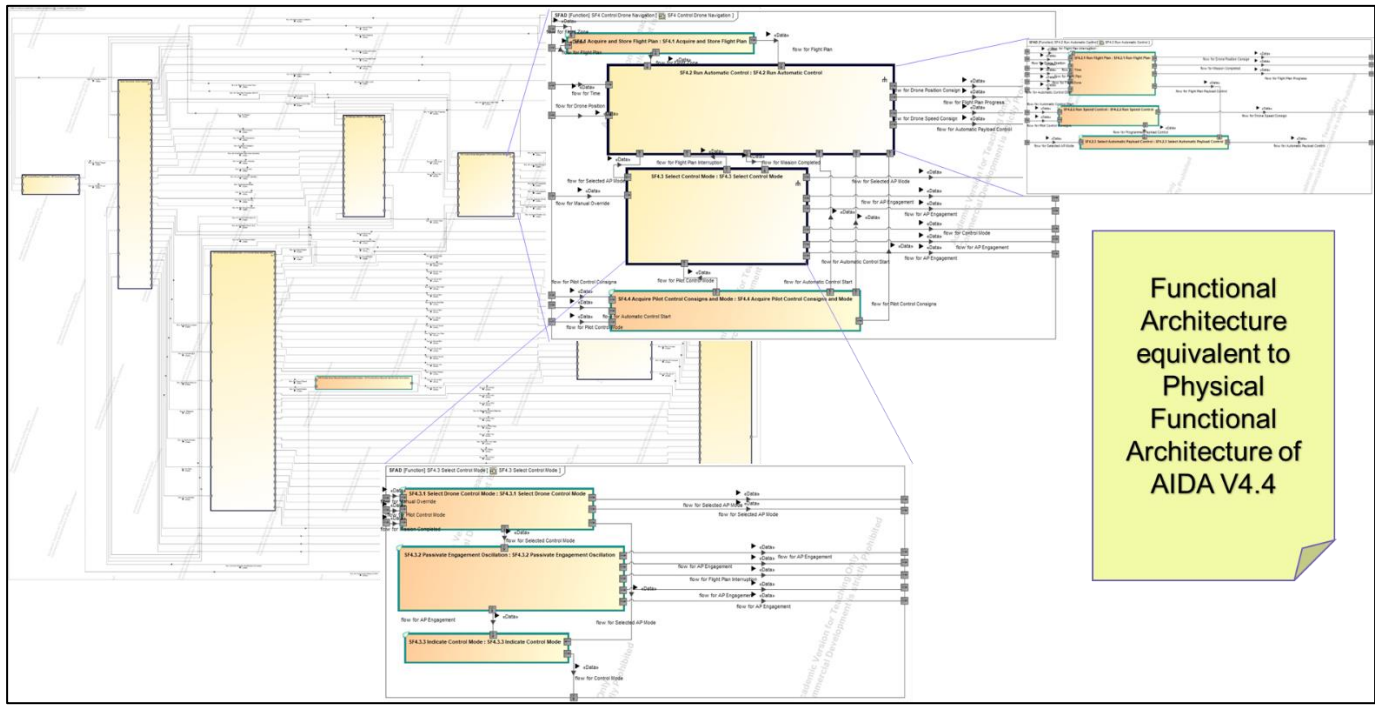
(a) **PoC Specialization**

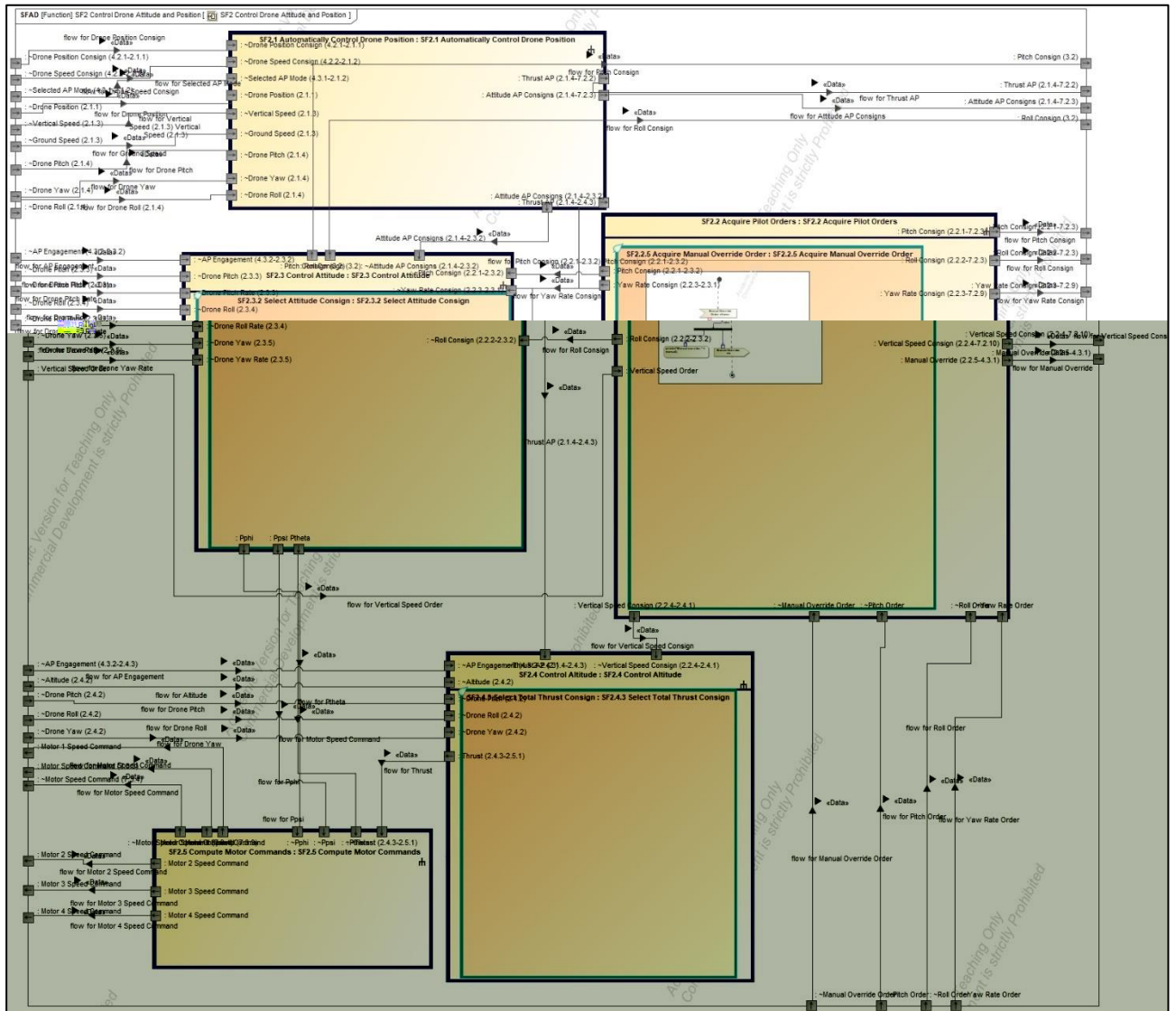
PoC A *PoC B*

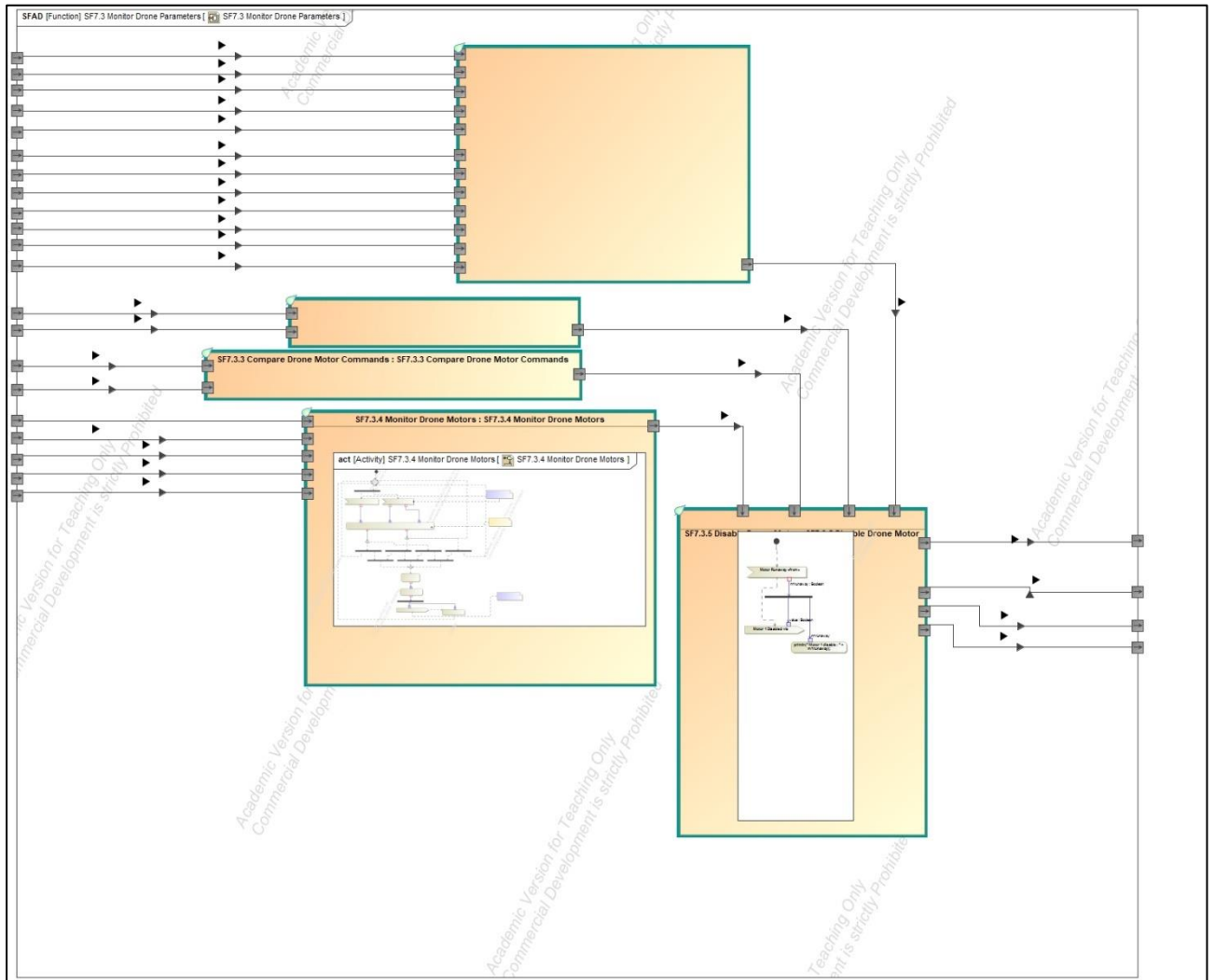
Annex SAMAREQ Profile

Annex Derived Result

➤ **SysML model – Functional Architecture overview**



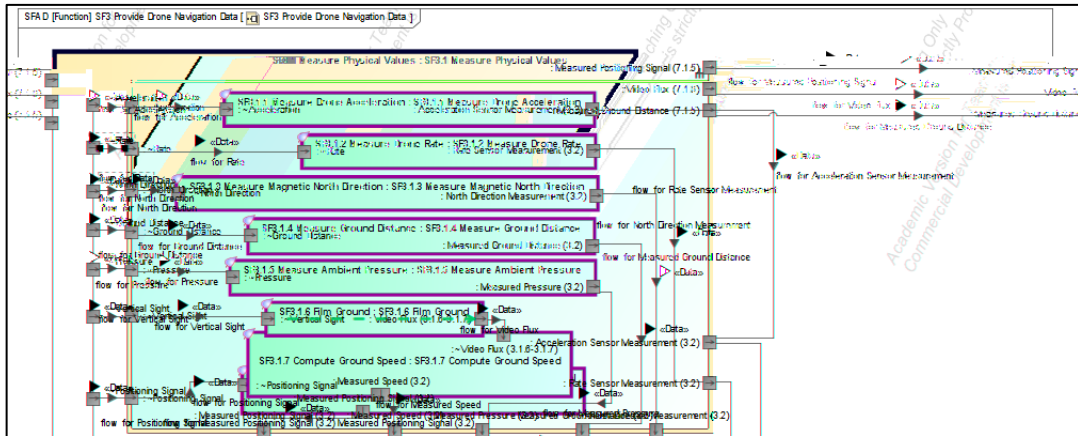




➤ Behavior associated to Leaf Function

➤ Generalization of delegation mechanism

➤ **Interface and Data Flows Specification**

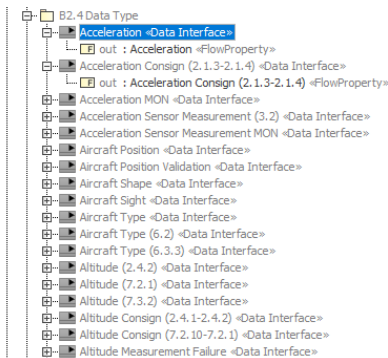


○ **Interface Block Type :**

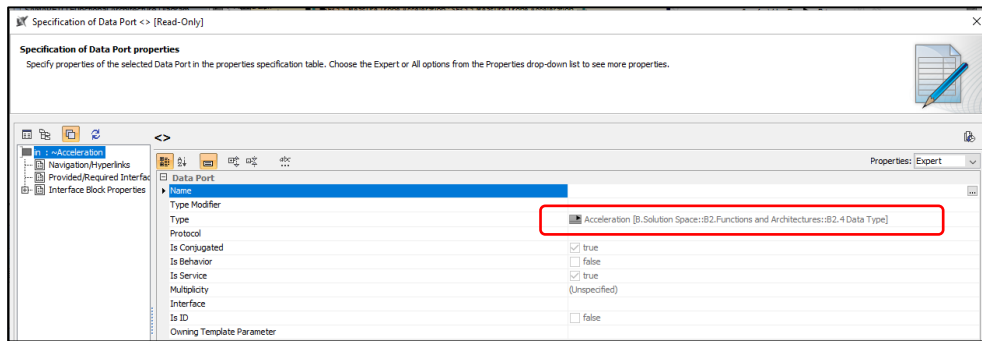
Specification of Data Interface Acceleration [Read-Only]

Specify properties of the selected Data Interface in the properties specification table. Choose the Expert or All options from the Properties drop-down list to see more properties.

