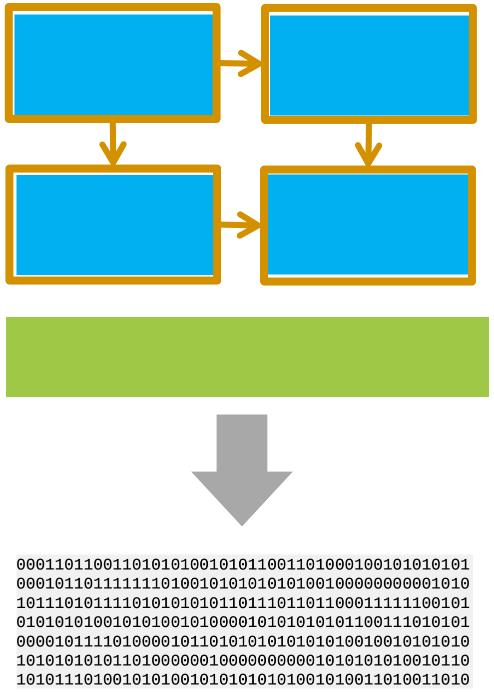
ARCHITECTURE-DRIVEN ASSURANCE

JENNIFER DAVIS, PH.D.
TRUSTED SYSTEMS
OCTOBER 2019

Collins Aerospace

APPROACH: ARCHITECTURE-DRIVEN ASSURANCE

1. Architecture model is correct
   - Properties, structure, behavior, interaction of components, interfaces, contracts
   - Analyzable
2. Components are correct
   - Consistent/realizable contracts
   - Components verified to implement contracts
3. System does what the model says
   - No other information flows (memory safety, isolation)
   - OS executes model correctly (incl. timing)
4. System implementation corresponds to model
   - Automatic build from component and architecture models



SELECTED PROGRAMS LEVERAGING ARCHITECTURE-DRIVEN ASSURANCE

- DARPA High Assurance Cyber Military Systems (HACMS)

- DARPA Cyber Assured Systems Engineering (CASE)

HIGH ASSURANCE CYBER MILITARY SYSTEMS (HACMS) TECHNOLOGIES

Open source tools, languages, software

1. Architecture modeling and analysis tools (AADL)
   - Assume-Guarantee Reasoning Environment (AGREE)
   - Architecture-based assurance cases (Resolute)
2. Ivory/Tower embedded Domain Specific Languages
   - Memory safe component software
   - Code generation from high-level specification
3. seL4 formally verified OS kernel
   - Isabelle/HOL proof of correctness
   - Security properties proven to binary level
4. Automated build from models
   - Support for seL4, eChronos, VxWorks, Linux

DARPA

Rockwell Collins

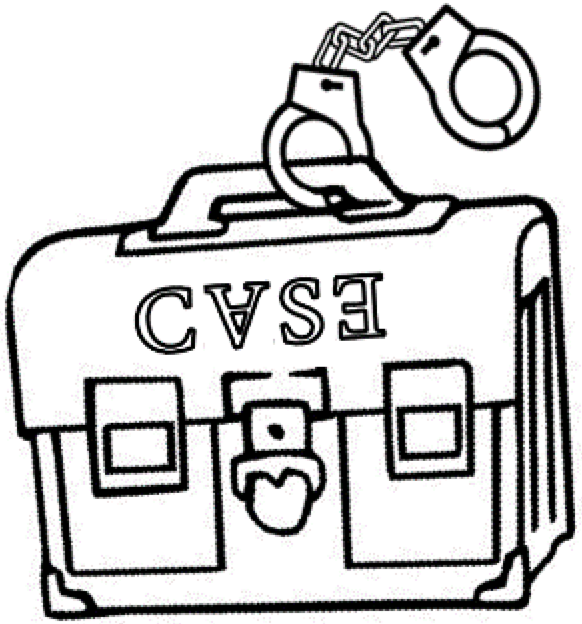
galois

DATA 61

University of Minnesota
Driven to Discover

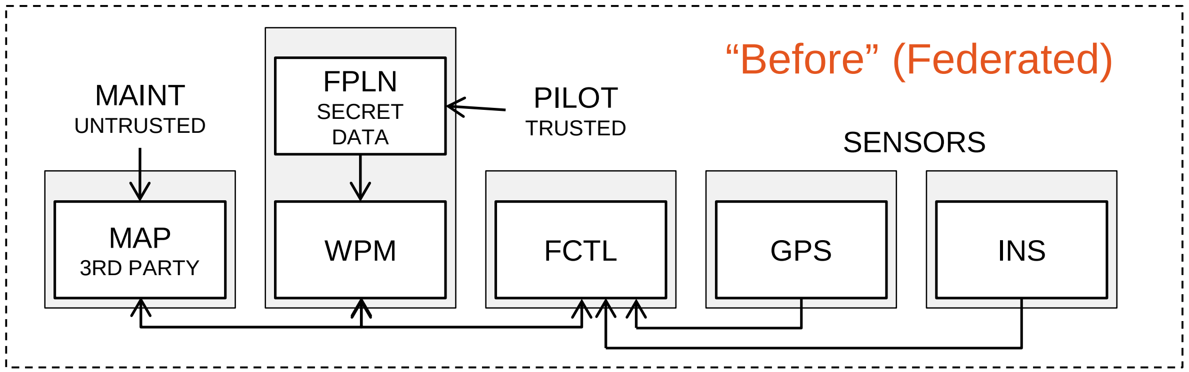
CYBER ASSURED SYSTEMS ENGINEERING (CASE)

