CYBER-RESILIENT ARCHITECTURE PATTERNS

HIGH CONFIDENCE SYSTEMS AND SOFTWARE
30 APRIL 2019

DR. DARREN COFER
TRUSTED SYSTEMS
DARREN.COFER@COLLINS.COM

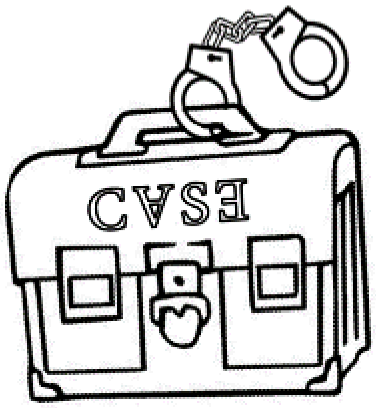
Collins Aerospace

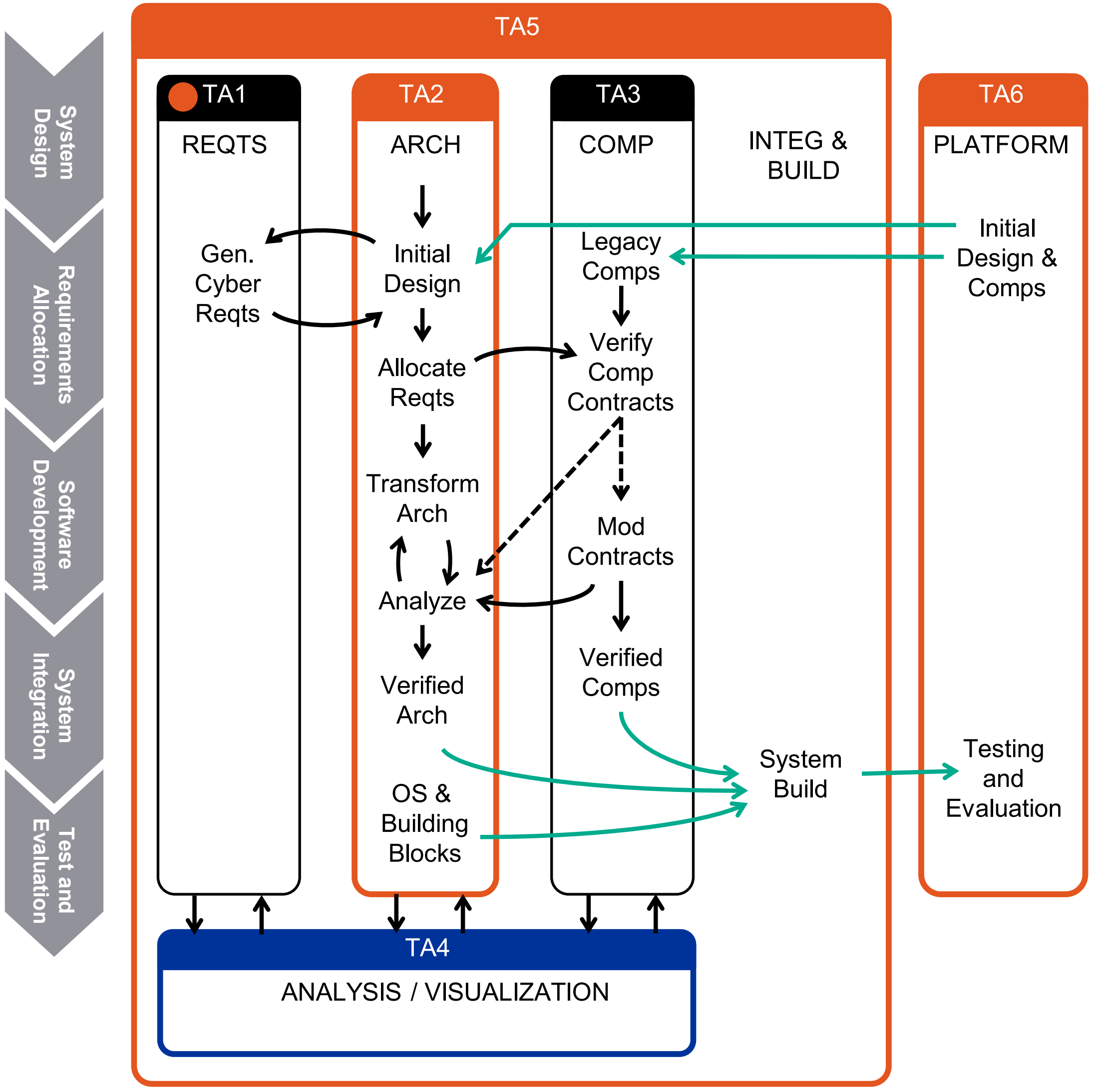
CYBER ASSURED SYSTEMS ENGINEERING (CASE)

- 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)

CYBER-ASSURED SYSTEMS ENGINEERING

- Innovate
  - Generate cyber-security requirements
  - Cyber-resiliency transformations
  - Verified architectural components (CakeML)
  - New analysis tools
- Cultivate (extend and mature)
  - OSATE/AADL modeling
  - AGREE analysis
  - Resolute assurance cases
- Integrate
  - Tool integration
  - Proof integration
  - System build for seL4
- Demonstrate
  - Phase 1: Simple UAV
  - Phase 2: UxAS + challenge problems
  - Phase 3: CH-47/CAAS + challenge problems