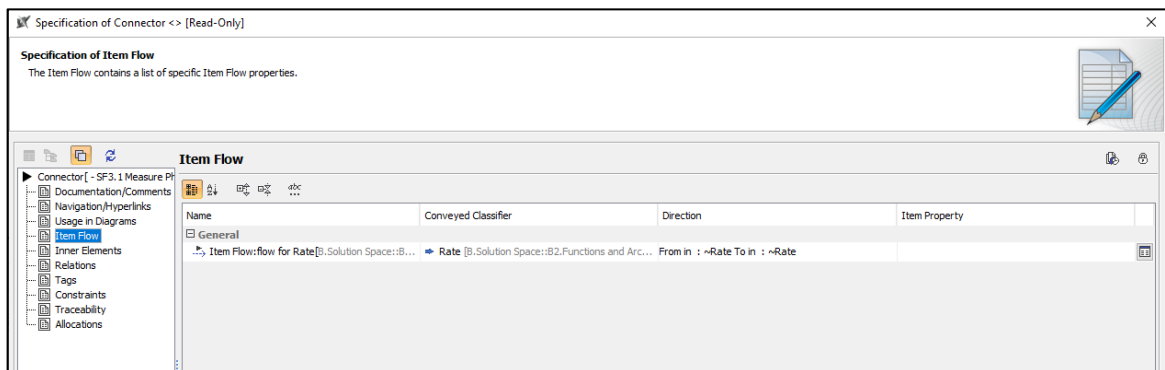
Acceleration	
Data Interface	Acceleration
Name	Acceleration
Qualified Name	B.Solution Space::B2.Functions and Architectures::B2.4 Data Type:Acceleration
Owner	B2.4 Data Type [B.Solution Space::B2.Functions and Architectures]
Applied Stereotype	Data Interface [Class] [SAMARETO_FA_Profile::Functional Analysis]
public	<input type="checkbox"/>
isLeaf	<input type="checkbox"/>
isFinalSpecialization	<input type="checkbox"/>
isActive	<input type="checkbox"/>
isAbstract	<input type="checkbox"/>
Active Hyperlink	<input type="checkbox"/>
Image	<input type="checkbox"/>
To Do	<input type="checkbox"/>
General	<input type="checkbox"/>
Is Encapsulated	<input type="checkbox"/>

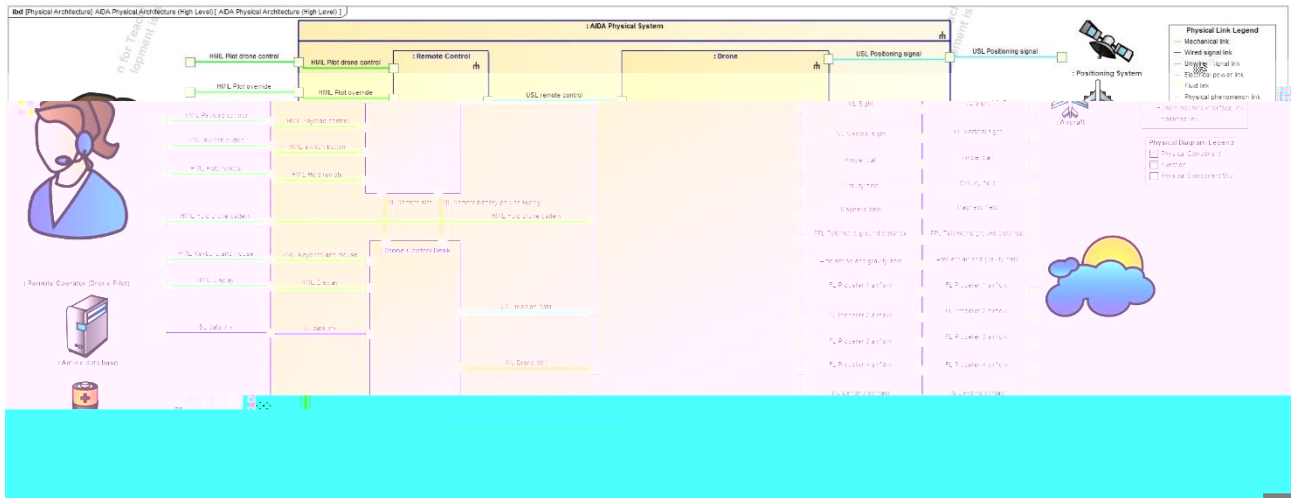


○ Port Definition



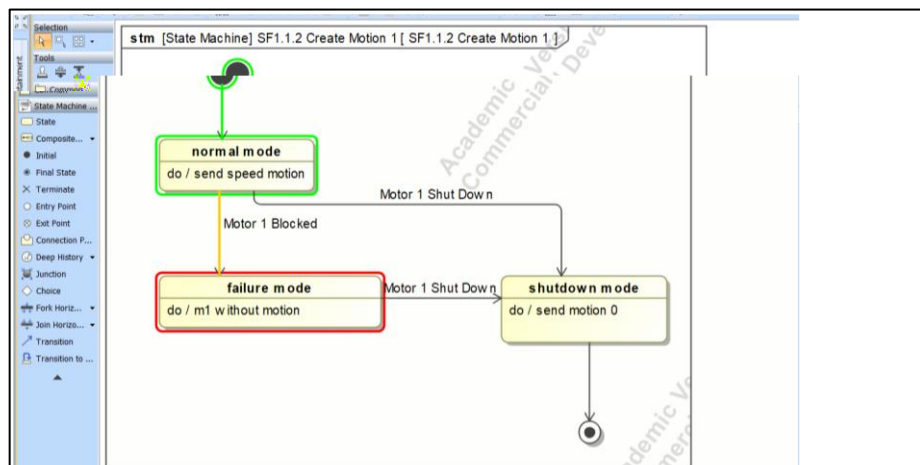
○ Item Flows

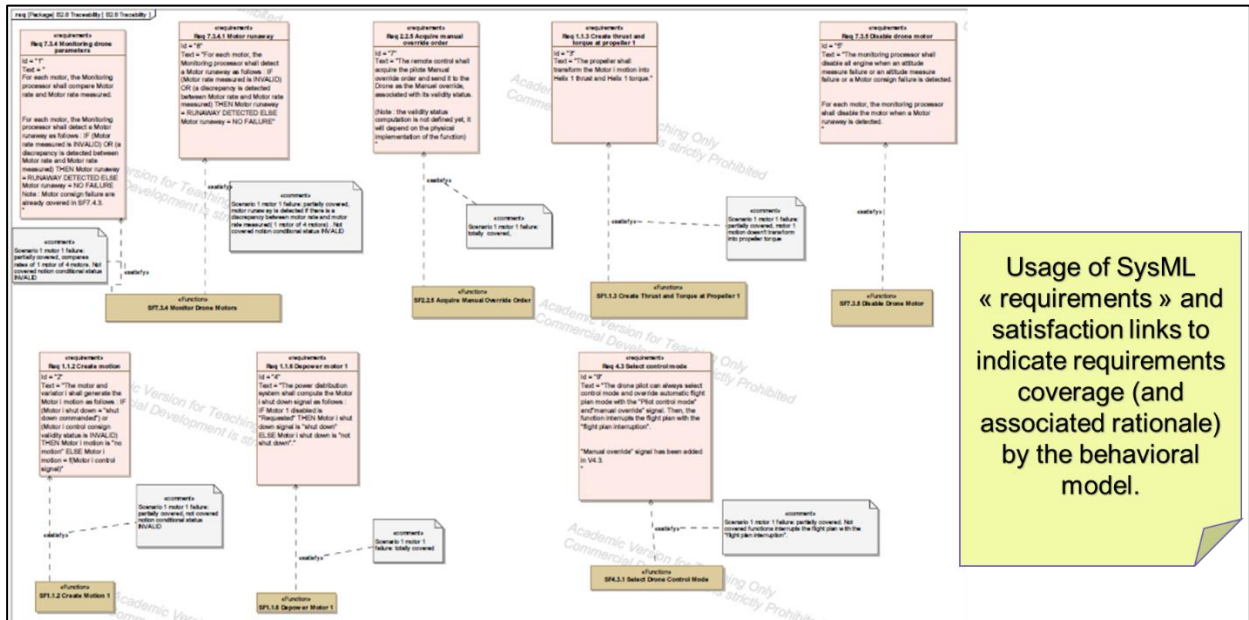
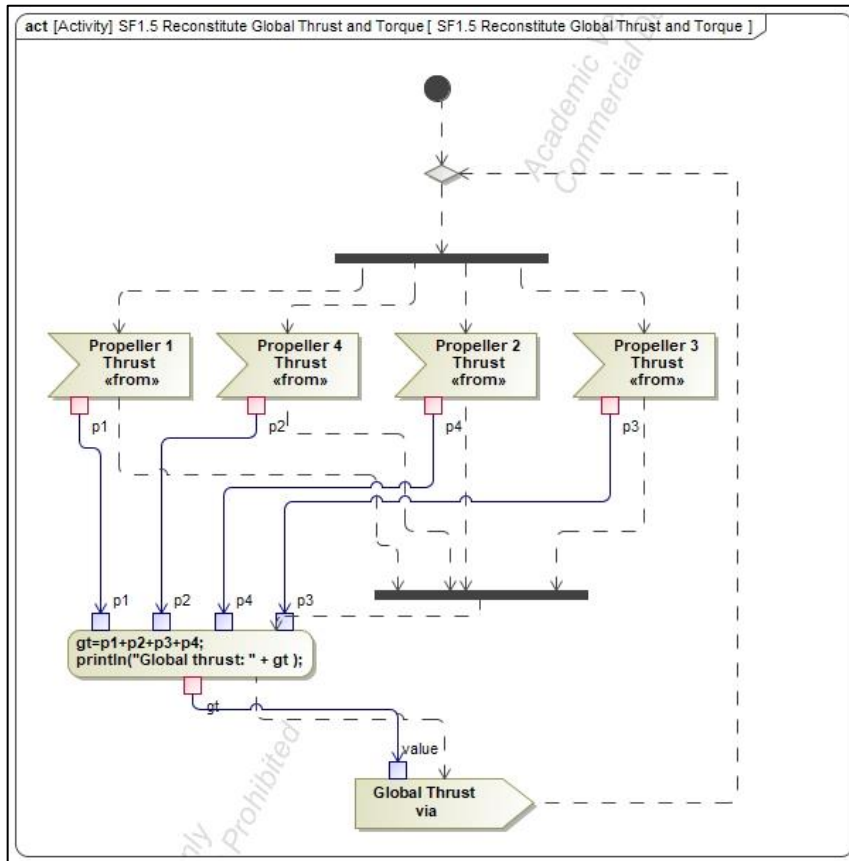




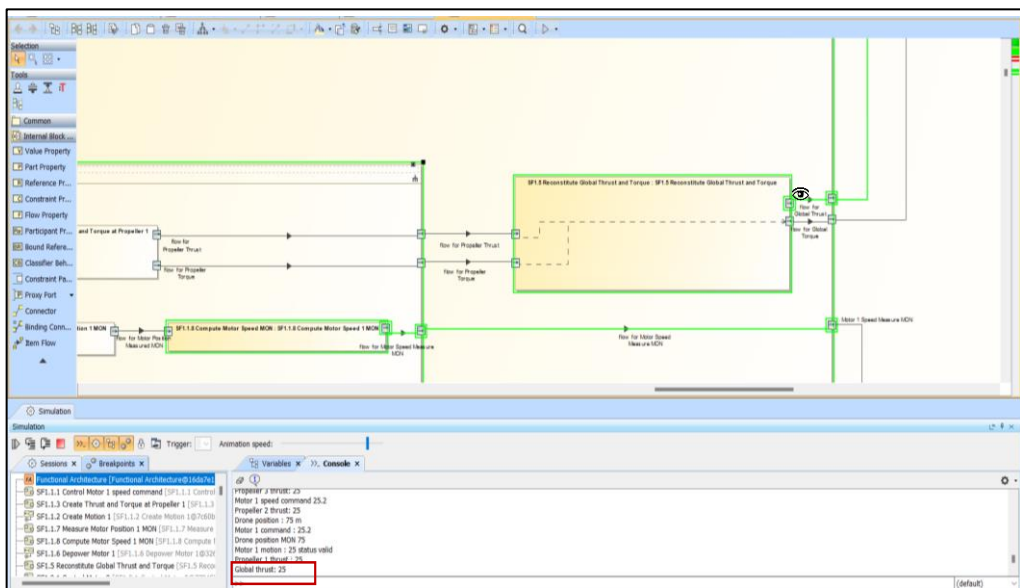
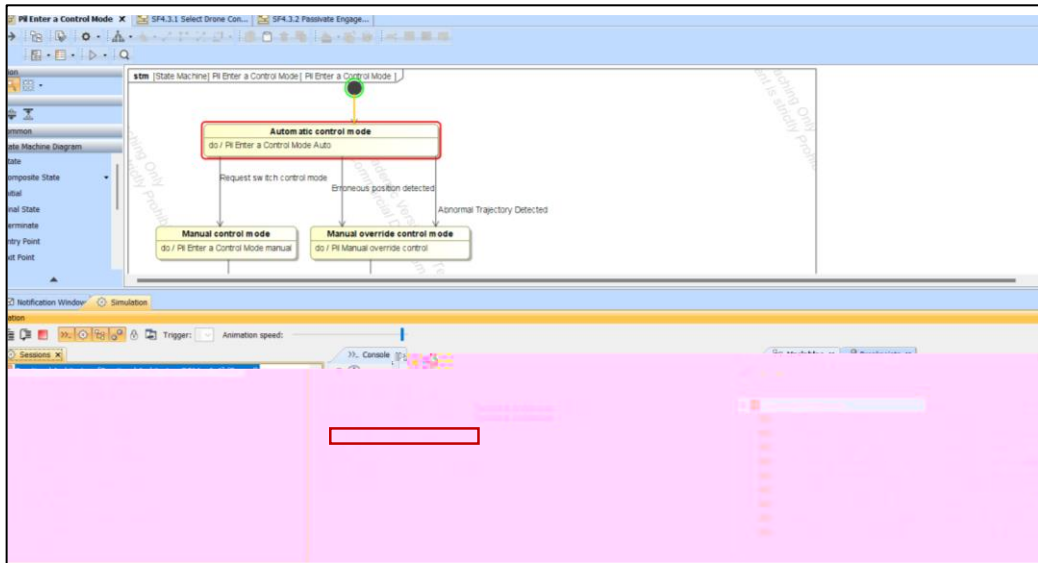
➤ **SysML model – Behavioral overview**

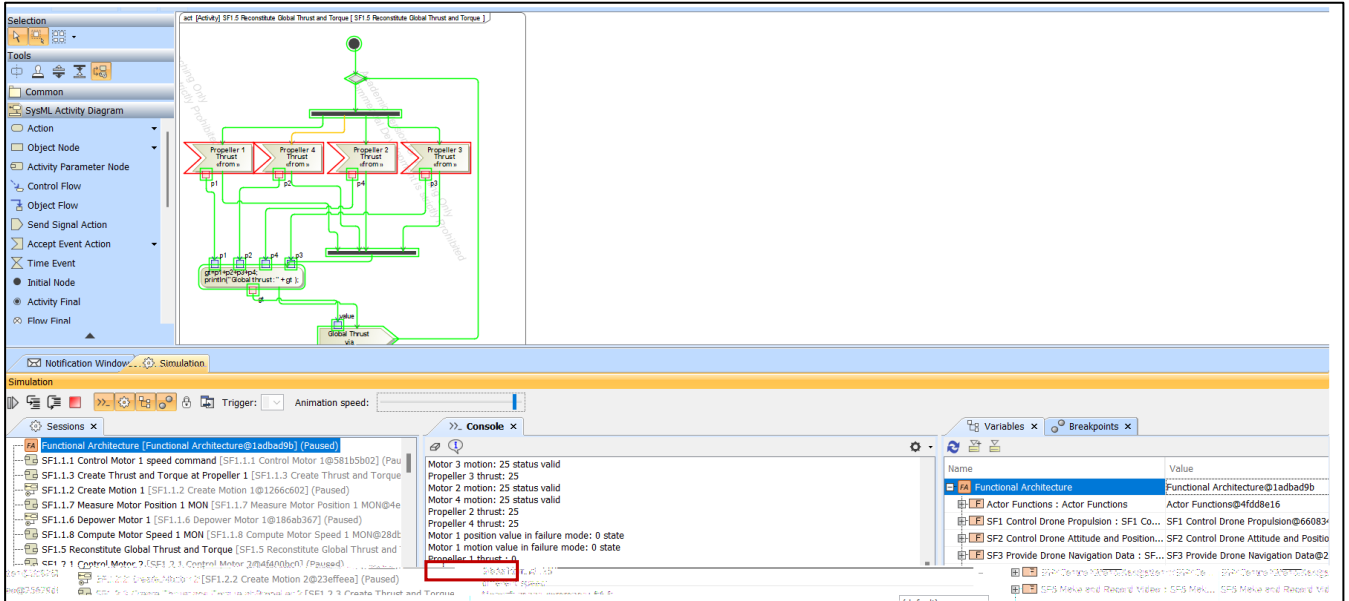
_____ **Loss of a Motor**





Usage of SysML « requirements » and satisfaction links to indicate requirements coverage (and associated rationale) by the behavioral model.