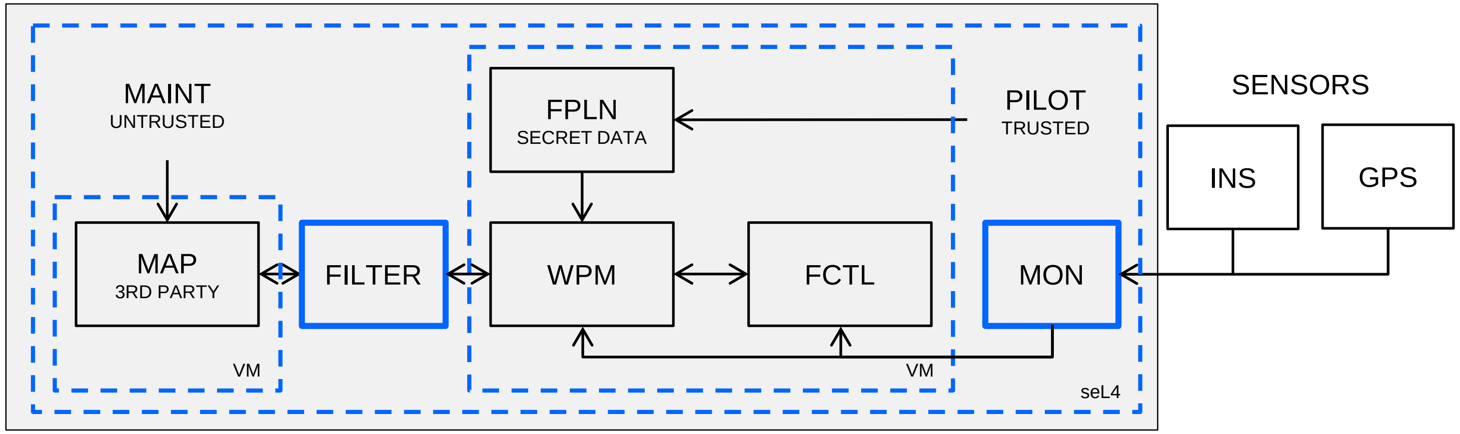
- The goal of CASE is to develop the necessary design, analysis and verification tools to allow system engineers to design-in cyber resiliency and manage tradeoffs as they do the other non-functional properties when designing complex embedded computing systems
    - Cyber resiliency means that the system is tolerant to cyberattacks in the same way that safety critical systems are tolerant to random faults – they recover and continue to execute their mission function
    - Cyber security requirements are addressed today by penetration testing late in the development, resulting in expensive rework
    - Cyber requirements are often “shall not” statements about the system, and so are not testable (formal methods required)



APPROACH

- Start with initial design, new or legacy
  - Federated avionics system
- Generate new cyber requirements
  - Possibly based on modified system architecture
- Tool-assisted transformations of system architecture
  - Satisfy cyber requirements
  - Manage other design trade-offs
  - Insertion/synthesis of high-assurance components may be needed
- Verification of cyber resiliency
- Generate system from architecture model