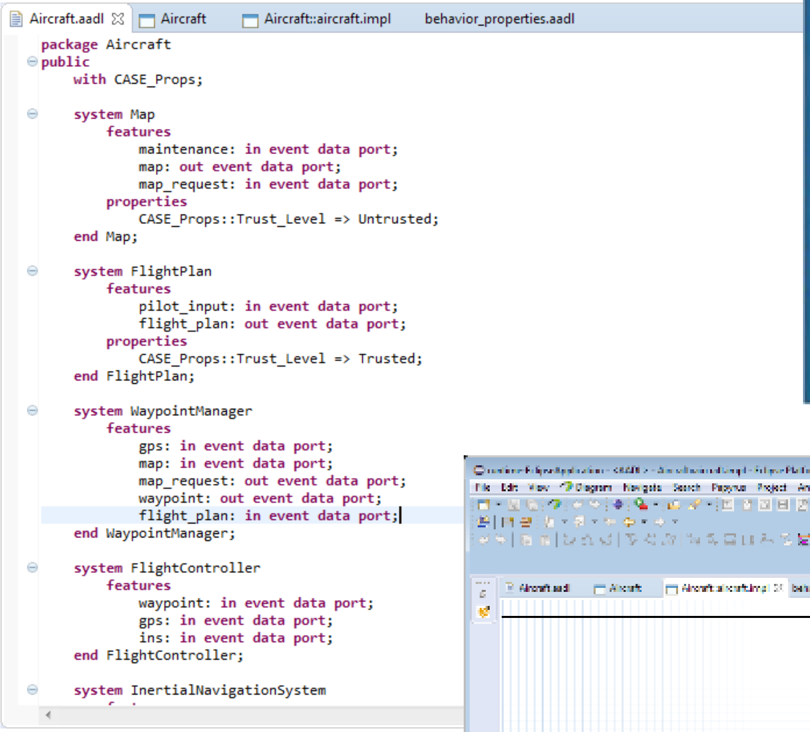


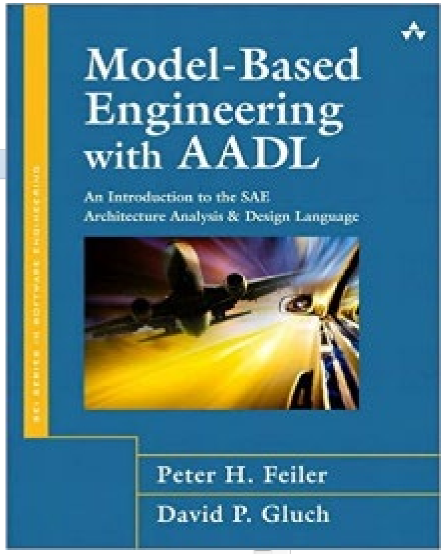


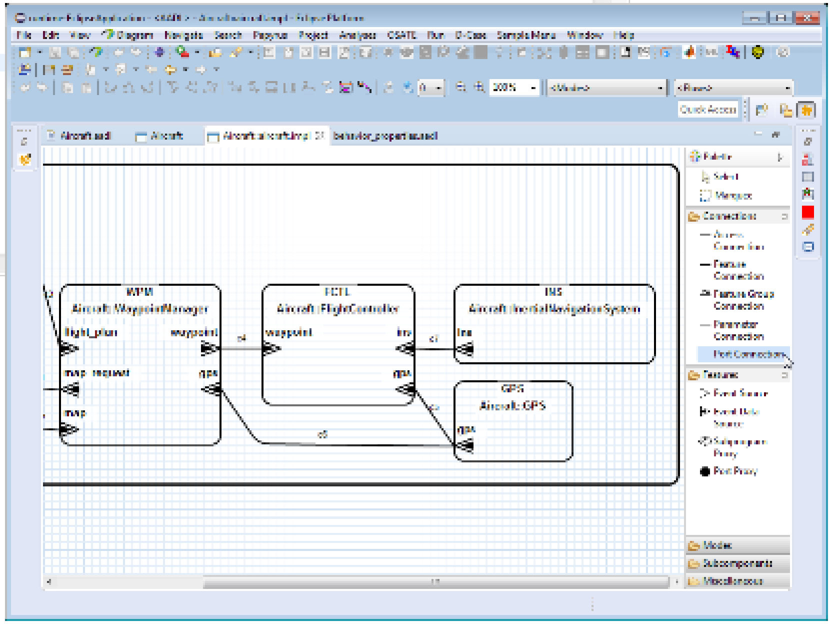
Build revolutionary design resiliency tools for systems engineers that provide cyber assurance for critical DoD systems

ARCHITECTURE ANALYSIS AND DESIGN LANGUAGE (AADL)

- SAE AS5506 standard
- Embedded, real-time, distributed systems
- Physical hardware
  - processors, buses, memory, devices
- Application software
  - software functions, data, threads, processes
- Extendable syntax (annex)
- Open source tools, supported by SEI
  - Open Source AADL Tool Environment (OSATE)





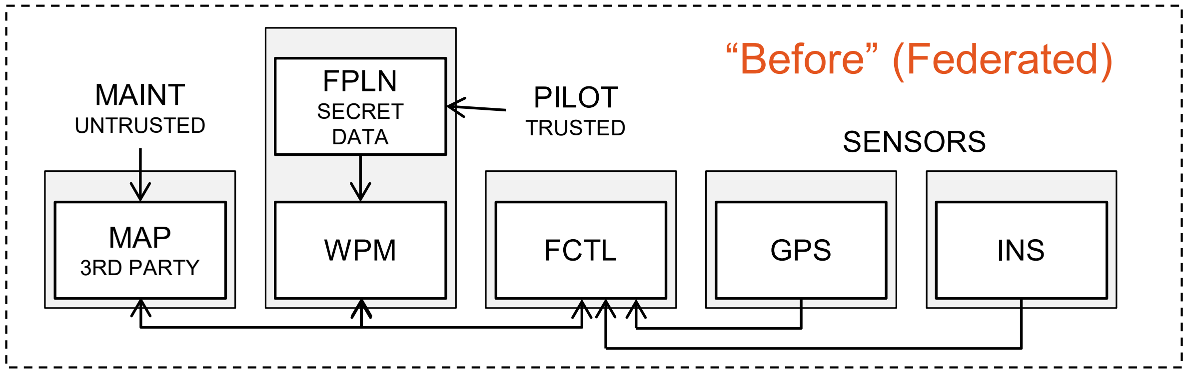


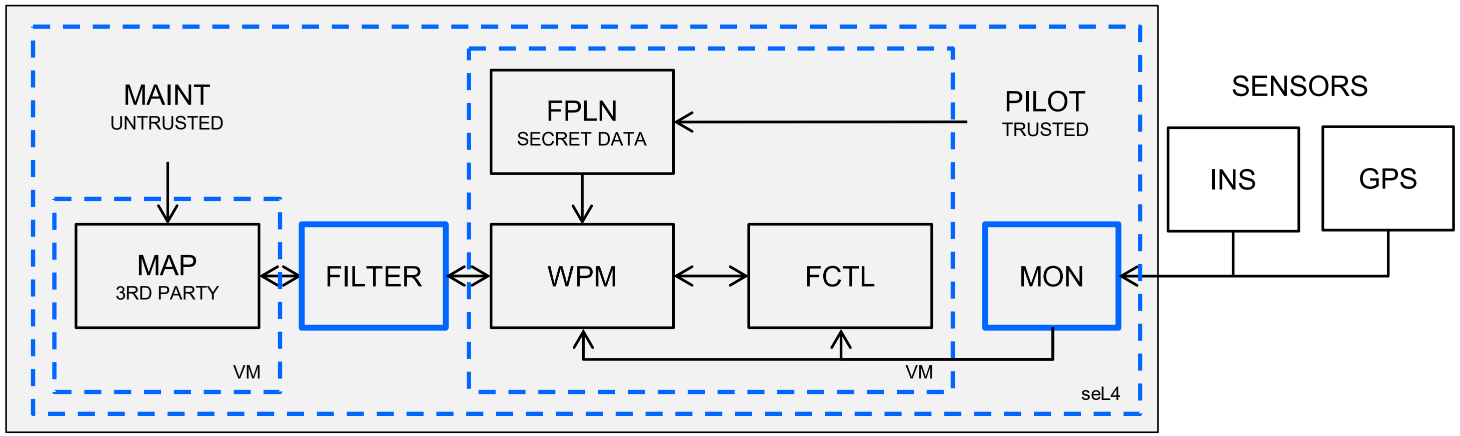
- Sufficiently rigorous semantics to support analysis
- Correct level of abstraction (supports construction)
- Syntax allows addition of new capabilities