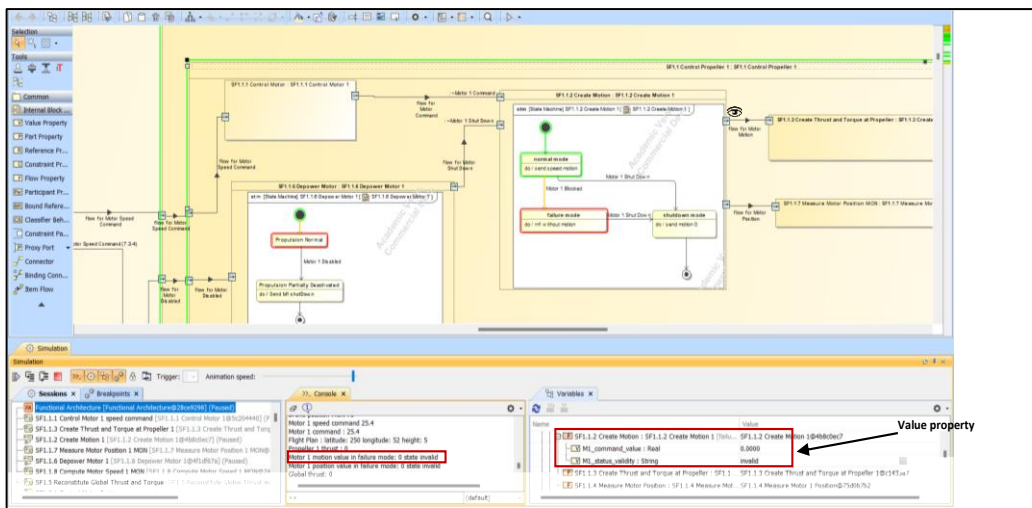


The screenshot shows a SysML simulation environment. The top part displays a SysML Activity Diagram with four propeller thrust nodes (Propeller 1, 2, 3, 4) and a global thrust variable. The bottom part shows the simulation console with the following log entries:

```

    Motor 3 motion: 25 status valid
    Propeller 3 thrust: 25
    Motor 2 motion: 25 status valid
    Propeller 2 thrust: 25
    Propeller 4 thrust: 25
    Motor 1 position value in failure mode: 0 state
    Motor 1 motion value in failure mode: 0 state
  
```

The console also shows a list of sessions and variables on the right side of the interface.

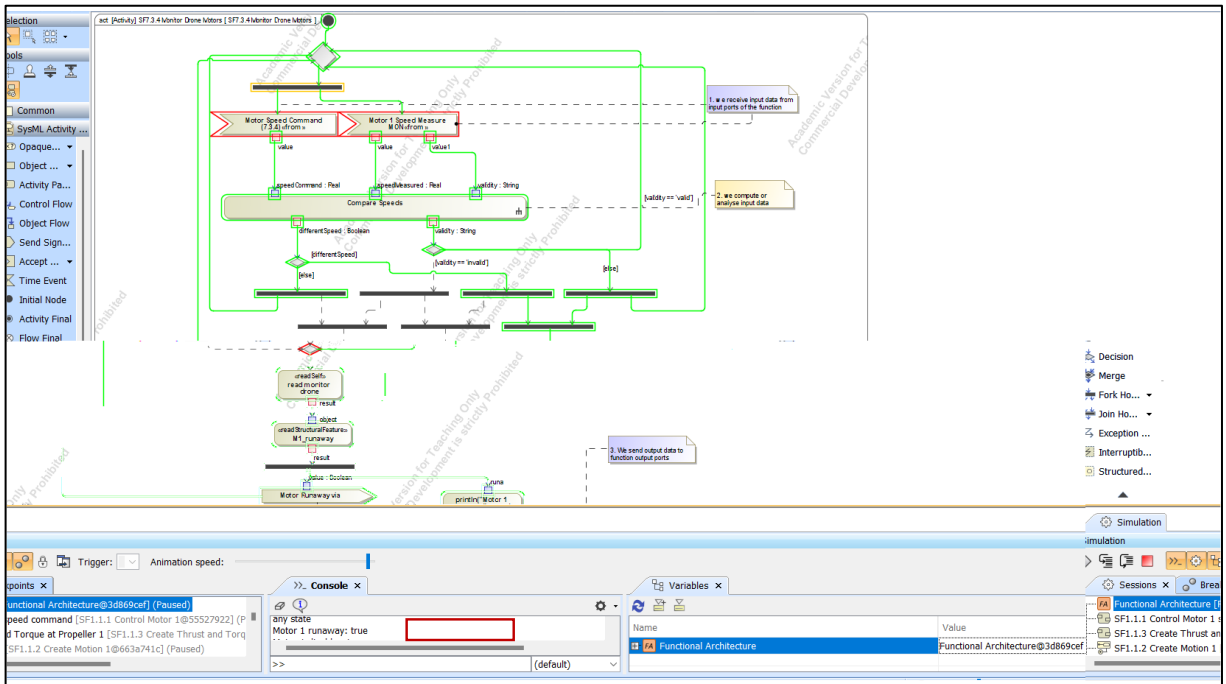


The screenshot shows a detailed SysML simulation environment for a propeller control system. The top part displays a SysML Activity Diagram with various control and motion nodes. The bottom part shows the simulation console with the following log entries:

```

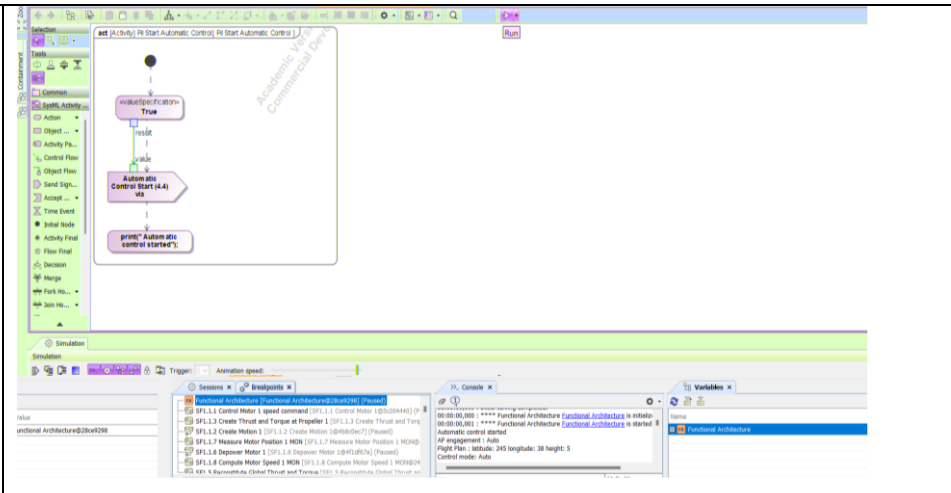
    Motor 1 speed command: 25.4
    Motor 2 command: 25.4
    Flight Plan: latitude: 250 longitude: 53 height: 5
    Propeller 4 thrust: 25
    Motor 1 position value in failure mode: 0 state mode
    Motor 1 motion value in failure mode: 0 state mode
    Global Thrust: 0
  
```

The console also shows a list of sessions and variables on the right side of the interface. An arrow points to the 'Value property' in the console output.



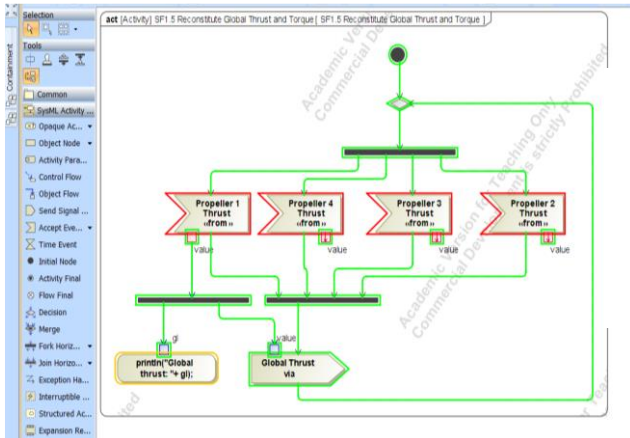
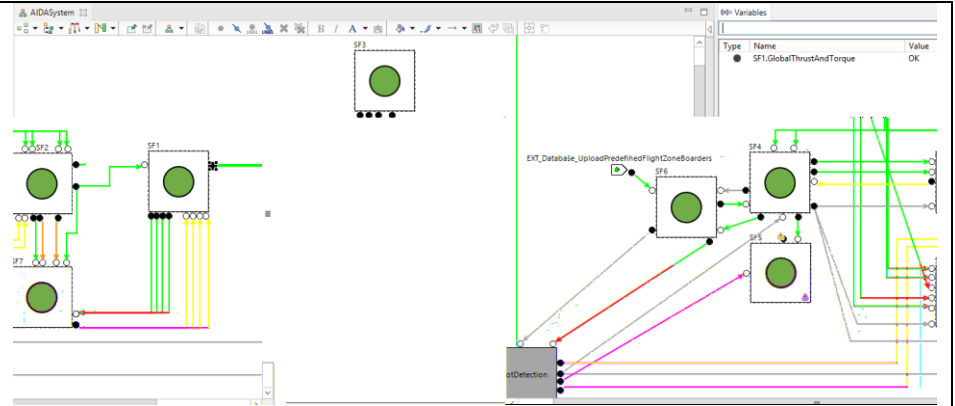
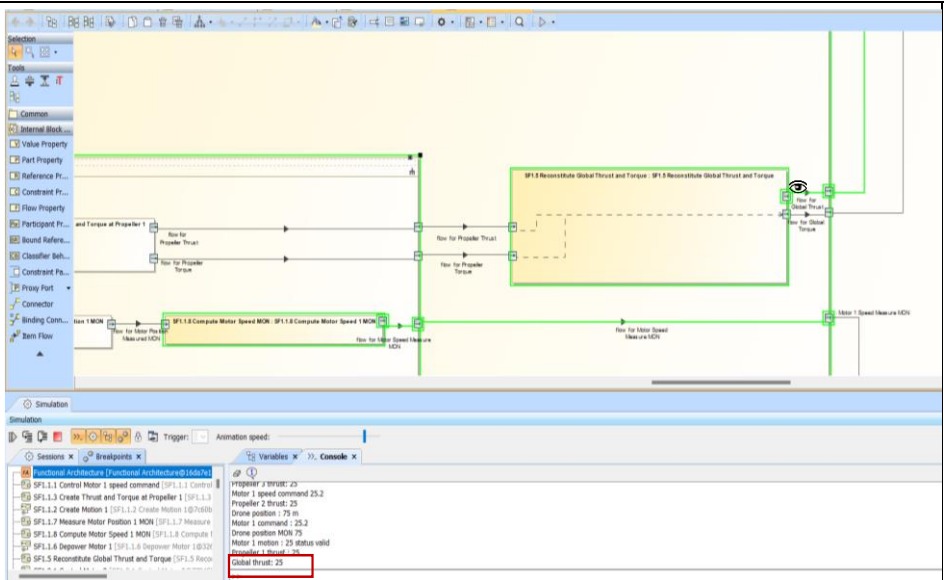
(b) PoC Results
 (i) AIDA-XX.YYY-SA-001

	SE	SA
Action:		
Expectation:	<p>Drone real position</p> <p>Control Mode = AUTOMATIC</p>	<p>EXT_PilotDetection.PilotSelectedMode = AUTO</p>
Result:		<p>EXT_PilotDetection.PilotSelectedMode = AUTO</p>

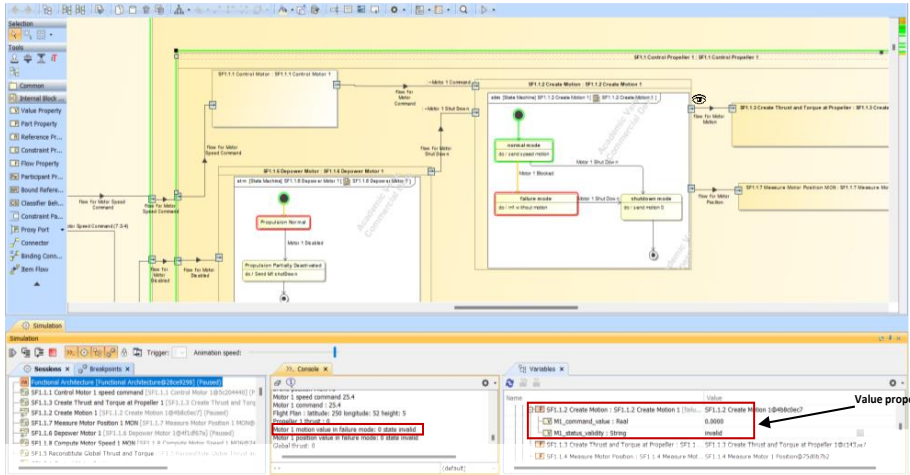
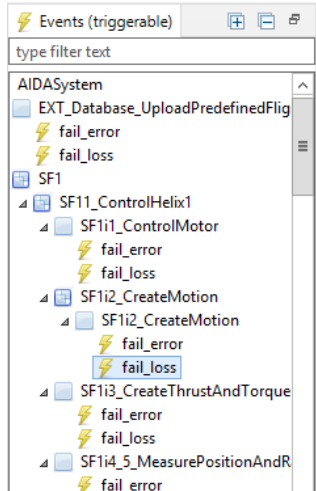
	 <p>The screenshot shows a simulation environment. The top part displays a state machine diagram with nodes like 'initialSpecification', 'result', 'start', 'Action start: Control Start (1.4)', and 'print: Automatic control started!'. The bottom part shows a console window with logs for 'SF1.1 Control Motor 1 speed command', 'SF1.2 Create Thrust and Torque at Propeller 1', 'SF1.3 Create Thrust and Torque at Propeller 2', 'SF1.4 Create Motor 1', 'SF1.5 Create Motor 2', 'SF1.6 Depressor Motor 1', 'SF1.7 Measure Motor Position 1', 'SF1.8 Depressor Motor 2', 'SF1.9 Create Motor Speed 1', and 'SF1.10 Create Motor Speed 2'. A 'Console' window shows 'Functional Architecture' logs.</p>	
<p>Status:</p>		
<p>Note (opt.):</p>		