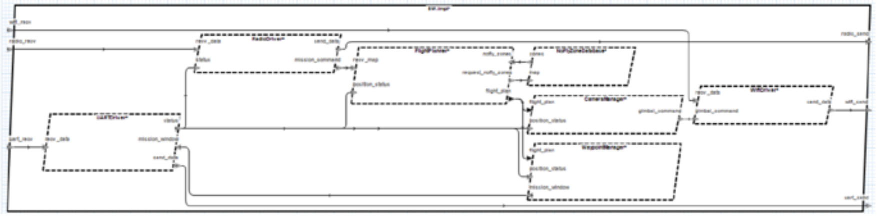


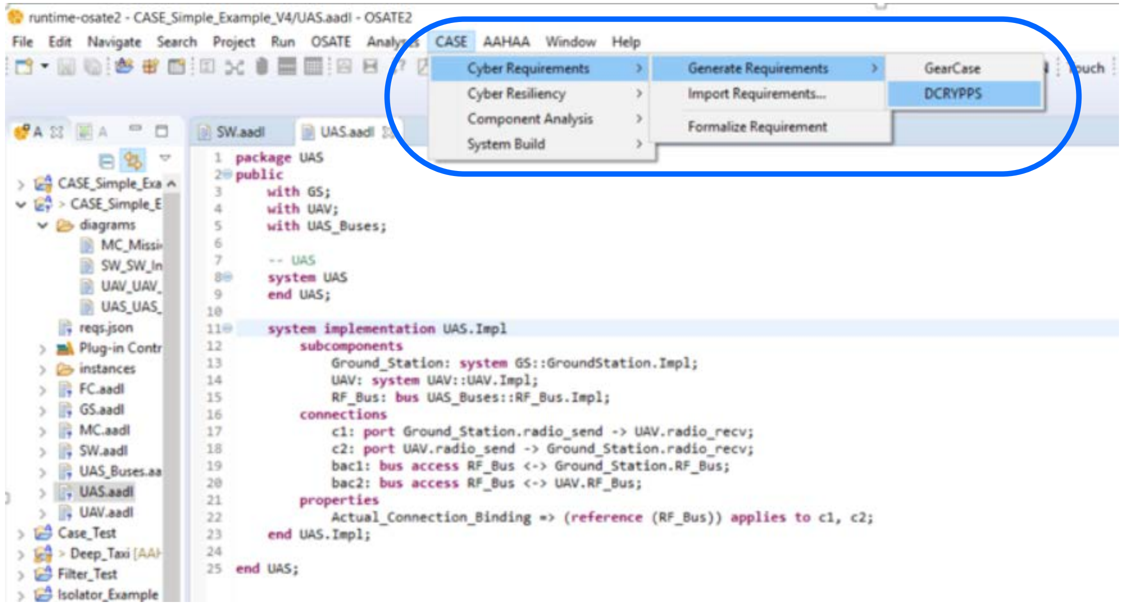


“After” (Integrated, Cyber-resilient)