Collins Aerospace

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)

1 : CYBER RESILIENCY REQUIREMENTS

- TA1 tools generate cyber requirements based on initial system model (AADL) and (possibly) functional requirements
- TA2 evaluation is still needed to
  - Determine applicability
  - Add additional details
- Assurance case provides a mechanism to receive, implement, and manage cyber requirements and attach them to relevant parts of the design model
- Allows us to specify exactly what evidence is necessary to satisfy each cyber requirement

```
"attack": replay_attack

"mitigation":
Context:MissionComputer.impl

   preventSpoofing(c6)

   preventSpoofing(c : connection) <=
   ** "spoofing of communication on c is prevented" **
   true

**************

"attack": command_injection

"mitigation":
Context: MissionComputer.impl

   well_formed(UARTDriver,WaypointManager,c6)

   permitWellFormedData(s1 : system, s2 : system, c :
      connection) <=
   ** "connection c only permits well-formed data to
      flow from s1 to s2" **
   true
```

Internal connection: N/A

Define "well-formed"?

ARCHITECTURAL ASSURANCE CASE

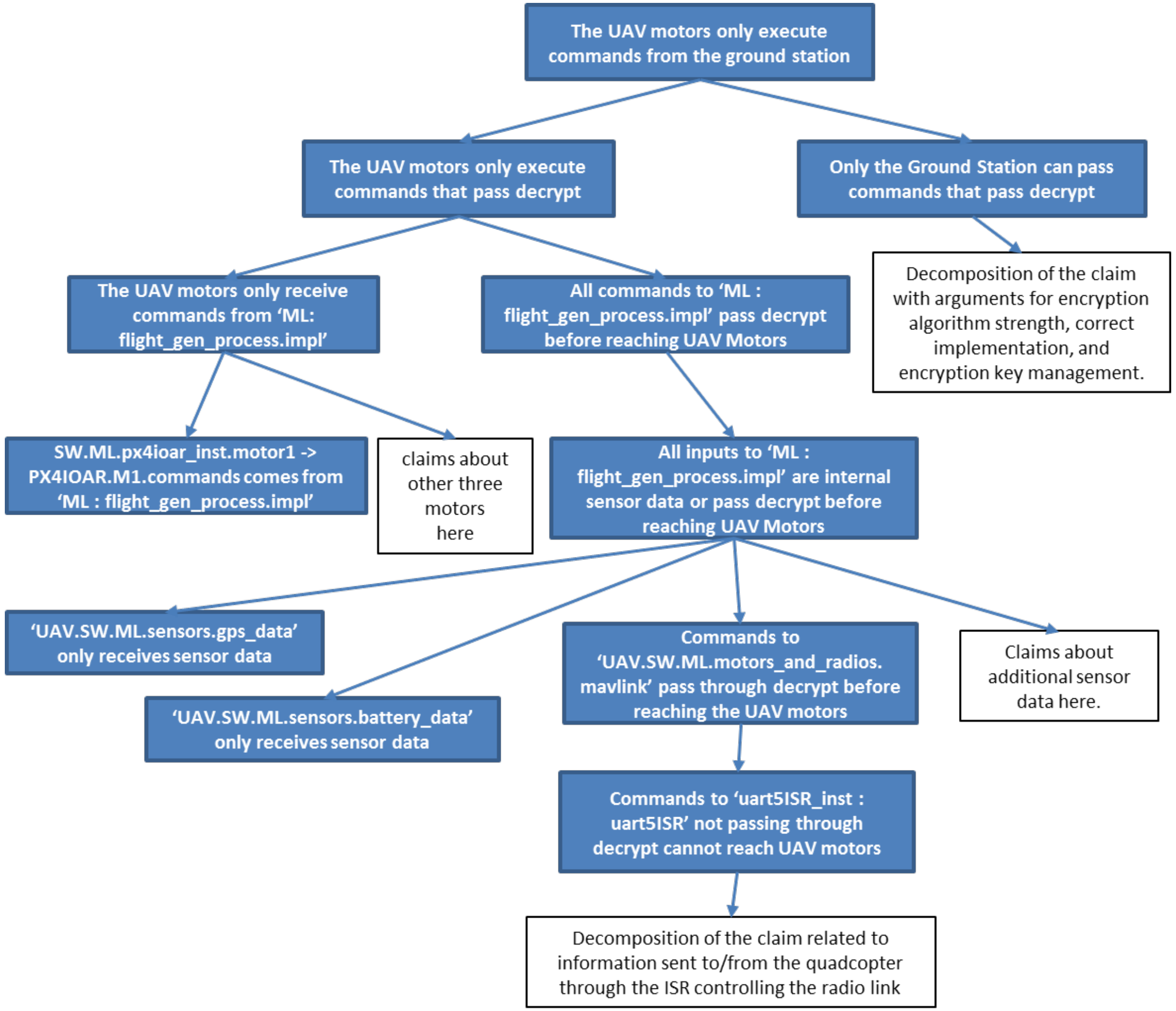
RESOLUTE

An assurance case is:

- Structured argument consisting of a tree of claims (goals), each supported by evidence, or subgoals and arguments
- Mechanism for capturing and combining all the analyses and verification performed on the architecture and its components

Resolute is a logic and tool for embedding assurance cases in AADL models

- Directly linked to architecture
- Includes different types/sources of evidence
- Reviewable by domain experts
- Adds precision to informal reasoning



Collins Aerospace

2 : CYBER RESILIENT ARCHITECTURE PATTERNS

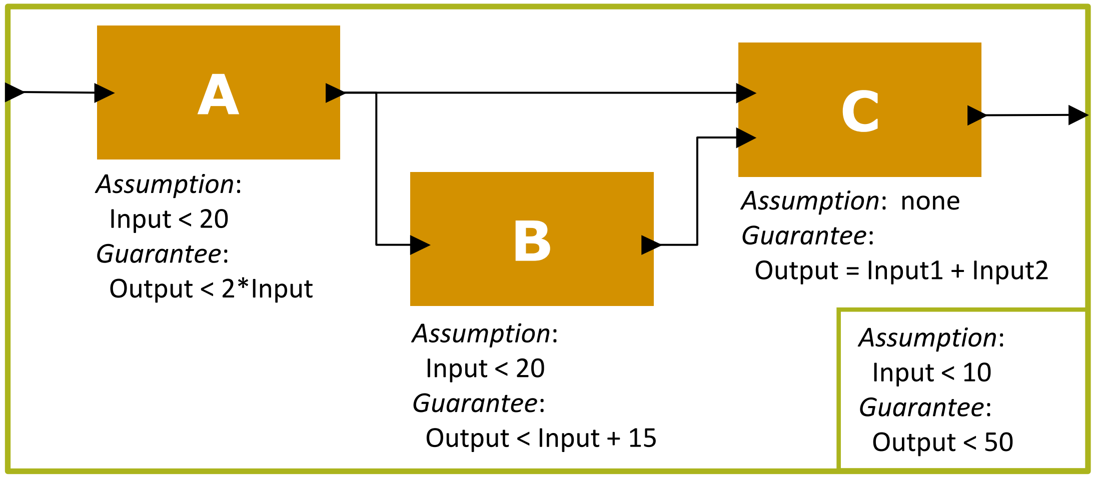
- Library of general, tool-assisted *architecture model transformations* that mitigate vulnerabilities or address cyber requirements
- Automatic insertion and verification of transform properties as assume-guarantee contracts and assurance case claims
- Examples
    - Filter ✅
    - Attestation ✅
  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