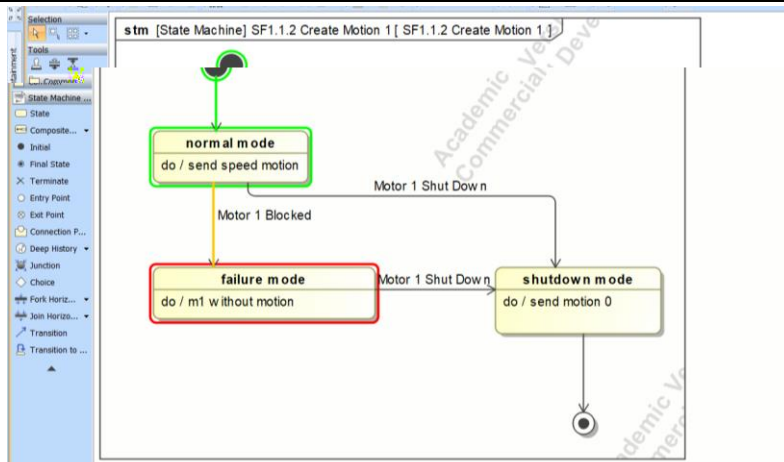
(ii) AIDA-XX.YYY-SA-002

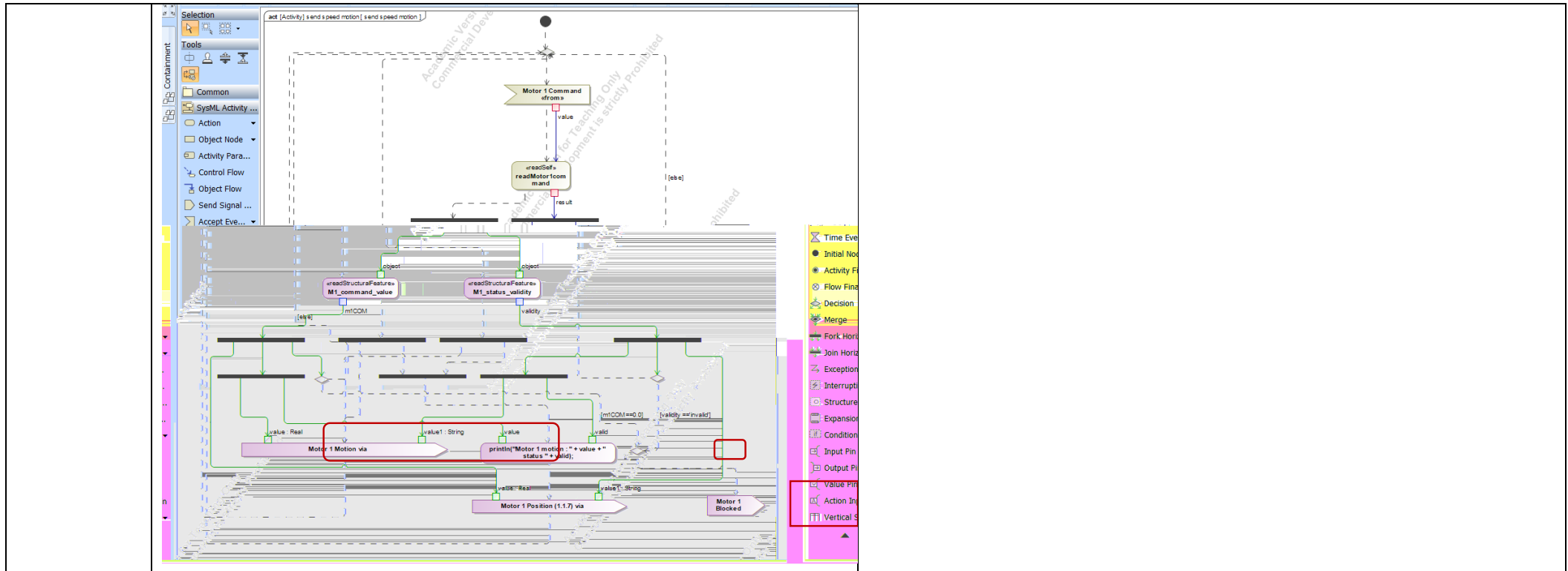
	SE	SA
<p>Action:</p>		
<p>Expectation:</p>	Global Thrust = Required thrust +/- acceptance margin	SF1.GlobalThrustAndTorque = OK
<p>Result:</p>		SF1.GlobalThrustAndTorque = OK



Status:

Note (opt.):		
(iii)	AIDA-XX.YYY-SA-003	
Action:	SE	SA fail_loss SF1i2_CreateMotion fail_loss
Expectation:	Motor x motion = 0	
Result:		

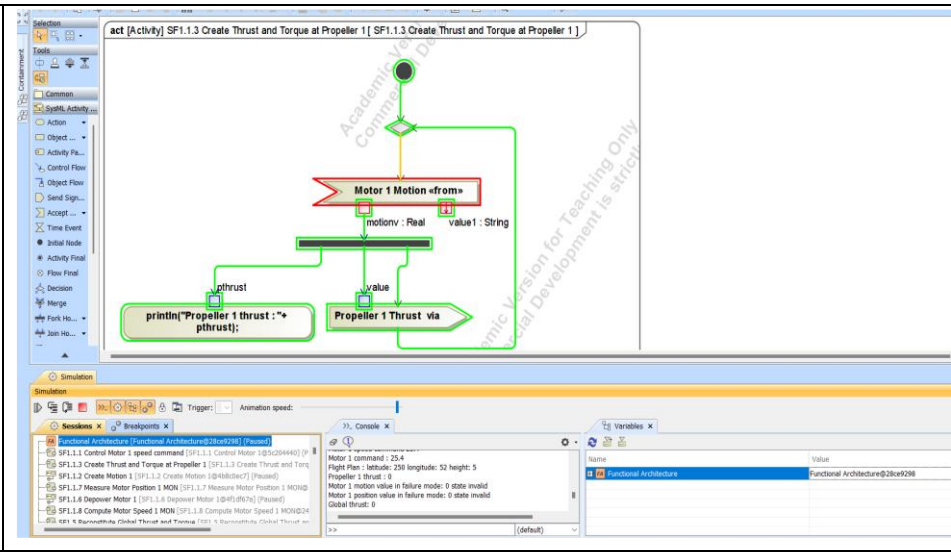




Status:		
Note (opt.):		

(iv) AIDA-XX.YYY-SA-004

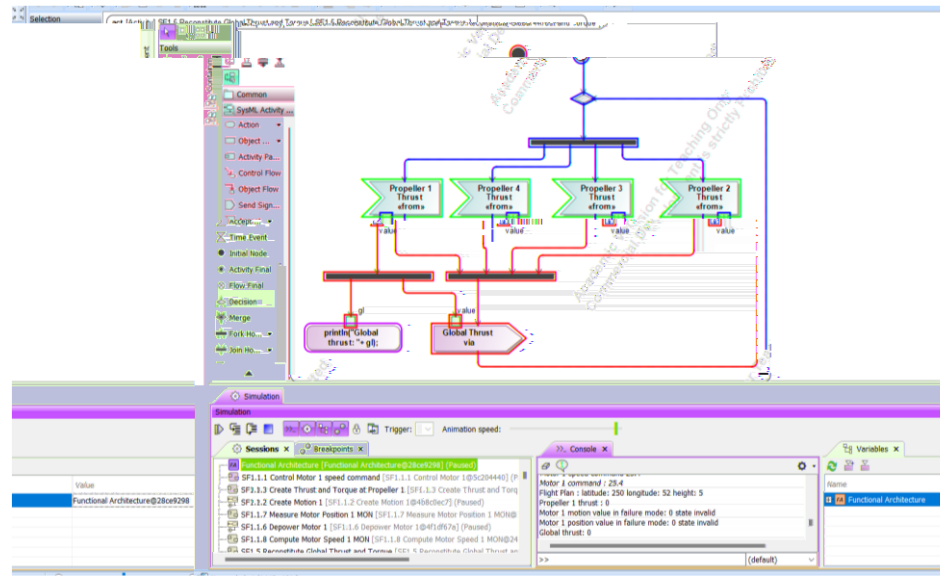
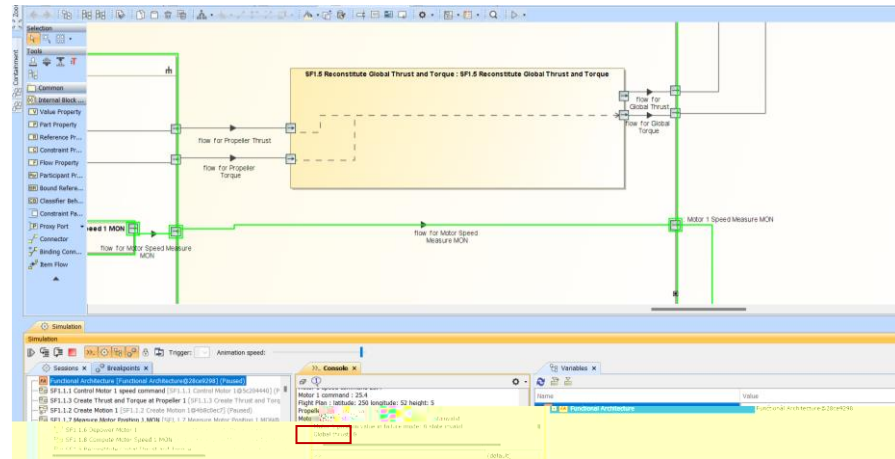
	SE	SA
Action:		
Expectation:	<i>Propeller x thrust = 0</i>	SF1.SF11_ControlHelix1.SF1i3_CreateThrustAndTorque.output = LOST

		
<p>Status:</p>		
<p>Note (opt.):</p>		

(v) AIDA-XX.YYY-SA-005

	<p>SE</p>	<p>SA</p>
<p>Action:</p>		
<p>Expectation:</p>	<p>Global thrust! = Required Thrust +/- acceptance margin</p>	<p>SF1.GlobalThrustAndTorque = ERRONEOUS</p>

Result:



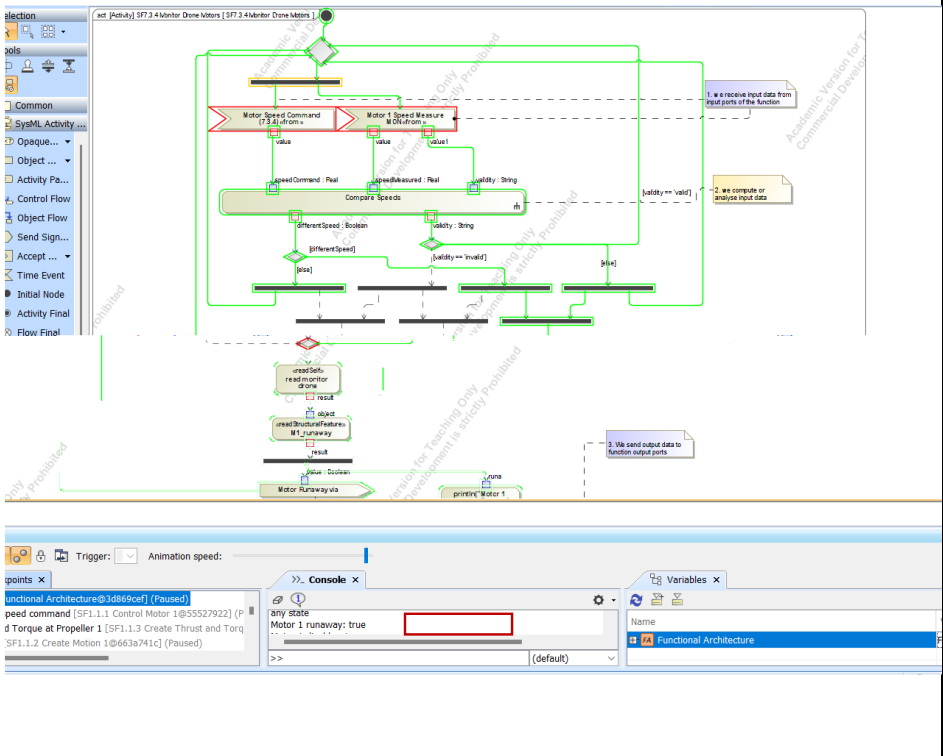
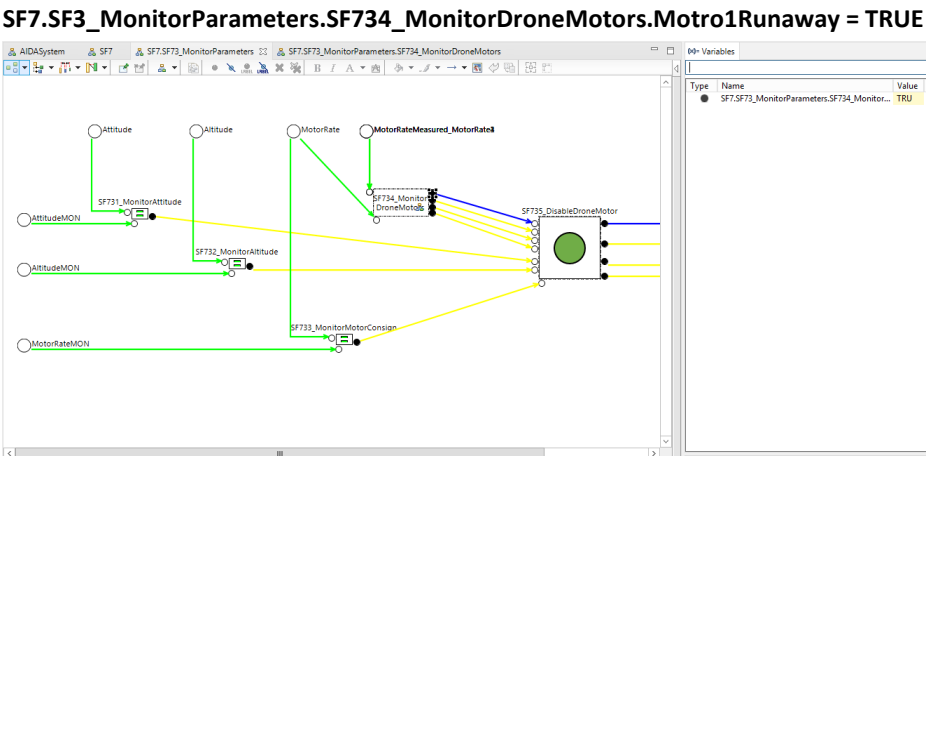
SF1.GlobalThrustAndTorque = ERRONEOUS

Status:		
Note (opt.):		