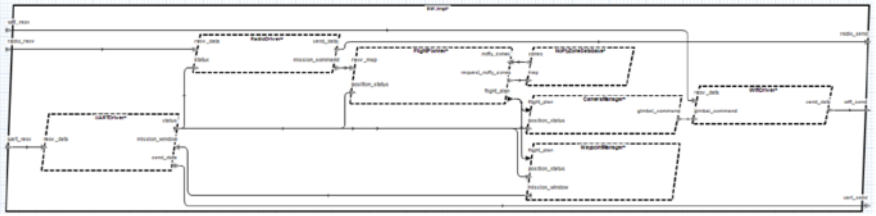
Collins Aerospace

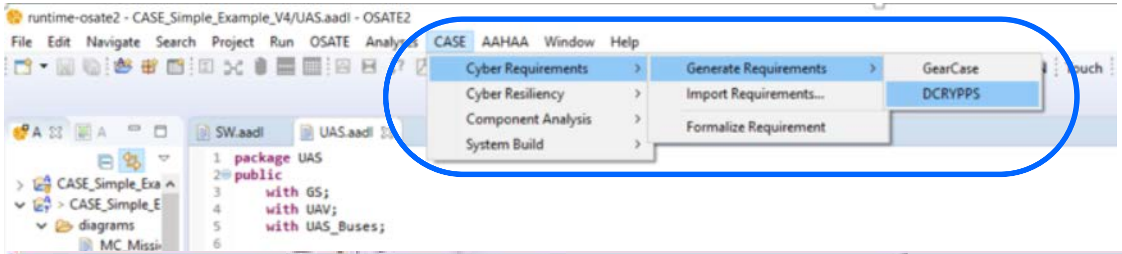
CYBER REQUIREMENTS

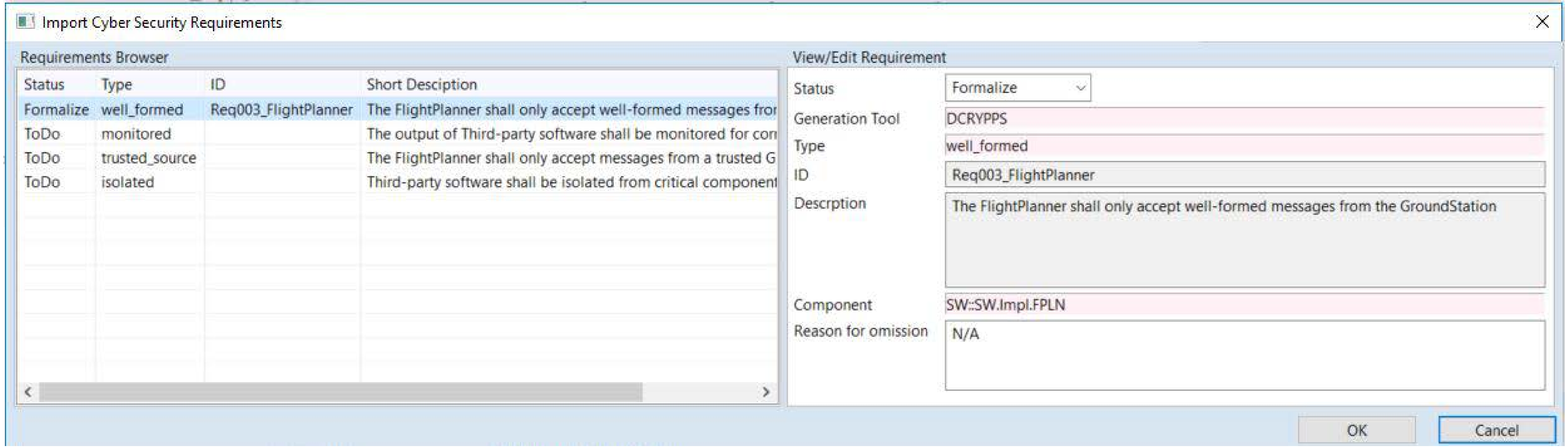




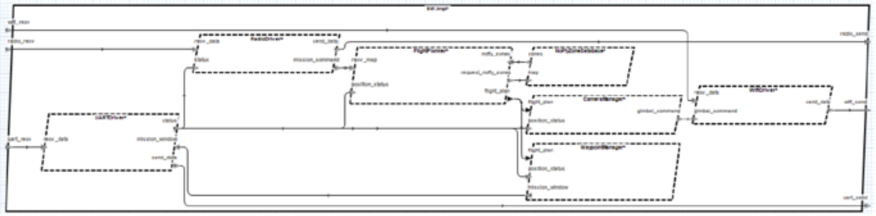
CYBER REQUIREMENTS







CYBER REQUIREMENTS



runtime-osate2 - CASE_Simple_Example_V4/UAS.aadl - OSATE2

File Edit Navigate Search Project Run OSATE Analyses CASE AAHAA Window Help

Cyber Requirements > Generate Requirements > GearCase / DCRYPPS
Cyber Resiliency > Import Requirements...
Component Analysis > Formalize Requirement
System Build