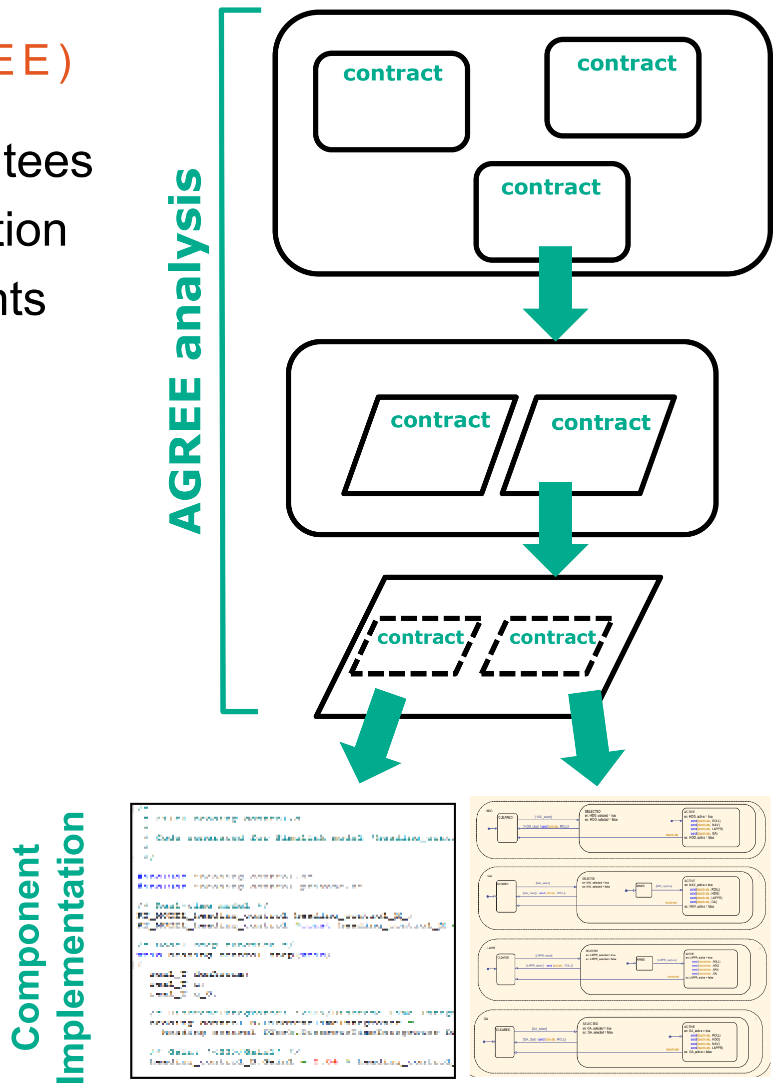
Collins Aerospace

COMPOSITIONAL REASONING

ASSUME GUARANTEE REASONING ENVIRONMENT (AGREE)

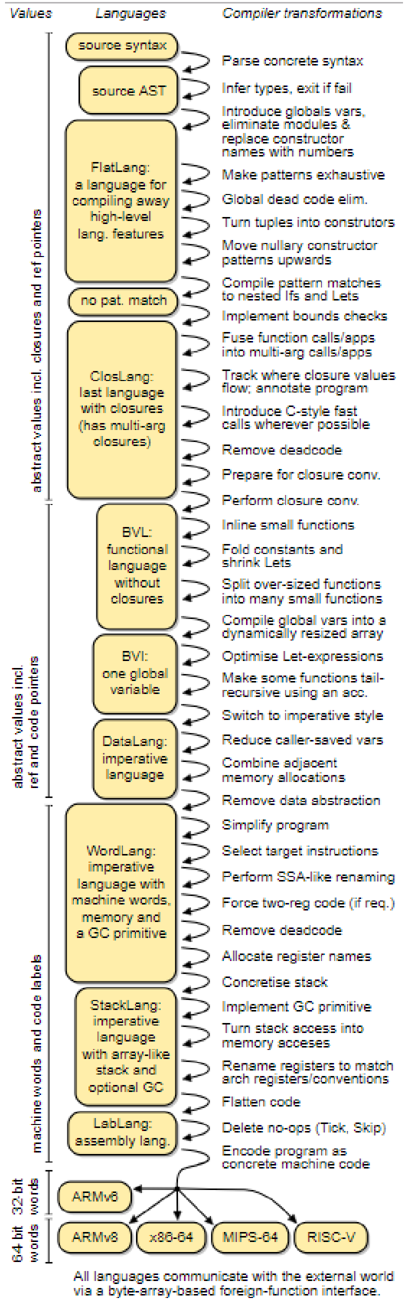
- Each subsystem has a contract consisting of assumptions and guarantees
- The contract of a component abstracts the behavior of its implementation
- Contracts at each layer must be satisfied by contracts of its components
- Leaf component contracts must be satisfied by implementation
- Compositional analysis provides **scalability**





3 : CYBER ASSURED COMPONENTS

- Many architecture transformations require the introduction of new special-purpose components
- These must be high-assurance components
  - Library of pre-verified components, or
  - Synthesized from formal specification (with proof)
- Generate high-assurance components using CakeML
  - AGREE specification -> CakeML specification
  - Provably correct synthesis
  - Additional infrastructure to create component interface
- In addition, we can build systems using the formally verified seL4 kernel and its build system (CAmkES)



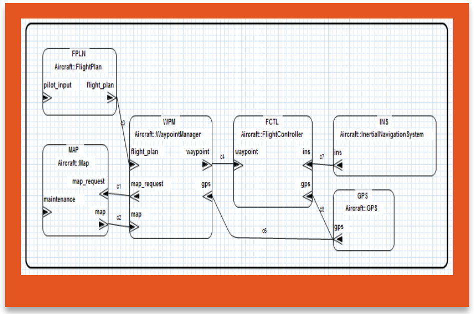
seL4 — Security. Performance. Proof.