(vi) AIDA-XX.YYY-SA-006

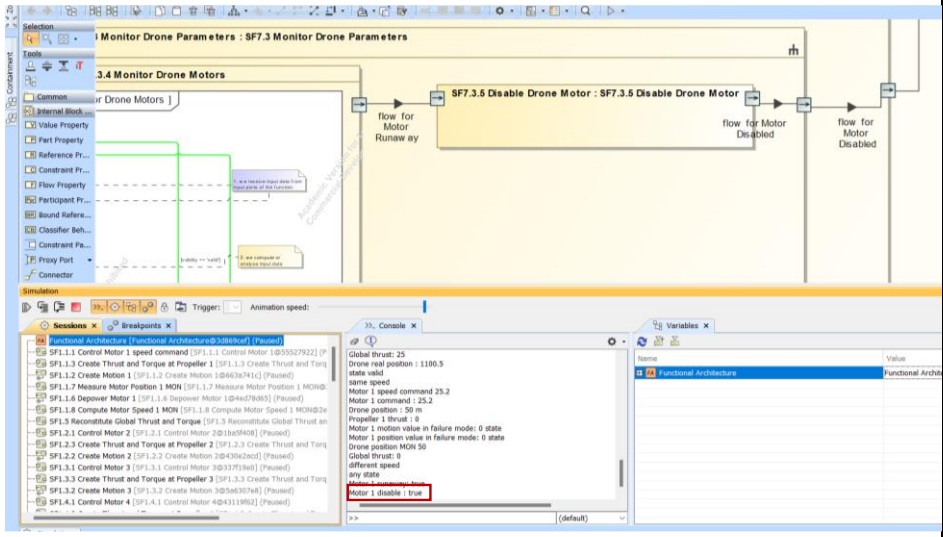
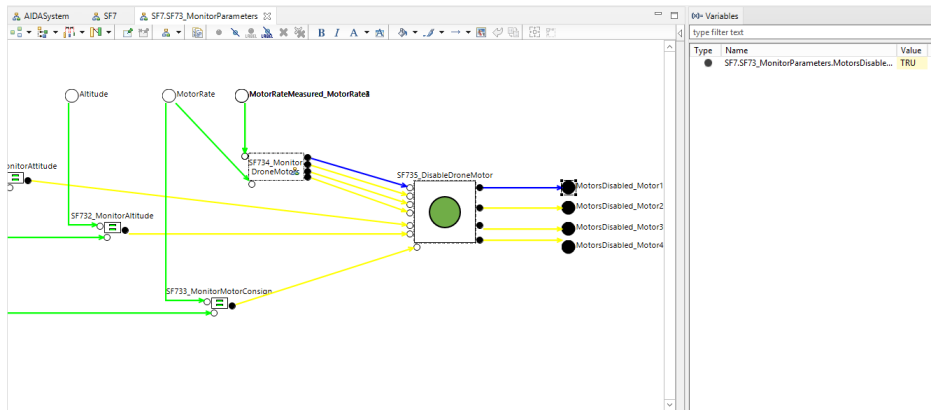
SE	SA
Action:	

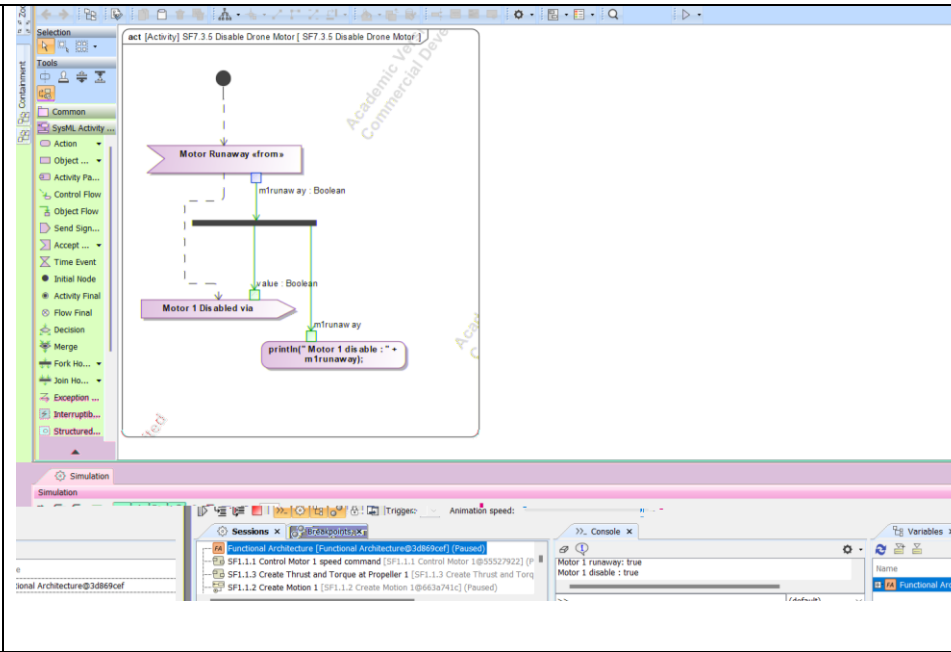
Expectation:	Motor x runaway = TRUE	SF7.SF3_MonitorParameters.SF734_MonitorDroneMotors.Motro1Runaway = TRUE
--------------	-------------------------------	--

<p>Result:</p> 	<p>SF7.SF3_MonitorParameters.SF734_MonitorDroneMotors.Motro1Runaway = TRUE</p> 
---	--

Status:		
Note (opt.):		

(vii) AIDA-XX.YYY-SA-007

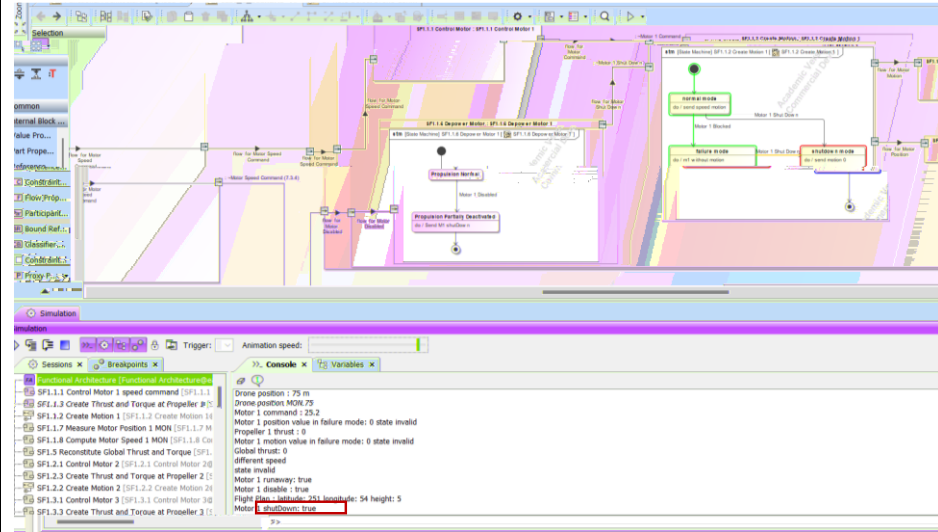
	SE	SA
Action:		
Expectation:	Motor x disabled = TRUE	SF7.SF73.MonitorParameters.MotorsDisabled_MotorX = TRUE
Result:		

	 <p>The screenshot shows a SysML activity diagram for 'SF7.3.5 Disable Drone Motor'. It features a state transition from 'Motor Runaway <from>' to 'Motor 1 Disabled via'. The transition is triggered by the condition 'mrunaway : Boolean'. The 'Motor 1 Disabled via' state is followed by an action 'print()' with the message 'Motor 1 disable : * mrunaway;'. The diagram is displayed in a software environment with various toolbars and a simulation console at the bottom.</p>	
<p>Status:</p>		
<p>Note (opt.):</p>		

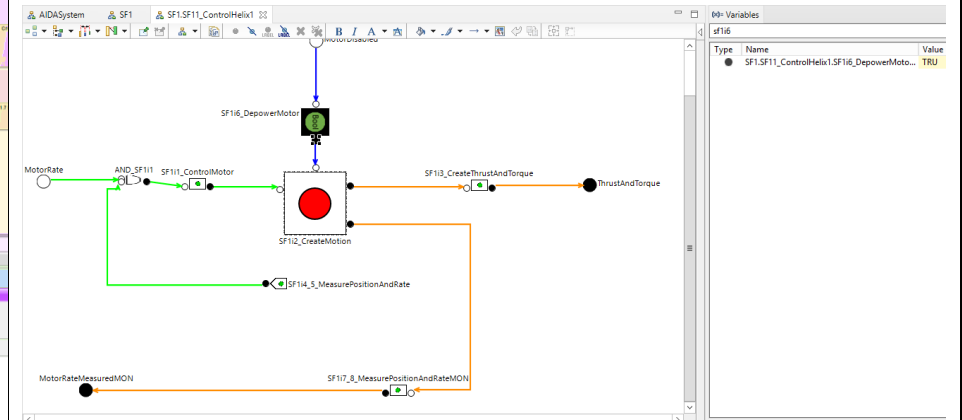
(viii) AIDA-XX.YYY-SA-008

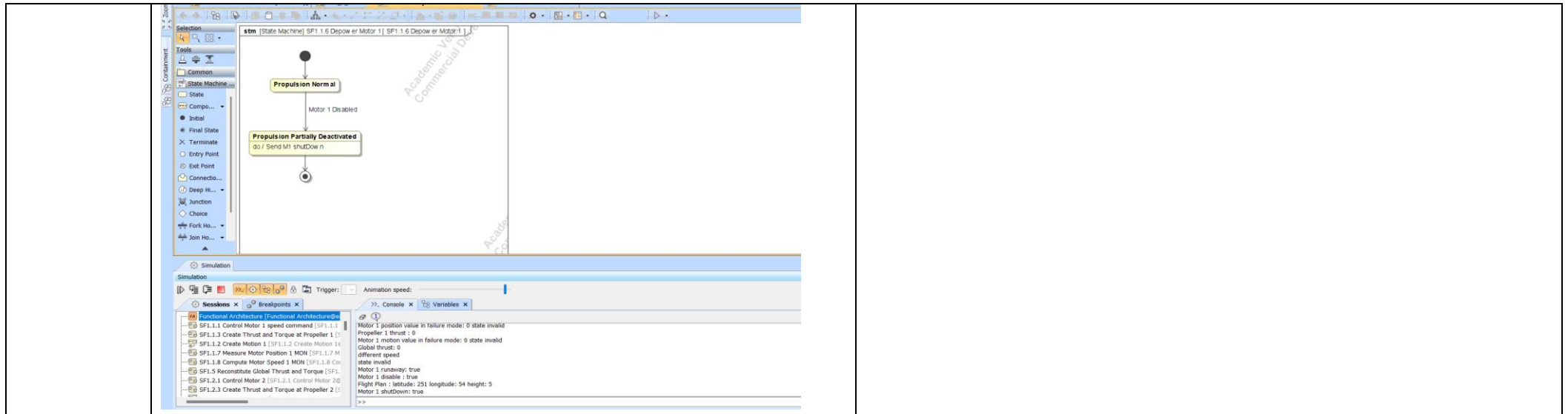
	<p>SE</p>	<p>SA</p>
<p>Action:</p>		
<p>Expectation:</p>	<p>Motor x Shutdown = TRUE</p>	<p>SF1.SF11_ControlHelix1.SF1i6_DepowerMotor = TRUE</p>

Result:



SF1.SF11_ControlHelix1.SF1i6_DepowerMotor = TRUE

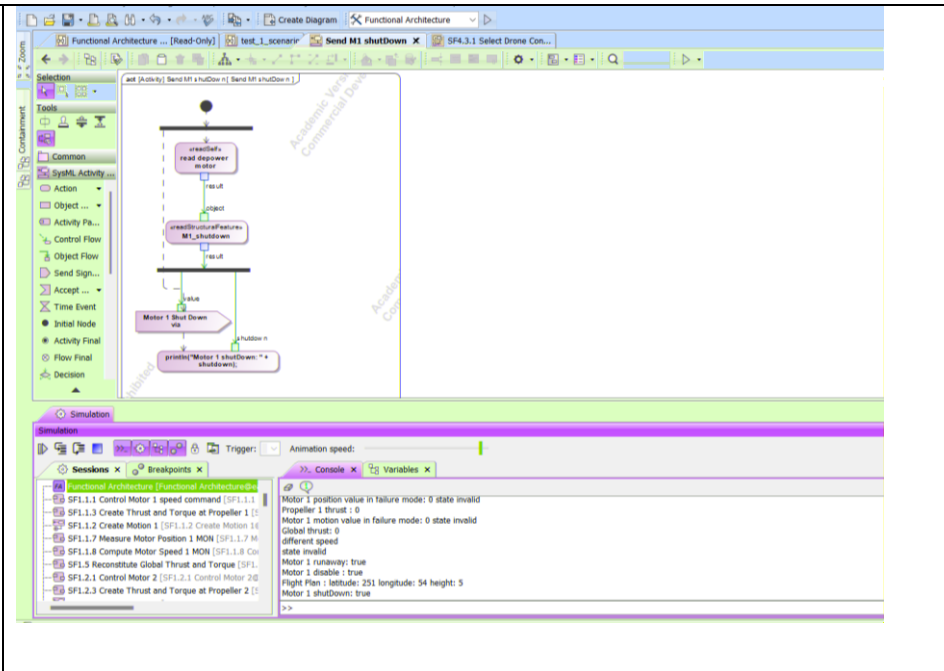




The screenshot displays a state machine simulation interface. The main window shows a state machine diagram with two states: 'Propulsion Normal' and 'Propulsion Partially Deactivated'. A transition arrow labeled 'Motor 1 Disabled' connects the two states. The 'Propulsion Partially Deactivated' state contains the action 'do / Send M1 shutDown n'. The left sidebar shows a project tree with 'stm [State Machine] SF1.1.6 Depow er Motor 1 | SF1.1.6 Depow er Motor 1 |'. Below the diagram is a simulation control panel with buttons for 'Stop', 'Play', 'Step', and 'Reset', along with a 'Trigger' field and 'Animation speed' controls. At the bottom, there are tabs for 'Sessions', 'Breakpoints', 'Console', and 'Variables'. The 'Console' tab is active, showing a list of messages:

```

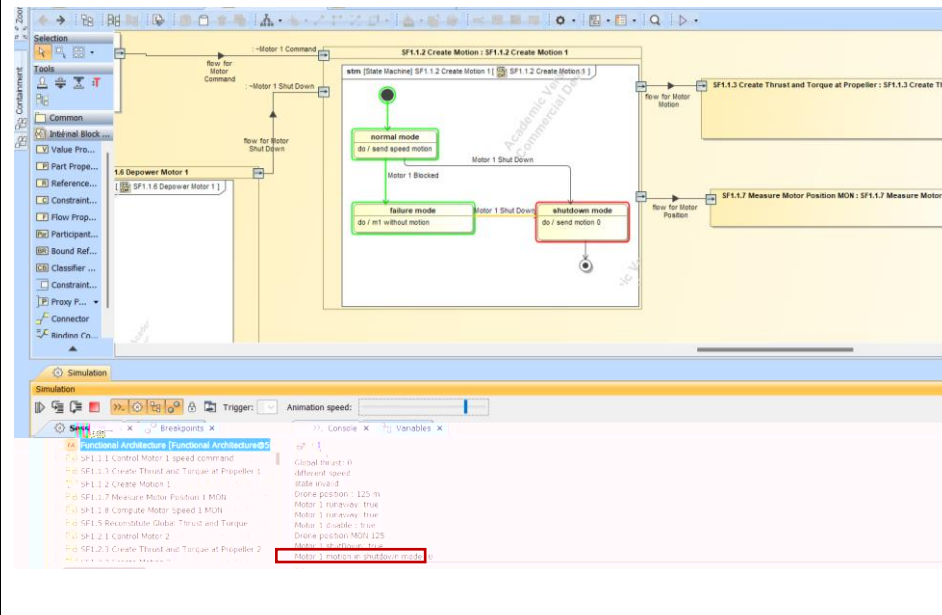
Motor 1 position value in failure mode: 0 state invalid
Propeller 1 thrust : 0
Motor 1 motion value in failure mode: 0 state invalid
Global thrust: 0
different speed
state invalid
Motor 1 runway: true
Motor 1 disable: true
Flight Plan : latitude: 251 longitude: 54 height: 5
Motor 1 shutDown: true
  
