CYBER ASSURED SYSTEMS ENGINEERING WITH AADL

AADL / ACVIP USER DAYS
4 FEBRUARY 2021

DARREN COFER
FELLOW / TRUSTED SYSTEMS
DARREN.COFER@COLLINS.COM

Collins Aerospace

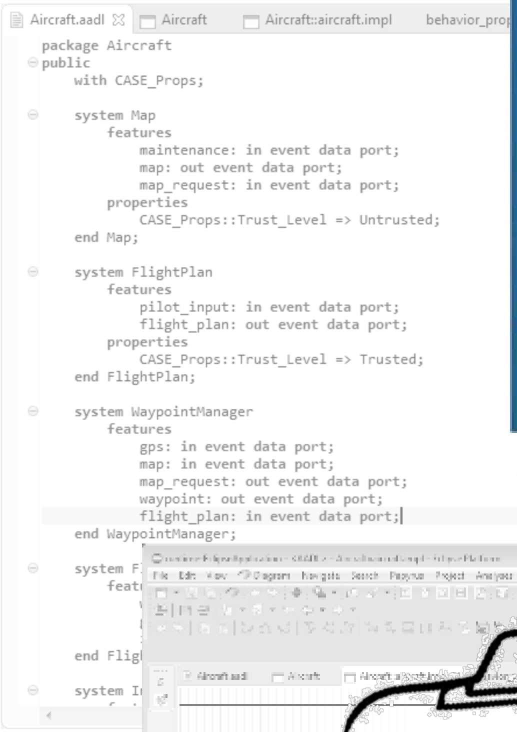
TEAM

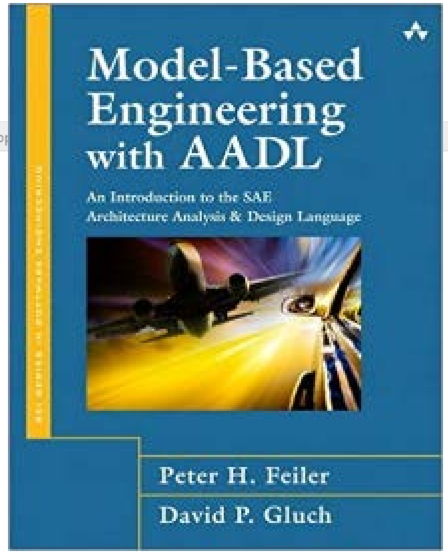
COLLINS-LED DARPA CYBER ASSURED SYSTEMS ENGINEERING (CASE) PROJECT

- Collins Aerospace
  - Architectural transformations for cyber-resilience
  - Component synthesis and proofs
  - Formal analysis and assurance case
  - Tool integration
- Data 61
  - seL4 formally verified secure microkernel for memory protection
  - Formally verified components (seL4, CakeML language)
- University of Kansas
  - Formally verified attestation for distributed computing platforms
- Adventium
  - Real-time scheduling
  - AADL modeling and code generation
- Kansas State University
  - Automatic code generation from architecture models with proof of equivalence
  - Information flow analysis

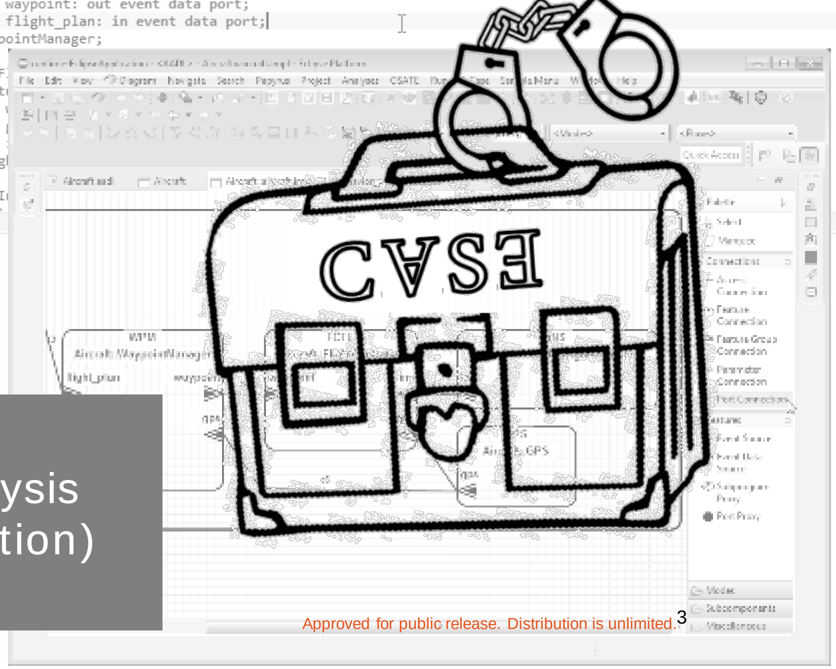
BRIEFCASE TOOL CAPABILITIES

- Integrated **model-based systems engineering** tool suite based on Architecture Analysis & Design Language (AADL) models
- Transform system design to satisfy **cyber-resiliency** requirements
- Generate new **high-assurance components** from formal specifications
- Verify system design using **formal methods** and document evidence/compliance with assurance case
- Generate **software integration code** directly from verified architecture models, targeting multiple operating systems (including seL4)