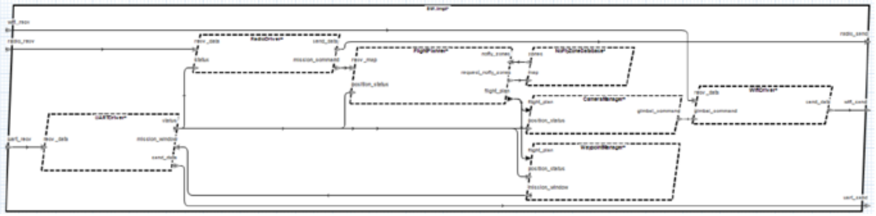
SW.aadl | UAS.aadl

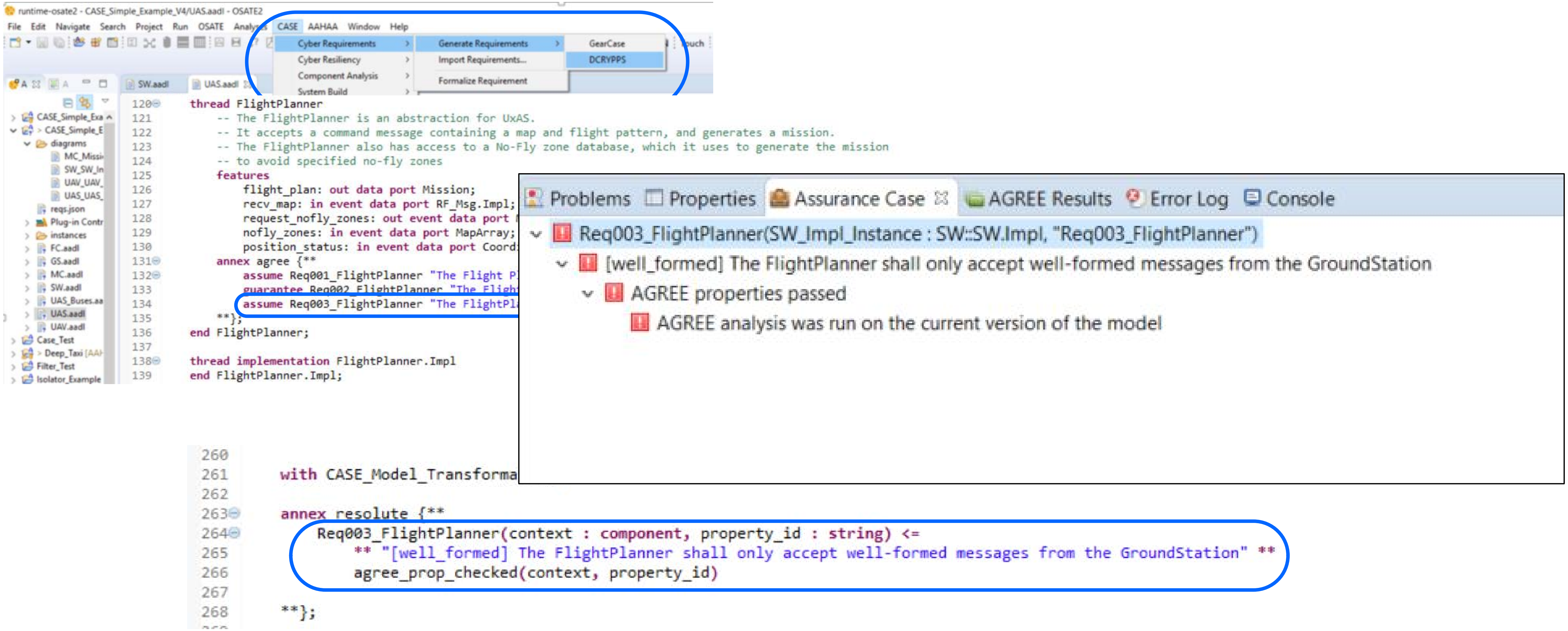
CASE_Simple_Exa
CASE_Simple_E
diagrams
MC_Missi
SW_SW_In
UAV_UAV_
UAS_UAS_
reqs.json
Plug-in Contr
instances
FC.aadl
GS.aadl
MC.aadl
SW.aadl
UAS_Buses.aa
UAS.aadl
UAV.aadl
Case_Test
Deep_Taxi [AAH
Filter_Test
Isolator_Example

```
thread FlightPlanner
    -- The FlightPlanner is an abstraction for UxAS.
    -- It accepts a command message containing a map and flight pattern, and generates a mission.
    -- The FlightPlanner also has access to a No-Fly zone database, which it uses to generate the mission
    -- to avoid specified no-fly zones
    features
        flight_plan: out data port Mission;
        recv_map: in event data port RF_Msg.Impl;
        request_nofly_zones: out event data port Map;
        nofly_zones: in event data port MapArray;
        position_status: in event data port Coordinate.Impl;
    annex agree {**
        assume Req001_FlightPlanner "The Flight Planner shall receive a valid message from the Ground Station" : VALID_MESSAGE(recv_map);
        guarantee Req002_FlightPlanner "The Flight Planner shall generate a valid mission" : good_mission(flight_plan);
        assume Req003_FlightPlanner "The FlightPlanner shall only accept well-formed messages from the GroundStation" : WELL_FORMED_MESSAGE(recv_map);
    **};
end FlightPlanner;

thread implementation FlightPlanner.Impl
end FlightPlanner.Impl;
```

```

with CASE_Model_Transformations;

annex resolute {**
    Req003_FlightPlanner(context : component, property_id : string) <=
        ** "[well_formed] The FlightPlanner shall only accept well-formed messages from the GroundStation" **
        agree_prop_checked(context, property_id)

**};
```