```

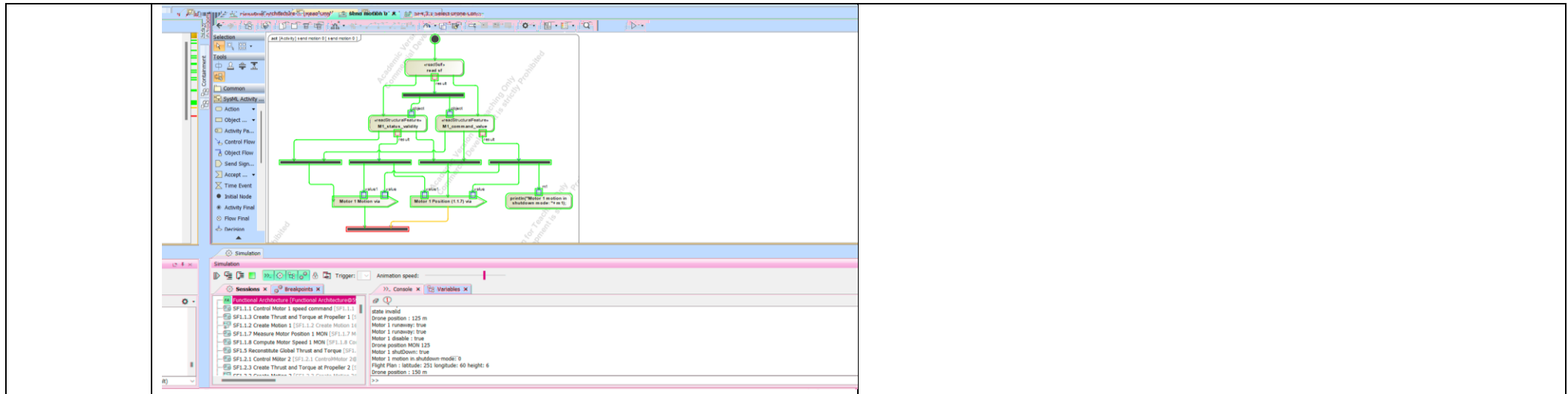
	 <p>The screenshot displays a SysML simulation environment. The main window shows an activity diagram for 'Send M1 shutdown'. The diagram includes nodes for 'read power motor', 'create motion M1_shutdown', and 'Motor 1 Shut Down via'. The console window at the bottom shows simulation logs with various status messages such as 'Motor 1 position value in failure mode: 0 state invalid' and 'Motor 1 shutdown: true'.</p>	
<p>Status:</p>		
<p>Note (opt.):</p>		

(ix) AIDA-XX.YYY-SA-009

	SE	SA
<p>Action:</p>		
<p>Expectation:</p>	<p><i>Motor x motion = 0</i></p>	

Result:



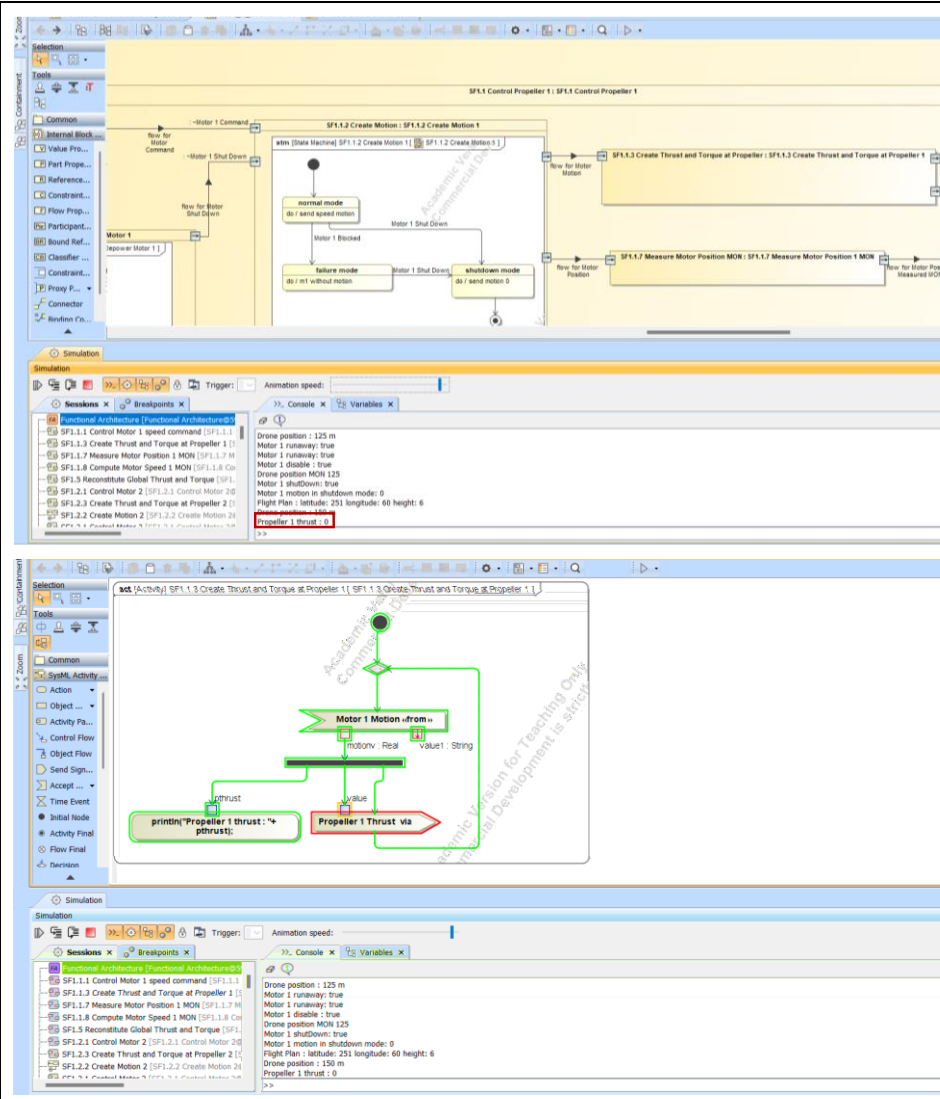


Status:		
Note (opt.):		

(x) AIDA-XX.YYY-SA-010

	SE	SA
Action:		
Expectation:	<i>Propeller x thrust = 0</i>	

Result:



The image displays two screenshots of a software development environment, likely Simulink or a similar tool, used for modeling and simulating a propeller control system.

Top Screenshot: Shows a state machine diagram titled "SF1.1 Control Propeller 1". The diagram includes states such as "normal mode" (with a guard condition "do / send speed motion") and "failure mode" (with a guard condition "do / #1 without motion"). Transitions are labeled with events like "Motor 1 Shud Down" and "Motor 1 Shud Open". The diagram also shows actions for "SF1.1.2 Create Motion 1" and "SF1.1.3 Create Thrust and Torque at Propeller 1".

Simulation Console (Top): Displays the execution flow of the state machine. The current state is "SF1.1.3 Create Thrust and Torque at Propeller 1". The console shows the following variables:

- Drone position : 125 m
- Motor 1 runaway: true
- Motor 1 runway: true
- Motor 1 disable : true
- Drone position MON 125
- Motor 1 shutdown: true
- Motor 1 motion in shutdown mode: 0
- Flight Plan : latitude: 251 longitude: 60 height: 6
- Propeller 1 thrust : 0** (highlighted in red)

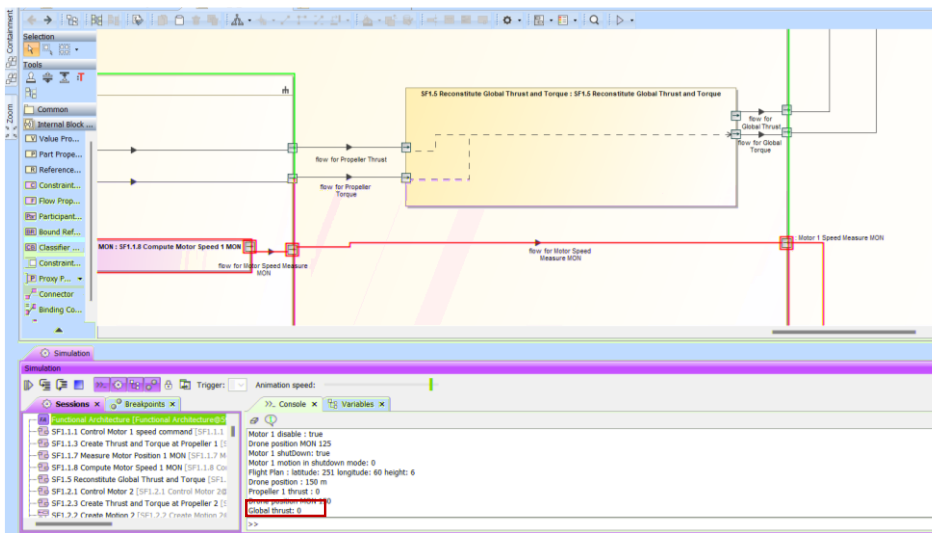
Bottom Screenshot: Shows a detailed view of the "SF1.1.3 Create Thrust and Torque at Propeller 1" state. The diagram illustrates the flow of data from "Motor 1 Motion vfrom" (with a "MotorV: Real" and "Value1: String" input) through a "prop" block to "printing('Propeller 1 thrust: '+ pthrust)" and "Propeller 1 Thrust via".

Simulation Console (Bottom): Shows the state machine's execution flow. The current state is "SF1.1.3 Create Thrust and Torque at Propeller 1". The console shows the following variables:

- Drone position : 125 m
- Motor 1 runaway: true
- Motor 1 runway: true
- Motor 1 disable : true
- Drone position MON 125
- Motor 1 shutdown: true
- Motor 1 motion in shutdown mode: 0
- Flight Plan : latitude: 251 longitude: 60 height: 6
- Drone position : 150 m
- Propeller 1 thrust : 0**

Status:		
Note (opt.):		

(xi) AIDA-XX.YYY-SA-011

	SE	SA
Action:		
Expectation:	Global thrust! = Required Thrust +/- acceptance margin	
Result:		
Status:		
Note (opt.):		

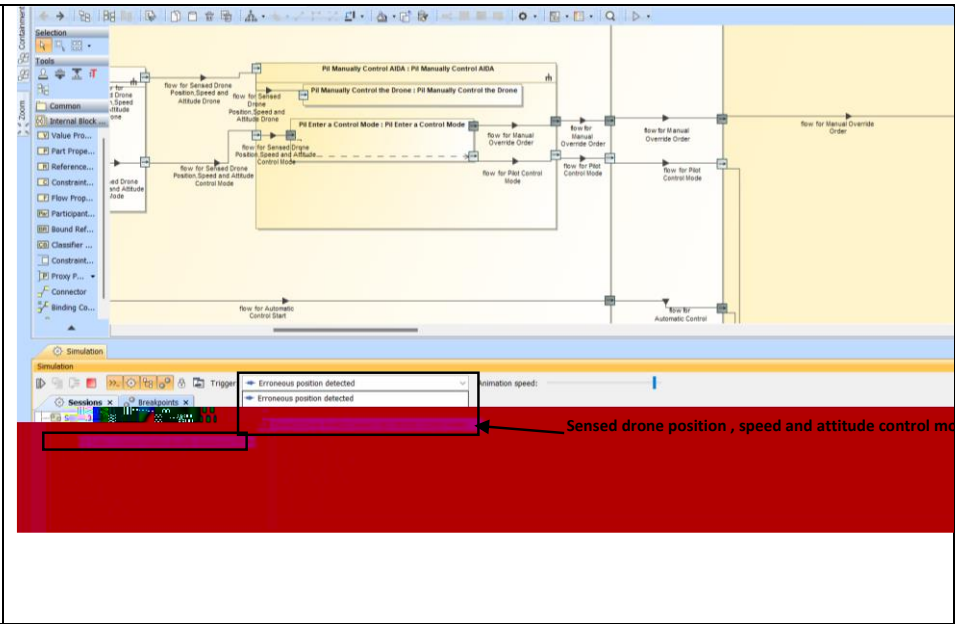


(xii) AIDA-XX.YYY-SA-012

	SE	SA
Action:		
Expectation:	Drone real position = (x,y,z position vector value)	
Result:		
Status:		
Note (opt.):		

(xiii) AIDA-XX-YYY-SA-013

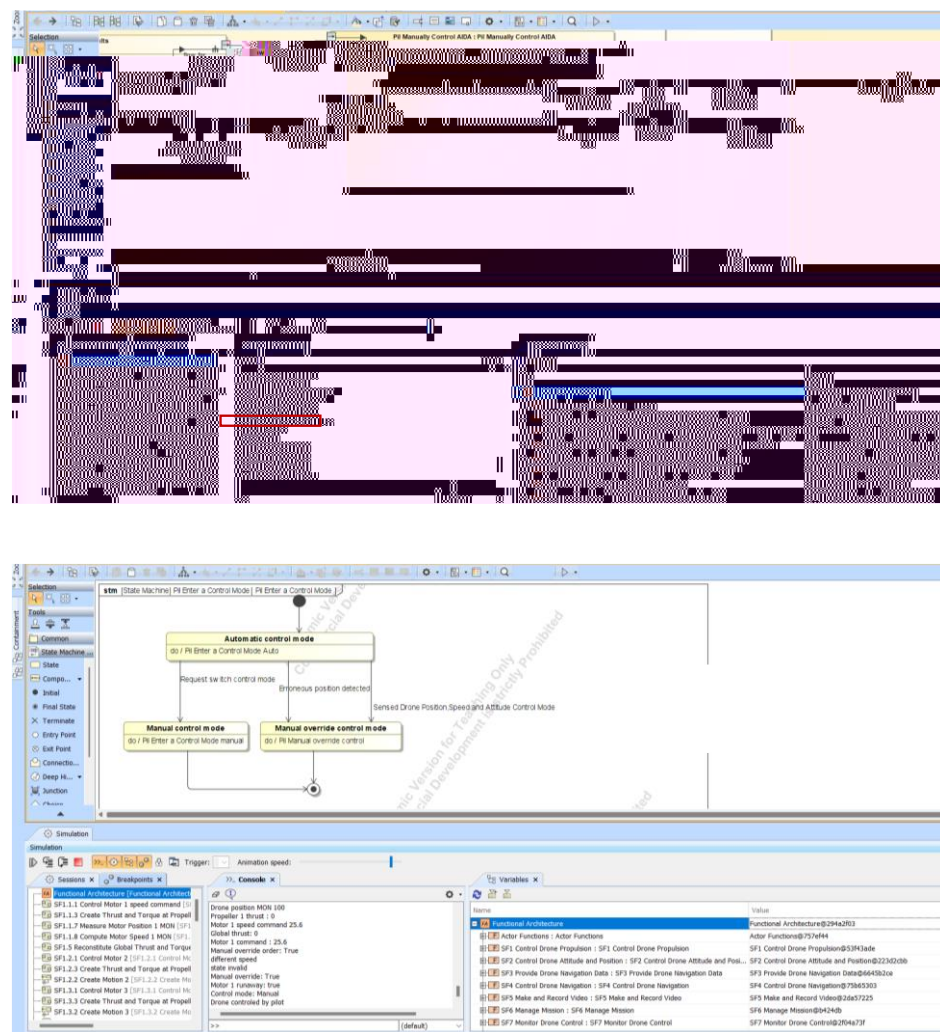
	SE	SA
Action:		
Expectation:	Sensed drone position= (x,y,z position vector value)	
Result:		