CAKEML — A Verified Implementation of ML

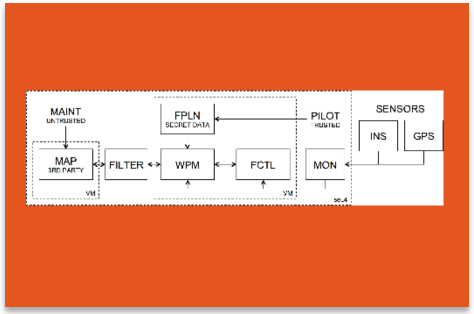
HIGH ASSURANCE IMPLEMENTATION

- Goal: End-to-end verification-based assurance
  - AADL properties -> running system properties

AADL Architecture



CAmkES architecture



seL4-based implementation

Verification of mapping AADL to CAmkES

Verification of mapping CAmkES to seL4

Assurance: separation properties hold for AADL

Assurance: separation properties hold for CAmkES architecture

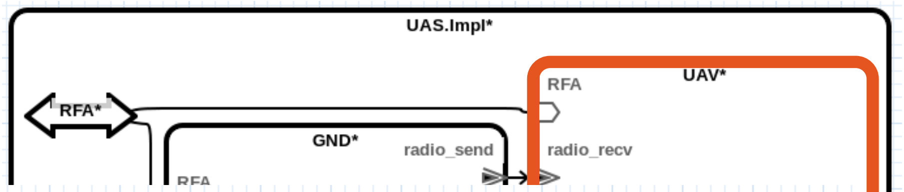
Assurance: separation properties hold for seL4 implementation

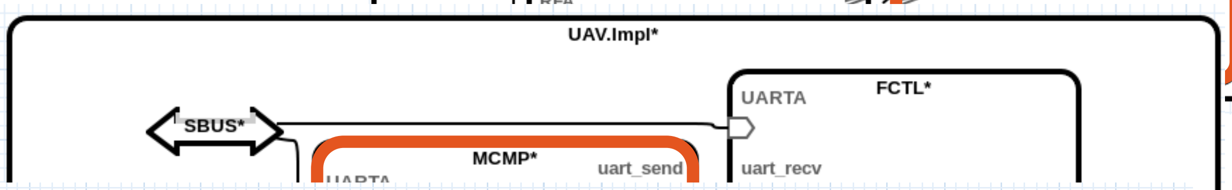
Collins Aerospace

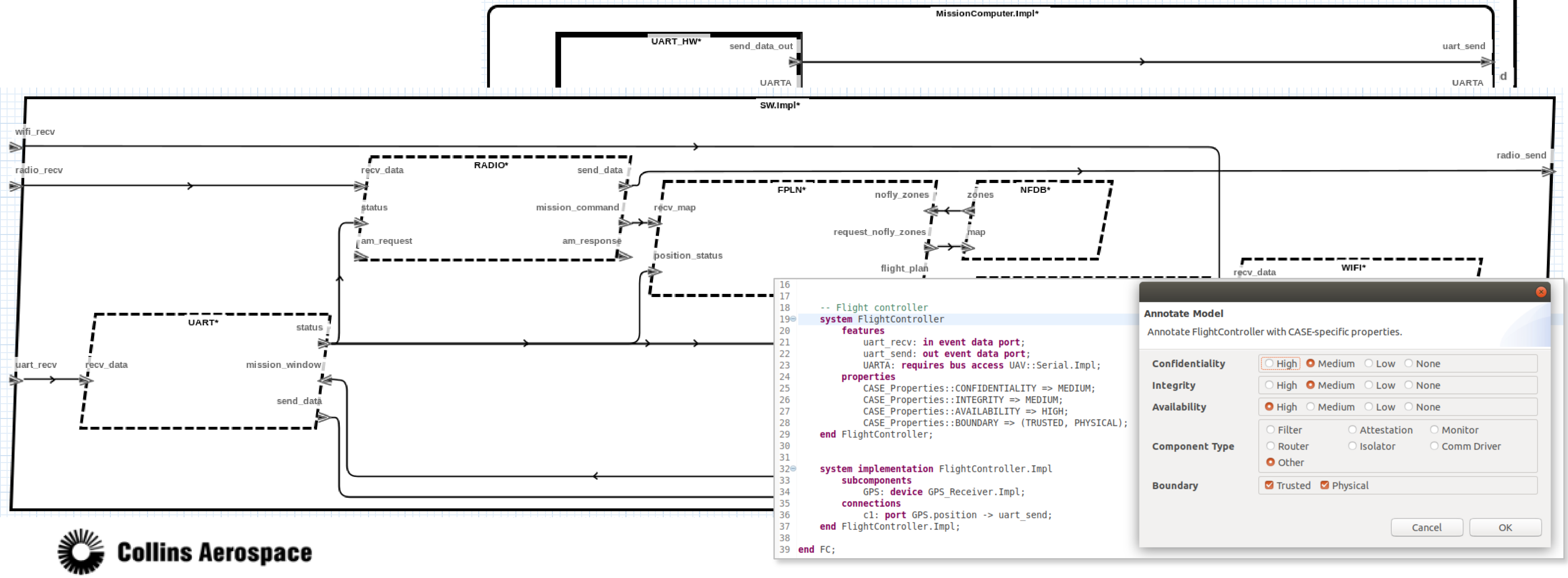
PUTTING IT ALL TOGETHER

- UAV model

https://github.com/loonwerks/CASE/tree/master/CASE_Simple_Example_V3

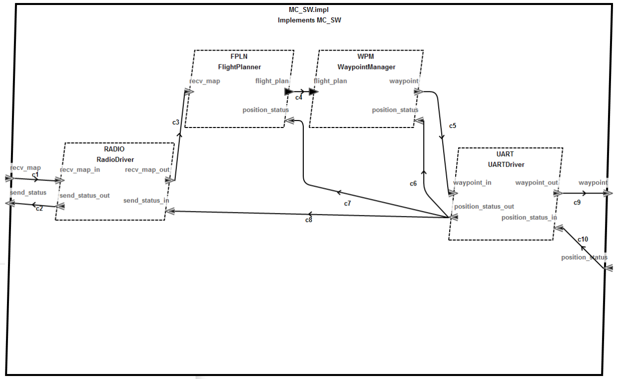






Collins Aerospace

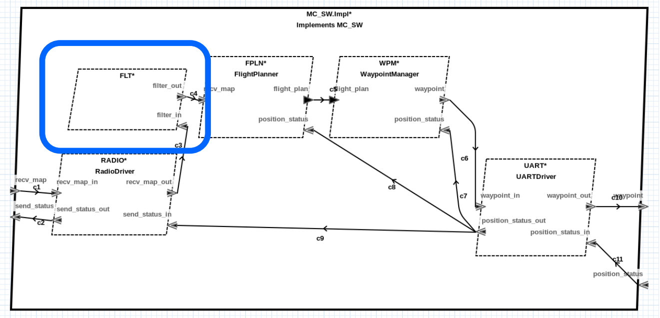
REQUIREMENTS ADDED TO AADL MODEL




```
thread FlightPlanner
    features
        flight_plan: out data port Mission.Impl;
        recv_map: in event data port Command.Impl;
        position_status: in event data port Coordinate.Impl;
    annex agree {**
        assume "The FlightPlanner shall receive a well-formed command from the GroundStation" : FALSE;
        assume "The Flight Planner shall receive an authenticated command from the Ground Station" : recv_map.HMAC = True;
        guarantee "The Flight Planner shall generate a valid mission" : SW.good_mission(flight_plan);
    **};
    annex resolute {**
        prove (well_formed())
    **};
end FlightPlanner;
```

```
package SW_CASE_Claims
  public

    annex resolute {**

        -- This connects to evidence that AGREE was previously run on the current version of the design.
        agree_prop_checked() <=
            ** "AGREE properties passed" **
            analysis("AgreeCheck")

        well_formed() <=
            ** "The FlightPlanner shall receive a well-formed command from the GroundStation" **
            agree_prop_checked()

    **};

end SW_CASE_Claims;
```