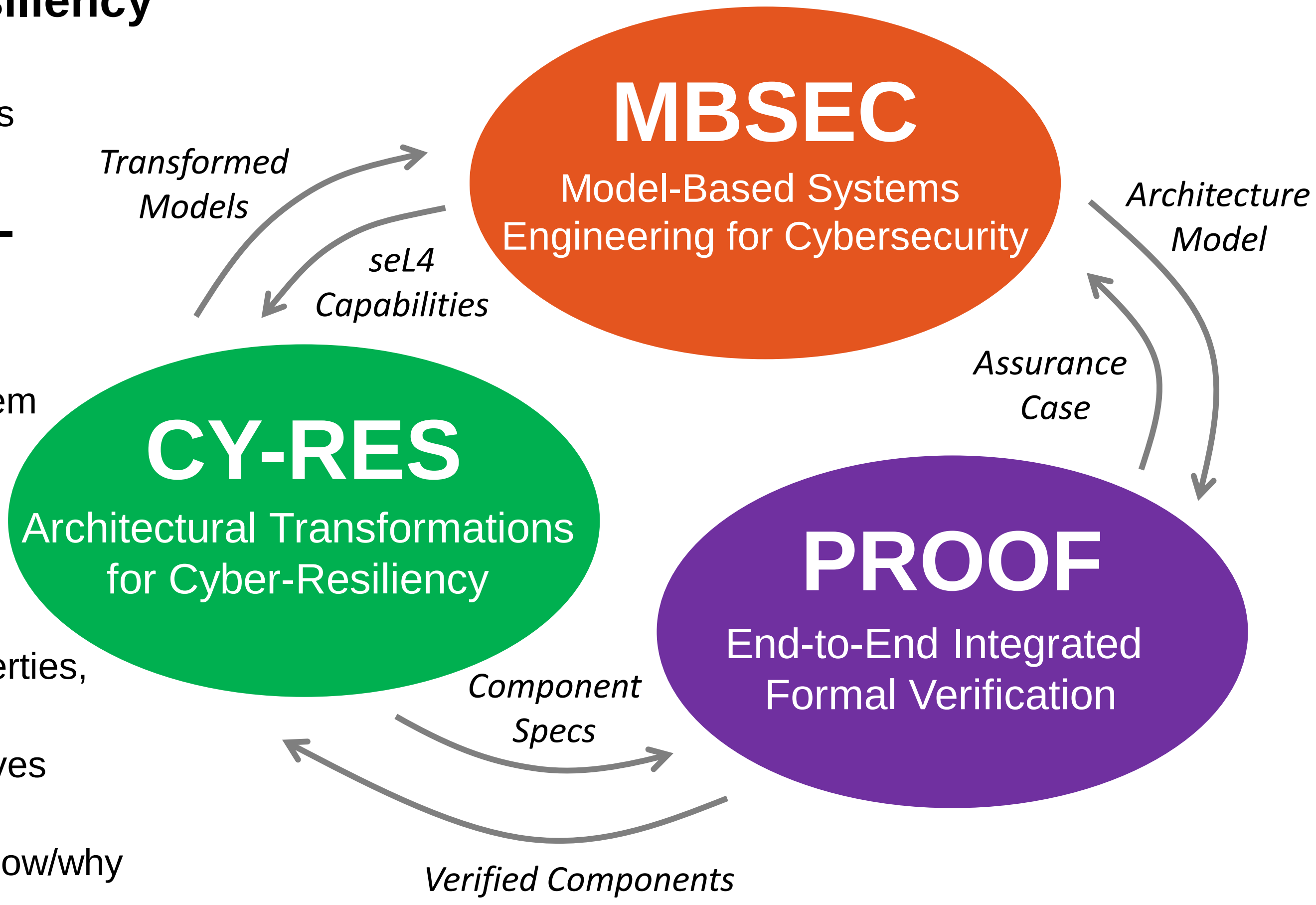
SAE AS5506 STANDARD



Why AADL?
- Sufficiently rigorous semantics to support analysis
- Correct level of abstraction (supports construction)
- OSATE supports addition of new capabilities