		
<p>Status:</p>		
<p>Note (opt.):</p>		

(xiv) AIDA-XX.YYY-SA-014

	SE	SA
<p>Action:</p>		
<p>Expectation:</p>	Manual Override Order = TRUE	

Result:

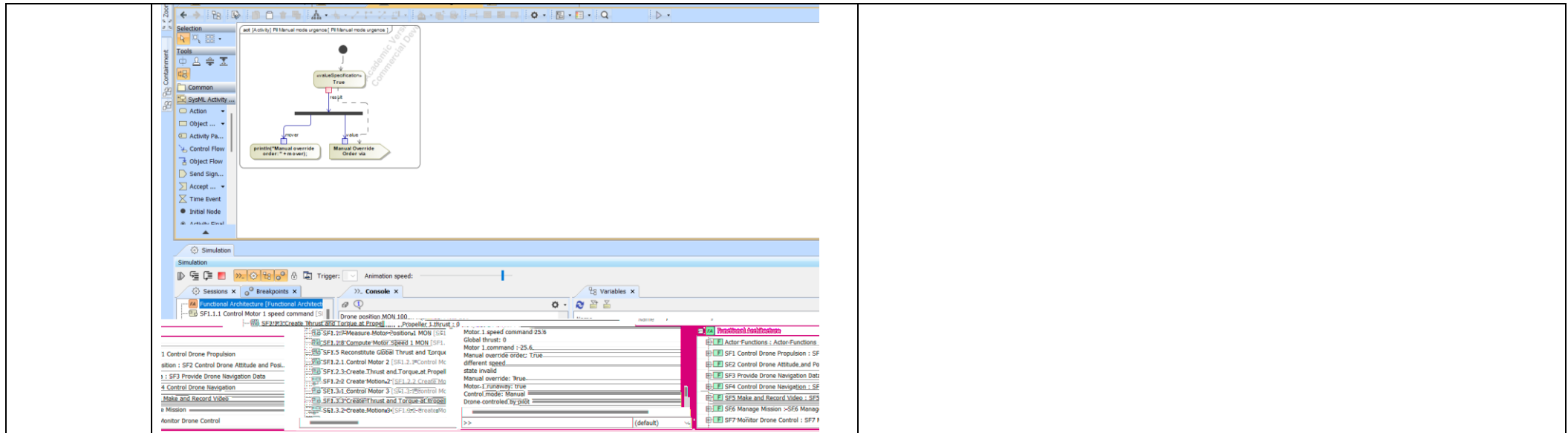


The image displays a simulation environment with two main components:

State Machine Diagram: A state machine diagram titled "Automatic control mode" is shown. It features three states: "Automatic control mode", "Manual control mode", and "Manual override control mode". Transitions are triggered by events such as "Request sw RTH control mode" and "Emergency position detected".

Simulation Console: The console window at the bottom shows a list of simulation sessions and a table of variables. The variables table includes the following data:

Name	Value
Drone position MON 100	Functional Architecture@29142R13
Propeller 1 Speed : 0	Actor Function@72444
Motor 1 speed command 25.6	SF1 Control Drone Propulsion
Global thrust 0	SF1 Control Drone Propulsion@53K1ade
Motor 3 command : 25.6	SF2 Control Drone Attitude and Position
Manual override order: True	SF2 Control Drone Attitude and Position@223d20b
different speed	SF3 Provide Drone Navigation Data
State ready	SF3 Provide Drone Navigation Data@464932ca
Manual override: True	SF4 Control Drone Navigation
Motor 1 Runway: true	SF4 Control Drone Navigation@73653303
Control mode: Manual	SF5 Make and Record Video
Drone controlled by pilot	SF5 Make and Record Video@2657225
	SF6 Manage Mission
	SF6 Manage Mission@424db
	SF7 Monitor Drone Control
	SF7 Monitor Drone Control@269473f

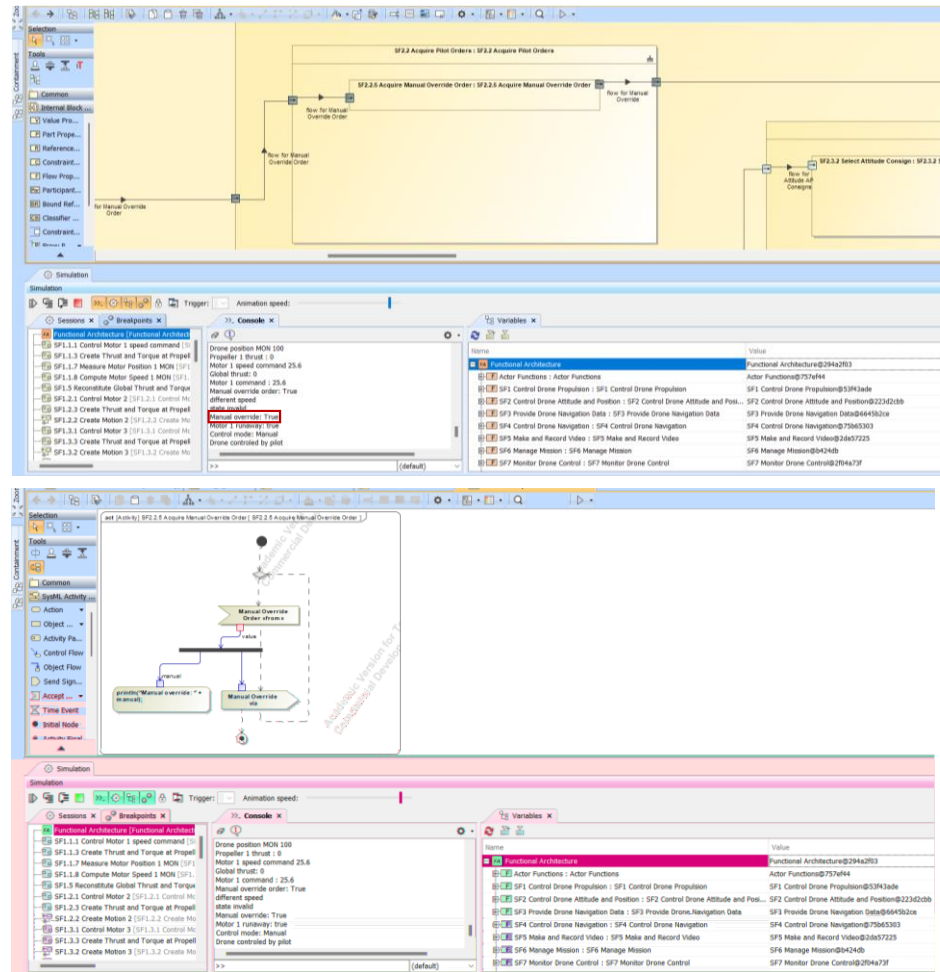


Status:		
Note (opt.):		

(xv) AIDA-XX.YYY-SA-015

	SE	SA
Action:		
Expectation:	Manual Override = TRUE	

Result:



The image displays three screenshots of a simulation software interface, likely Simulink or a similar modeling environment, showing flight control logic and console output.

Top Screenshot: Shows a block diagram of flight control logic. Key blocks include "SF2.2 Acquire Pilot Orders", "SF2.2 Acquire Manual Override Order", and "SF2.2 Select Attitude Consign".

Middle Screenshot: Shows a console window with the following output:

```

Drope position MON 100
Propeller 1 Brunt 1.0
Motor 1 speed command 25.6
Global Thrust: 0
Motor 1 command: 25.6
Manual override order: True
different speed
data invalid
Manual override: True
Control mode: Manual
Drope controlled by pilot
  
```

Bottom Screenshot: Shows a console window with the following output:

```

Drope position MON 100
Propeller 1 Brunt 1.0
Motor 1 speed command 25.6
Global Thrust: 0
Motor 1 command: 25.6
Manual override order: True
different speed
data invalid
Manual override: True
Motor 1 runaway: true
Control mode: Manual
Drope controlled by pilot
  
```

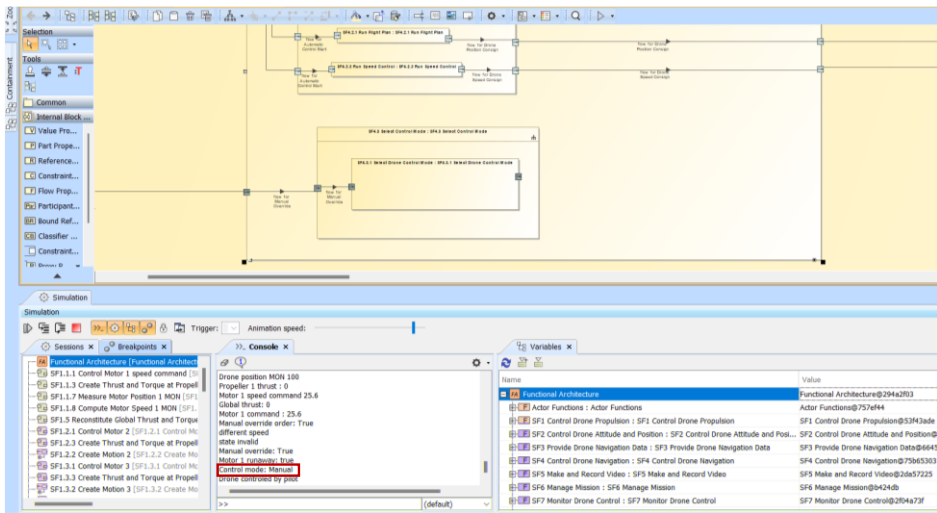
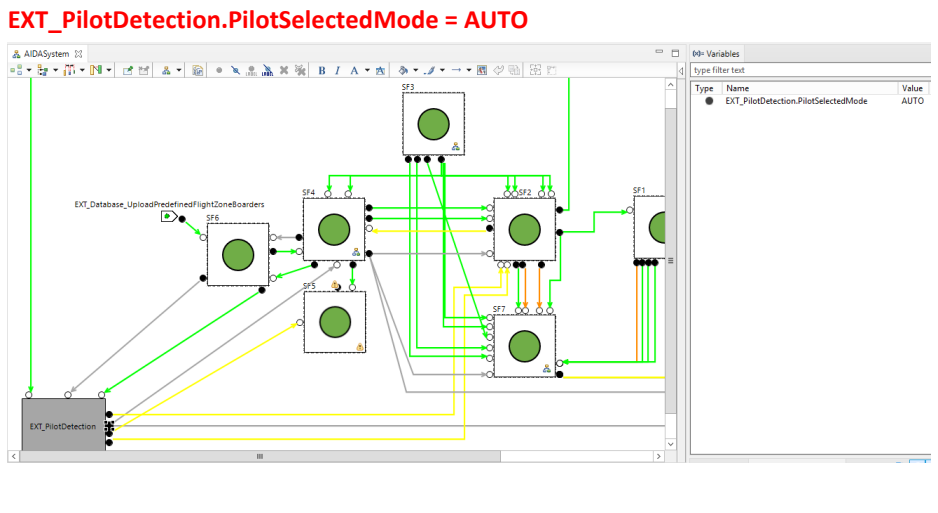
The interface also includes a "Variables" window showing a table of parameters:

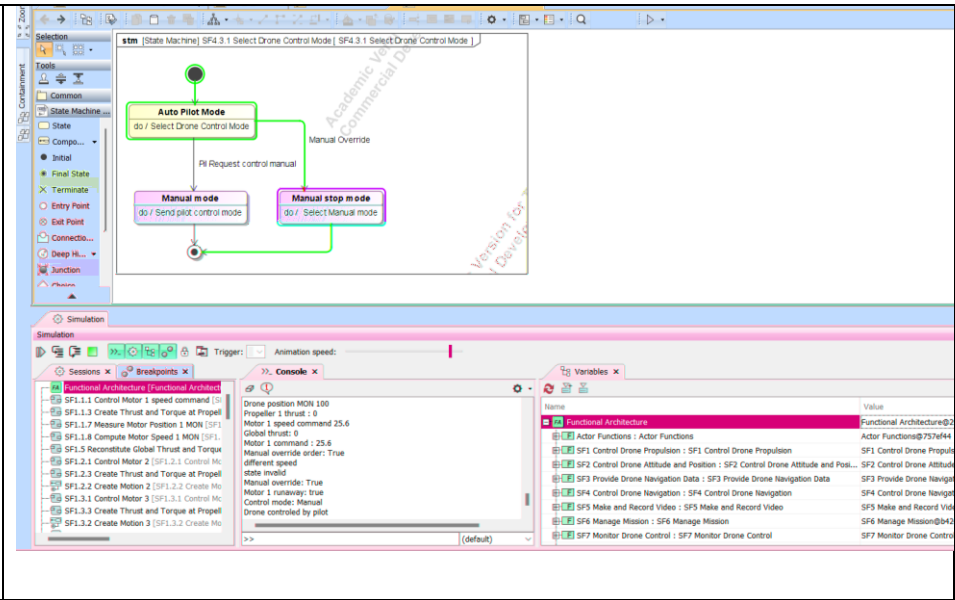
Name	Value
Functional Architecture	Functional Architecture@294a70f3
Actor Functions - Actor Functions	Actor Functions@7574f4
SF1 Control Drone Propulsion - SF1 Control Drone Propulsion	SF1 Control Drone Propulsion@53f43ade
SF2 Control Drone Attitude and Position - SF2 Control Drone Attitude and Pos...	SF2 Control Drone Attitude and Position@22342dbb
SF3 Provide Drone Navigation Data - SF3 Provide Drone Navigation Data	SF3 Provide Drone Navigation Data@644932ca
SF4 Control Drone Navigation - SF4 Control Drone Navigation	SF4 Control Drone Navigation@75045303
SF5 Make and Record Video - SF5 Make and Record Video	SF5 Make and Record Video@24647225
SF6 Manage Mission - SF6 Manage Mission	SF6 Manage Mission@8424db
SF7 Monitor Drone Control - SF7 Monitor Drone Control	SF7 Monitor Drone Control@204a73f