TRANSFORMATION: FILTER ADDED TO AADL MODEL




```
thread Filter
    features
        filter_in: in event data port Command.Impl;
        filter_out: out event data port Command.Impl;
    properties
        CASE::COMP_TYPE => FILTER;
        CASE::COMP_IMPL => "CakeML";
        CASE::COMP_SPEC => "(\\i{-90,90}\\i{-180,180}\\i{0,
    annex agree {**
        guarantee "The Flight Planner shall receive a well-
        guarantee "Authenticated command from the Ground Sta
    **};
end Filter;
```

```
package SW_CASE_Claims
  public

    annex resolute {**

        -- This connects to evidence that AGREE was pre
        agree_prop_checked() <=
            ** "AGREE properties passed" **
            analysis("AgreeCheck")

        well_formed(c : component, msg_type : data) <=
            ** "The FlightPlanner shall receive a well-
            agree_prop_checked() and add_filter(c, msg_

    **};

end SW_CASE_Claims;
```

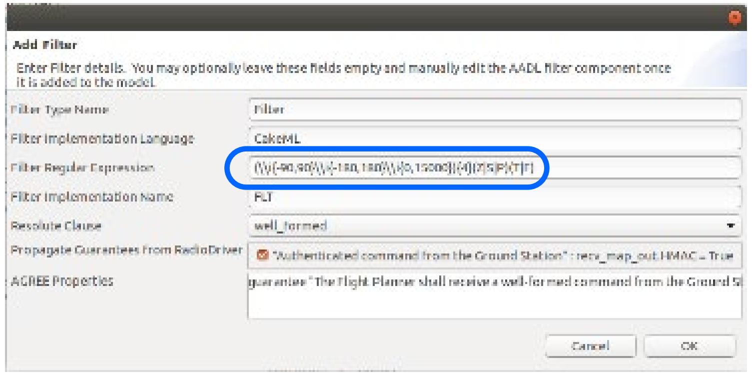
```
package Model_Transformations
public

    with CASE;

    annex Resolute {**

        ----------------------------
        -- MODEL TRANSFORMATIONS --
        ----------------------------

        -- Top-level claim for proper insertion of a filter
        add_filter(c : component, message_type : data) <=
        ** "Filter inserted before " c **
        filter_exists(c) and not_bypassed(c, message_type) and filter_prop_checked()

        -- Top-level claim for proper insertion of a router
        add_router(c : component) <=
        ** "Router inserted after " c **
        true

        -- Top-level claim for proper insertion of a monitor
        add_monitor(c : component) <=
        ** "Monitor inserted on " c **
        true

        -- Top-level claim for proper insertion of an isolator
        add_isolator(c : component) <=
        ** "Isolator added to " c **
        true

        -- Top-level claim for proper insertion of remote attestation
        add_remote_attestation(c : component) <=
        ** "Remote Attestation added at " c **
        true
```

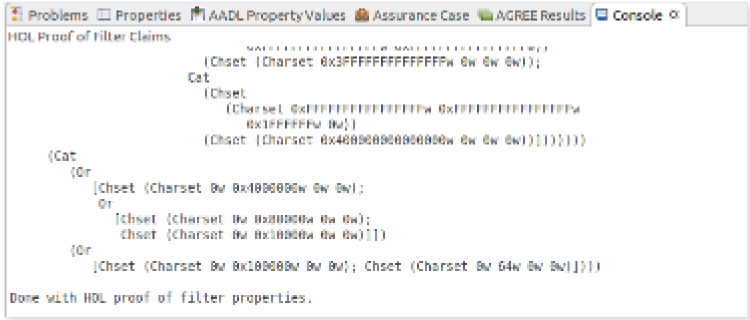
Problems | Properties | AADL Property Values | AGREE Results | Console | Progress | Assurance Case

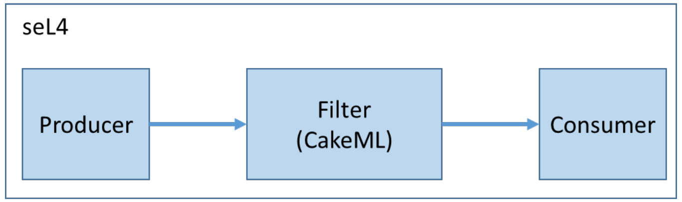
- ✔ well_formed(FPLN : SW::FlightPlanner, "good_gs_command")
  - ✔ FPLN : SW::FlightPlanner only receives well-formed messages
    - ✔ A filter exists on the communication pathway immediately before FPLN : SW::FlightPlanner
    - ✔ Filter cannot be bypassed
    - ✔ Filter property implemented by CakeML
    - ✔ AGREE property passed: [good_gs_command]