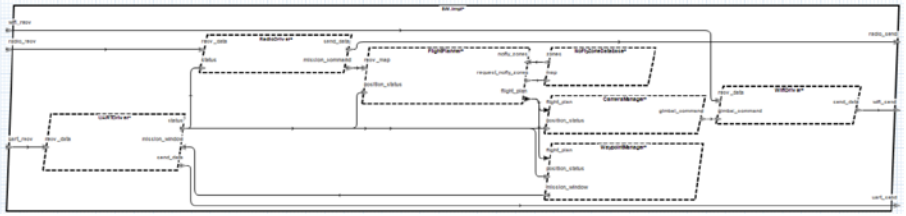
CYBER REQUIREMENTS

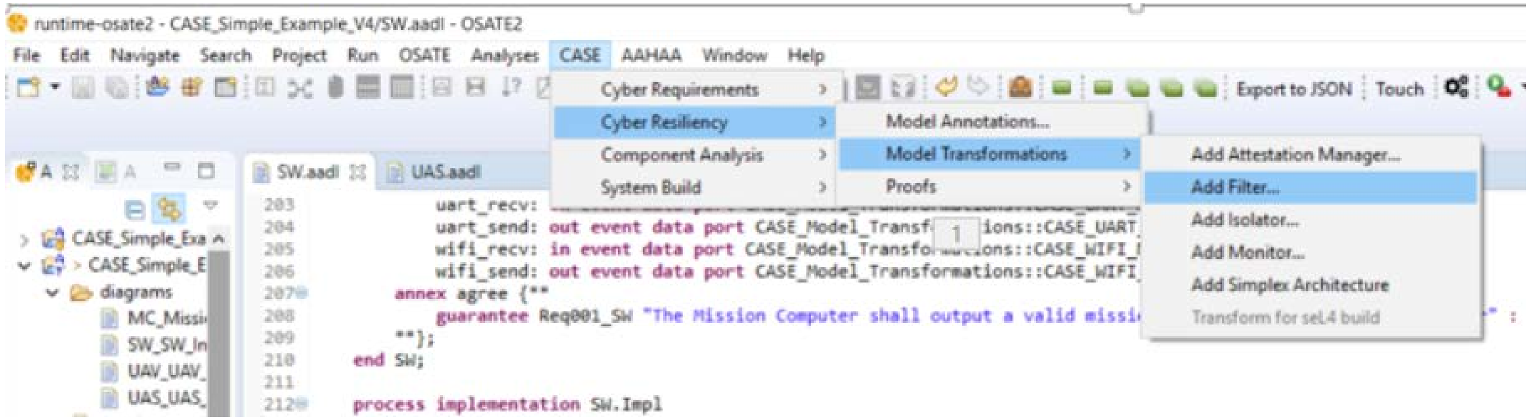




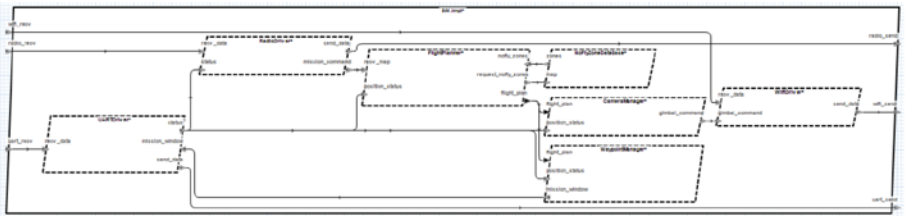
Collins Aerospace

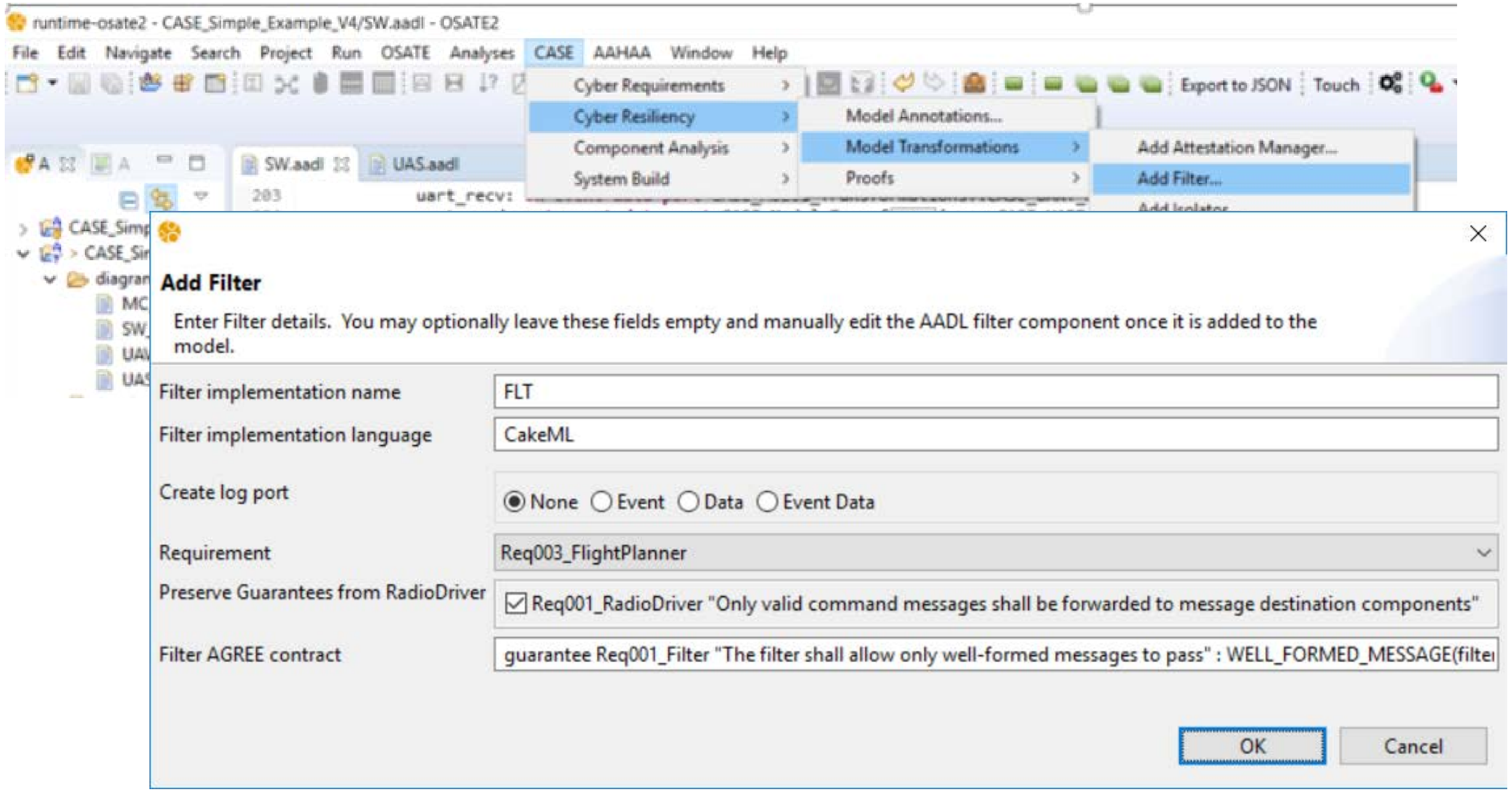
TRANSFORMATION:
FILTER ADDED TO AADL MODEL