Status:

Note (opt.):		
--------------	--	--

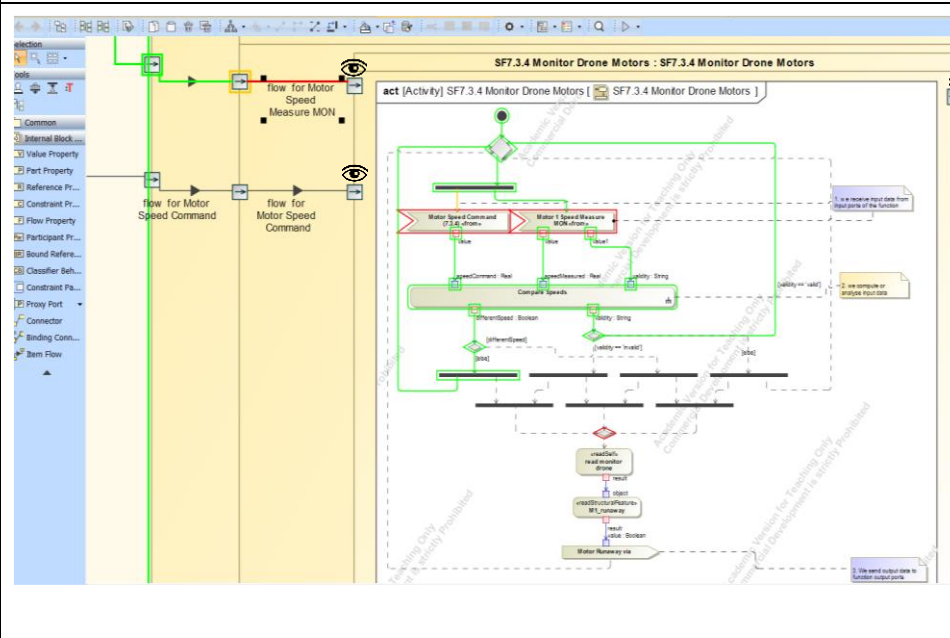
(xvi) AIDA-XX.YYY-SA-016

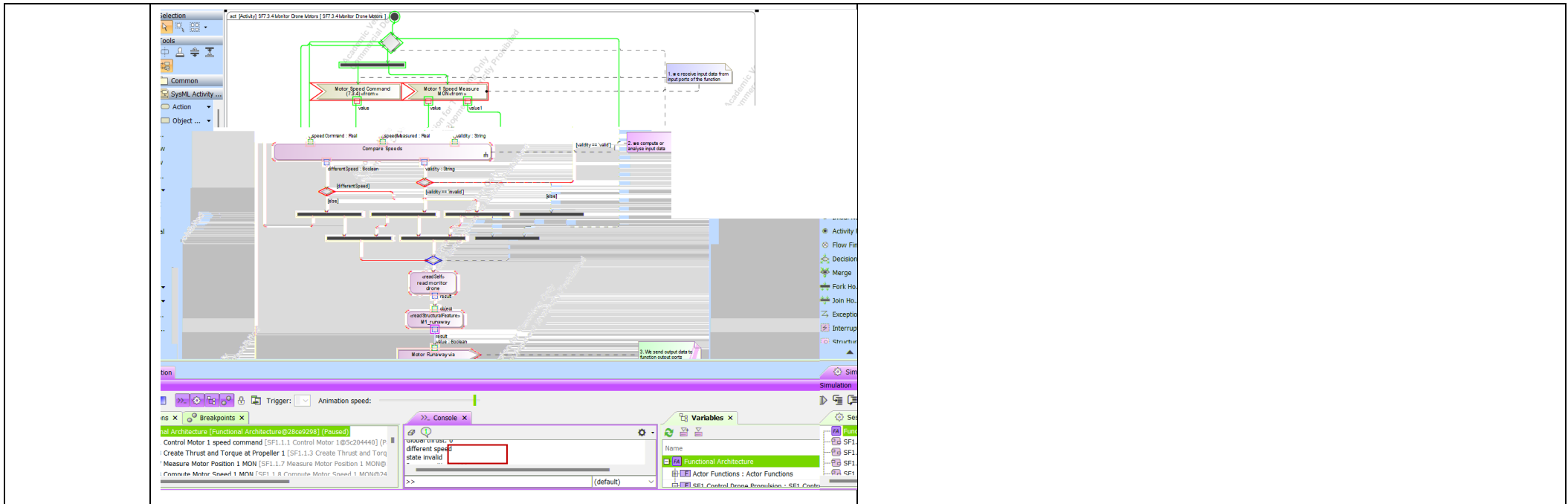
	SE	SA
Action:		
Expectation:	Control mode = MANUAL	EXT_PilotDetection.PilotSelectedMode = MANUAL
Result:		

	 <p>The screenshot shows a state machine diagram with three states: 'Auto Pilot Mode', 'Manual mode', and 'Manual stop mode'. Transitions include 'PI Request control manual' from Auto Pilot to Manual mode, and 'Manual Override' from Manual mode to Manual stop mode. The simulation console below shows logs for SF1.1 through SF1.12, including motor speed commands and manual override status. A variables table on the right lists functional architecture components like Actor Functions, SF1-SF7, and their values.</p>	
<p>Status:</p>		
<p>Note (opt.):</p>		

(xvii) AIDA-XX.YYY-SA-xxx (optional)

	<p>SE</p>	<p>SA</p>
<p>Action:</p>		

<p>Expectation:</p>	<p>Motor x speed measure COM! = Motor x speed measure MON</p>	
<p>Result:</p>		



The screenshot displays a SysML modeling environment. The main workspace shows a functional architecture diagram for a motor control system. Key components include:

- Motor Speed Command (Function):** Receives input data from the function's inputs.
- Motor Speed Measure (Function):** Receives input data from the function's inputs.
- Compare Speeds (Function):** Analyzes input data to determine if speeds are different.
- Control Logic:** A series of decision diamonds and flow lines that manage the state of the system based on speed comparisons.
- Motor Runaway via (Function):** Outputs data to the function's output ports.

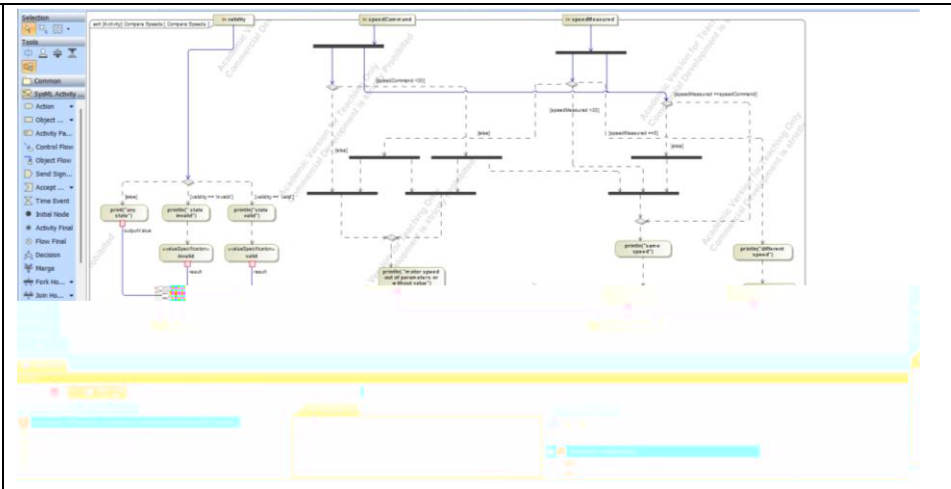
The interface includes a left-hand toolbar with various modeling tools, a top menu bar, and a bottom panel with a console window showing execution logs and a variables window.

Console Log:

```
Functional Architecture (Functional Architecture@28ce9298) [Paused]
Control Motor 1 speed command [SF1.1.1 Control Motor 1@5c204440] (P)
  Create Thrust and Torque at Propeller 1 [SF1.1.3 Create Thrust and Torq
  Measure Motor Position 1 MON [SF1.1.2 Measure Motor Position 1 MON@
  Promote Motor Speed 1 MON [SET 1.1.1 Promote Motor Speed 1 MON@2024
```

Variables Window:

Name
Functional Architecture
Actor Functions : Actor Functions
SET 1 Control Drone Evolution : SET 1 Contr

		
<p>Status:</p>		
<p>Note (opt.):</p>		

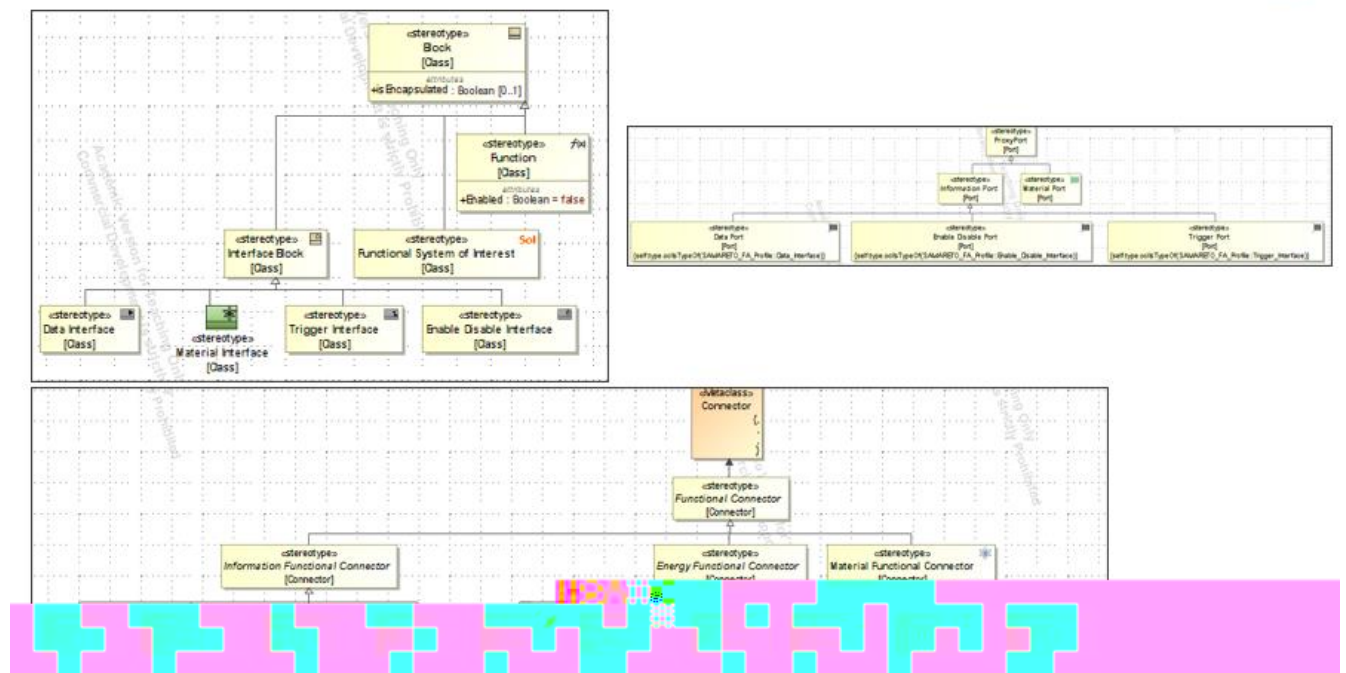
(c) PoC Conclusion

6 Conclusion

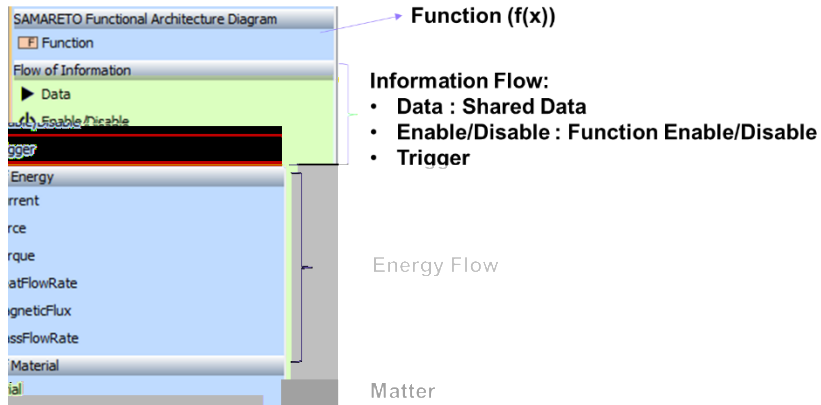
7 Appendices

7.1 SAMAREQ Profile

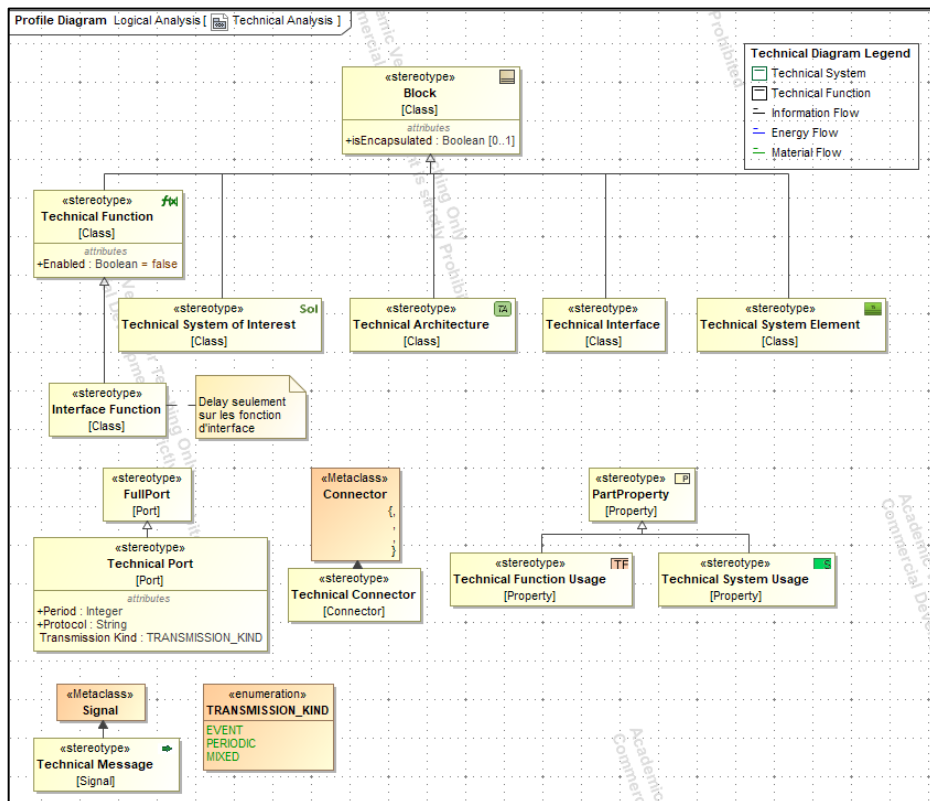
7.1.1 SAMAREQ Functional Architecture Profile



7.1.2 SAMAREQ Diagram Customization and Palette



7.1.3 Logical Architecture Profile



7.2 Derived Result : Consistency Check Capella Model and Cameo SysML Model