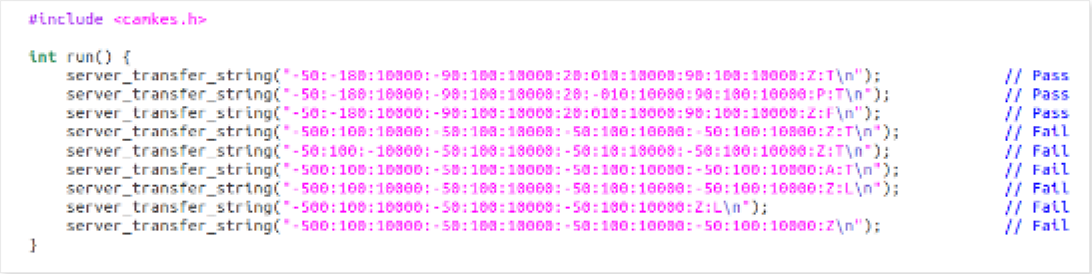
COMPONENT SYNTHESIS

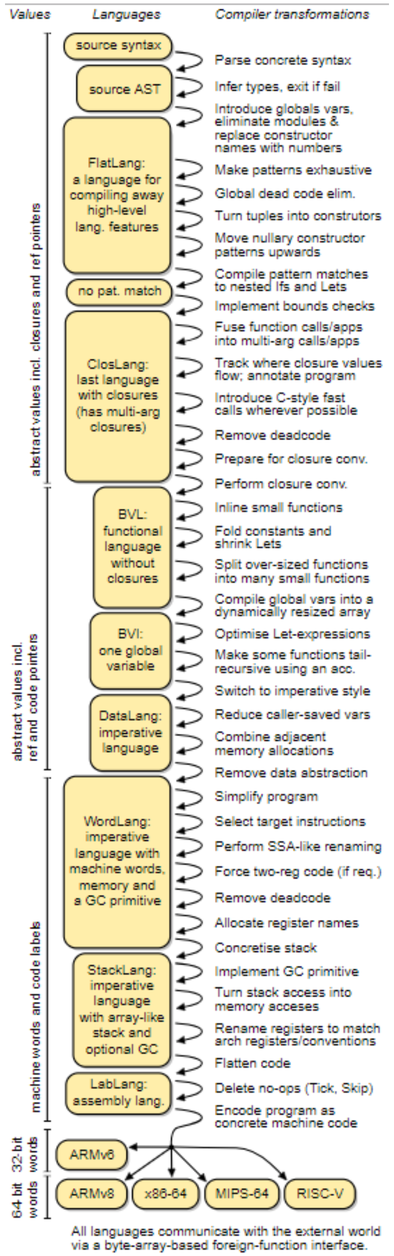
- Verified component built using CakeML
- Generated from regex filter spec
- Proof that regex implements AGREE property
- Generate CAmkES component for insertion into system