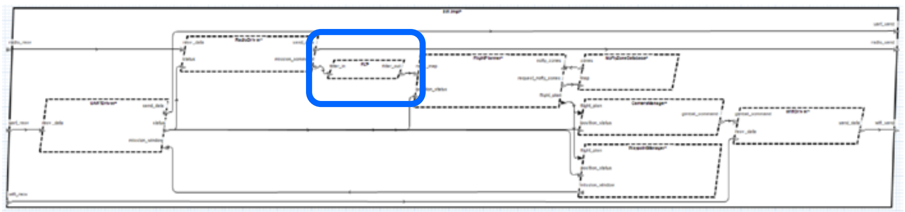
TRANSFORMATION:
FILTER ADDED TO AADL MODEL

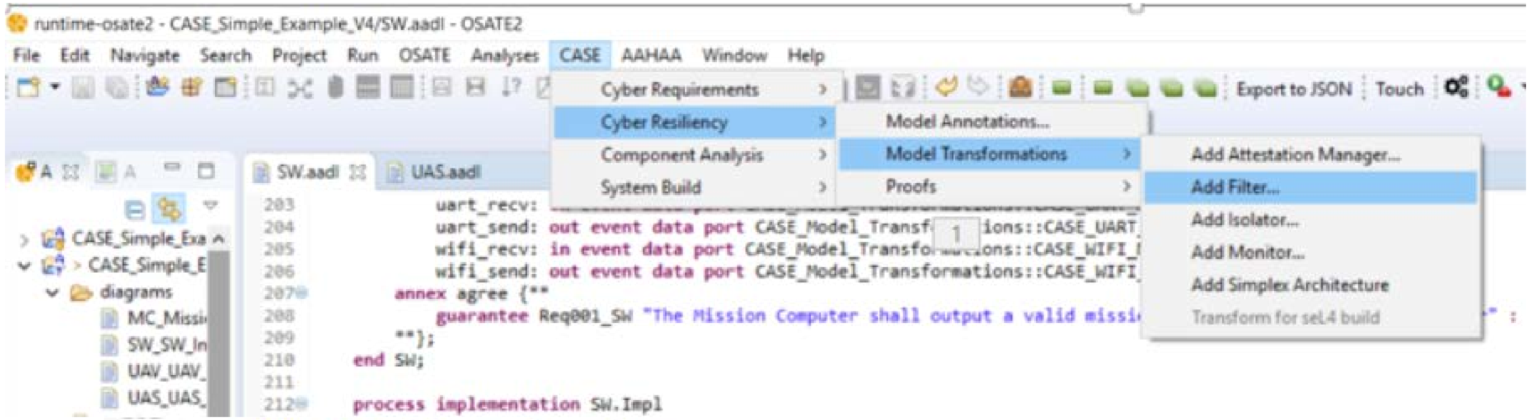




Collins Aerospace

TRANSFORMATION:
FILTER ADDED TO AADL MODEL

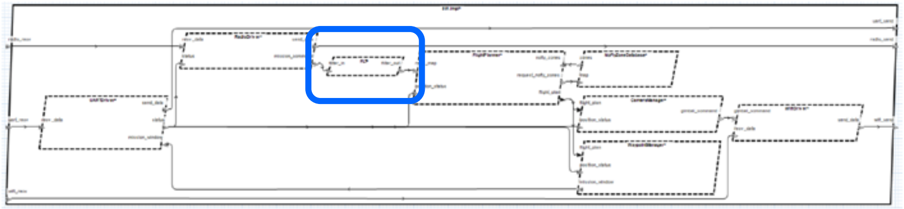


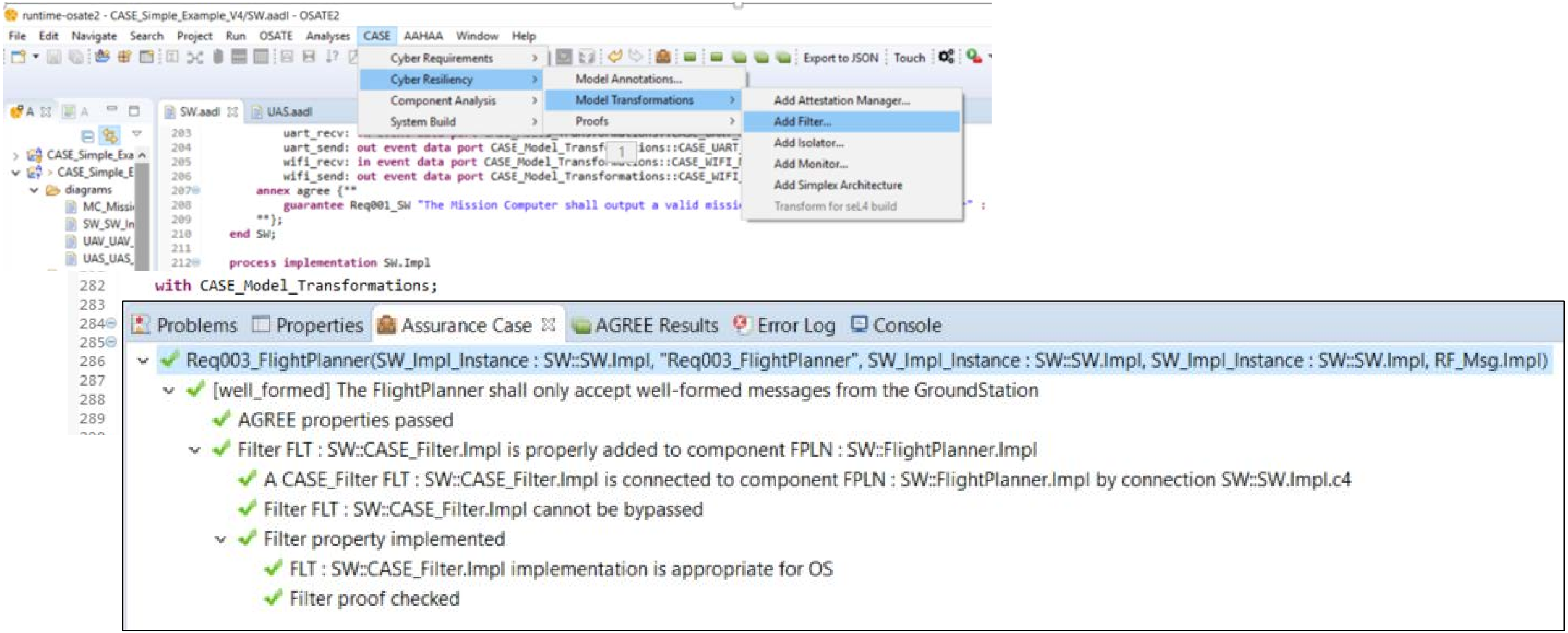



```
with CASE_Model_Transformations;

annex resolute {**
    Req003_FlightPlanner(context : component, property_id : string, filter : component, connection_name : string, message_type : data) <=
        ** "[well_formed] The FlightPlanner shall only accept well-formed messages from the GroundStation" **
        agree_prop_checked(context, property_id) and add_filter(context, filter, connection_name, message_type)

**};
```

```
-------------------------------
-- MODEL TRANSFORMATIONS --
-------------------------------

-- Top-level claim for proper insertion of a filter
add_filter(context : component, filter : component, conn_name : string, msg_type : data) <=
    ** "Filter " filter " is properly added to component " context **
    filter_exists(filter, context, conn_name) and filter_not_bypassed(filter, context, msg_type) and filter_prop_checked()
```

TRANSFORMATION:
FILTER ADDED TO AADL MODEL





CYBER RESILIENT ARCHITECTURE PATTERNS

- Library of general, tool-assisted *architecture model transformations* that mitigate vulnerabilities or address cyber requirements
- Automatic insertion and verification of transform properties as assume-guarantee contracts and assurance case claims
- Examples
  - Filter ✅
  - Attestation ✅
  - Isolation ✅
  - Monitor/Simplex ✅
  - Distributed Action (e.g., Zeroize) ✔️
  - seL4 implementation ✔️

Collins Aerospace

CASE TARGETS

- Experimental platform: AFRL UxAS
- Demonstration platform: CH-47 CAAS

FOR MORE INFORMATION…

- Send me an email at jen.davis@collins.com
- Releases of CASE tool suite (includes Resolute and AGREE) https://github.com/loonwerks/formal-methods-workbench/releases
- HAMR (for System Build): https://github.com/sireum/hamr-plugin-update-site
- More information on CASE project at http://loonwerks.com/projects/case.html
- Tool descriptions and papers
  - Resolute: http://loonwerks.com/tools/resolute.html
  - AGREE: http://loonwerks.com/tools/agree.html
- The seL4 Microkernel: https://sel4.systems/

The tools are open source and free to use!

Collins Aerospace