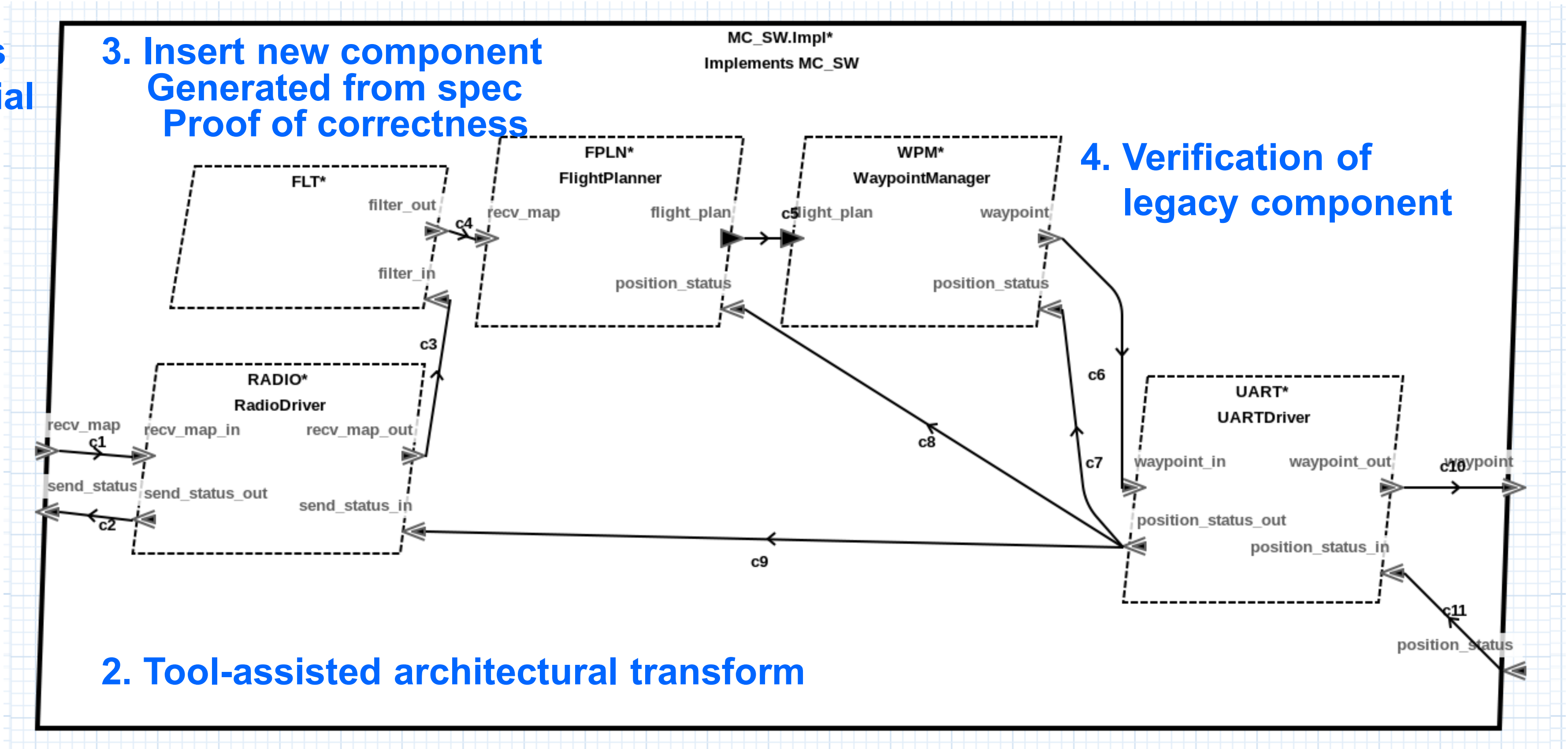
Collins Aerospace

CYBER RESILIENT TRANSFORMED SYSTEM

1. Cyber requirements generated from initial system design

3. Insert new component
   Generated from spec
   Proof of correctness

4. Verification of legacy component

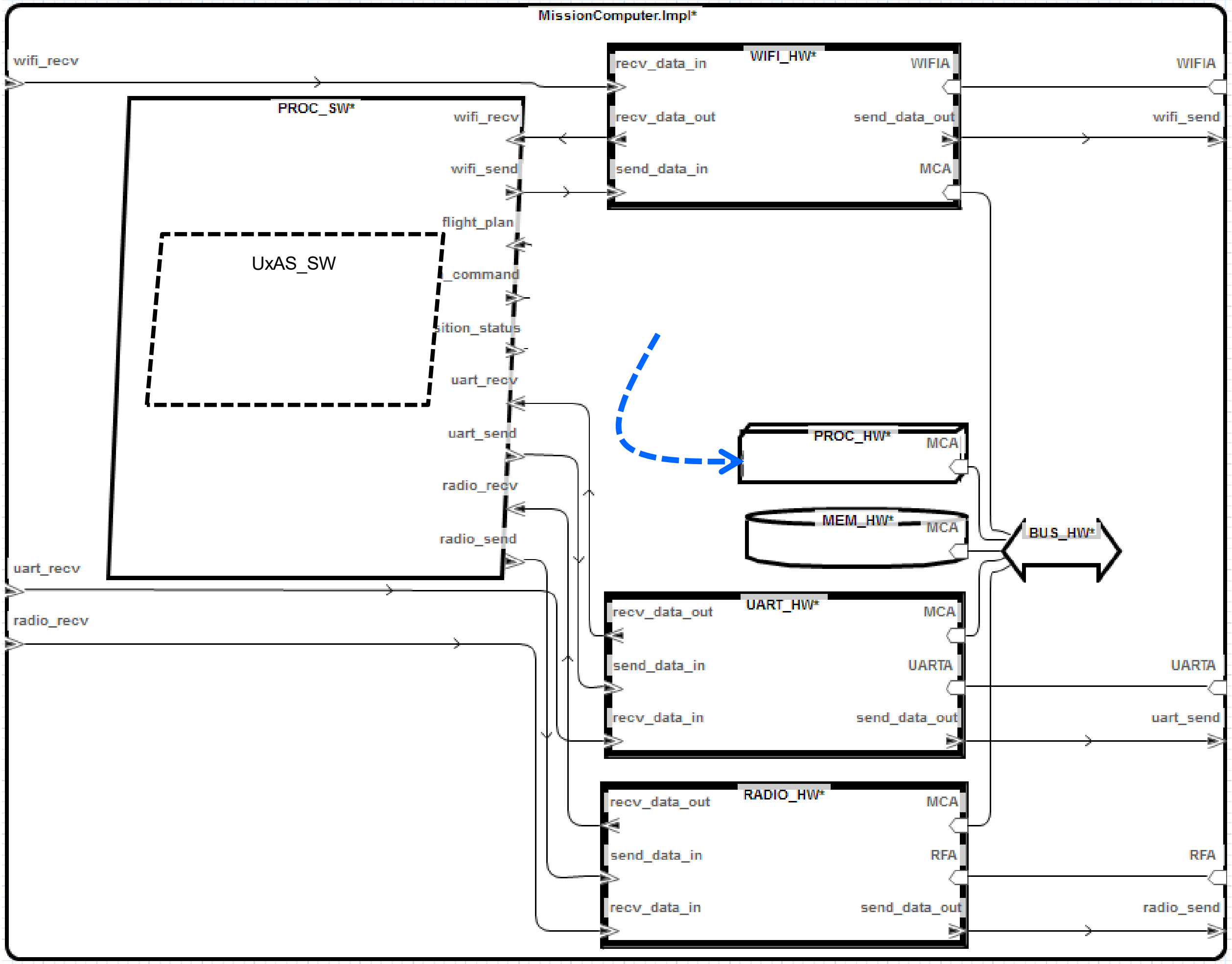


2. Tool-assisted architectural transform

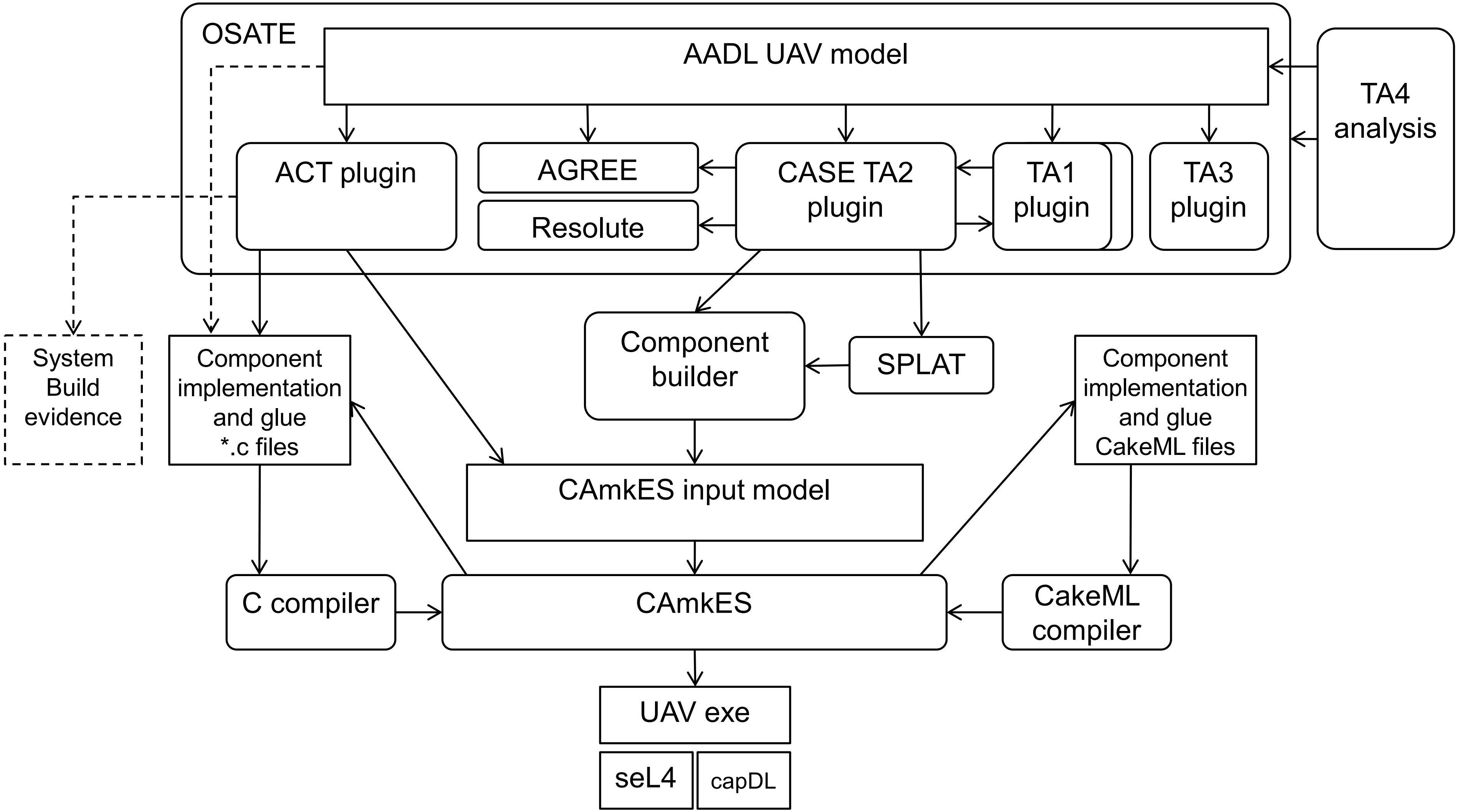
5. Transformed system satisfies cyber requirements
   Assurance case integrates evidence
   Implementation generated from model with proof of equivalence

NEXT : ISOLATION TRANSFORM

- Goal: Automate much of the manual engineering effort from HACMS
- Identify software to be isolated
  - Thread, thread group, or process
- Apply isolation transform
  - Creates virtual processor
  - Converts software to process
  - Converts connections as needed
  - Binds process to virtual processor
- Apply seL4 implementation transform



INTEGRATED TOOL ARCHITECTURE



Collins Aerospace

CASE TARGETS

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

Loonwerks

Code, papers, videos available at:

Loonwerks.com