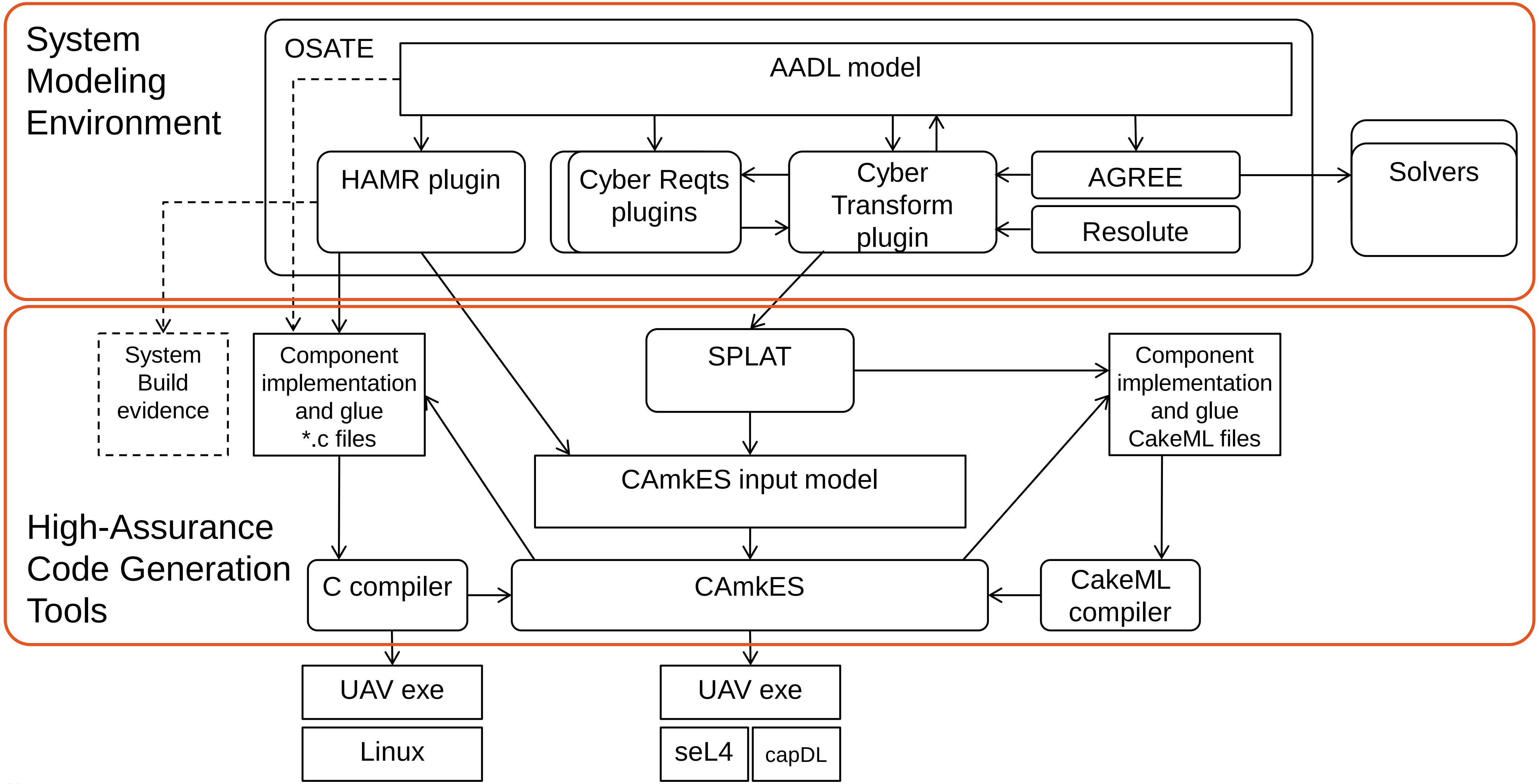
3 TECHNOLOGY PILLARS

1. **Developer assistance to implement cyber-resiliency**
   - Automated architecture transforms for threat mitigation
   - High assurance components generated from specifications
   - Techniques to deal with legacy code (cyber retrofit)
2. **MBSE environment for high-assurance cyber-resilient system development**
   - Build system directly from detailed, verified AADL model
   - Makes the security guarantees of seL4 accessible to system developers
   - Ability to target different platforms to facilitate incremental debugging/development
3. **Integration of formal verification/proof**
   - Formal verification of functional and cyber-resiliency properties, information flow properties, component proofs
   - Code generation equivalence to model, seL4 build preserves properties
   - Integrate evidence as an assurance case demonstrating how/why requirements are satisfied



Collins Aerospace

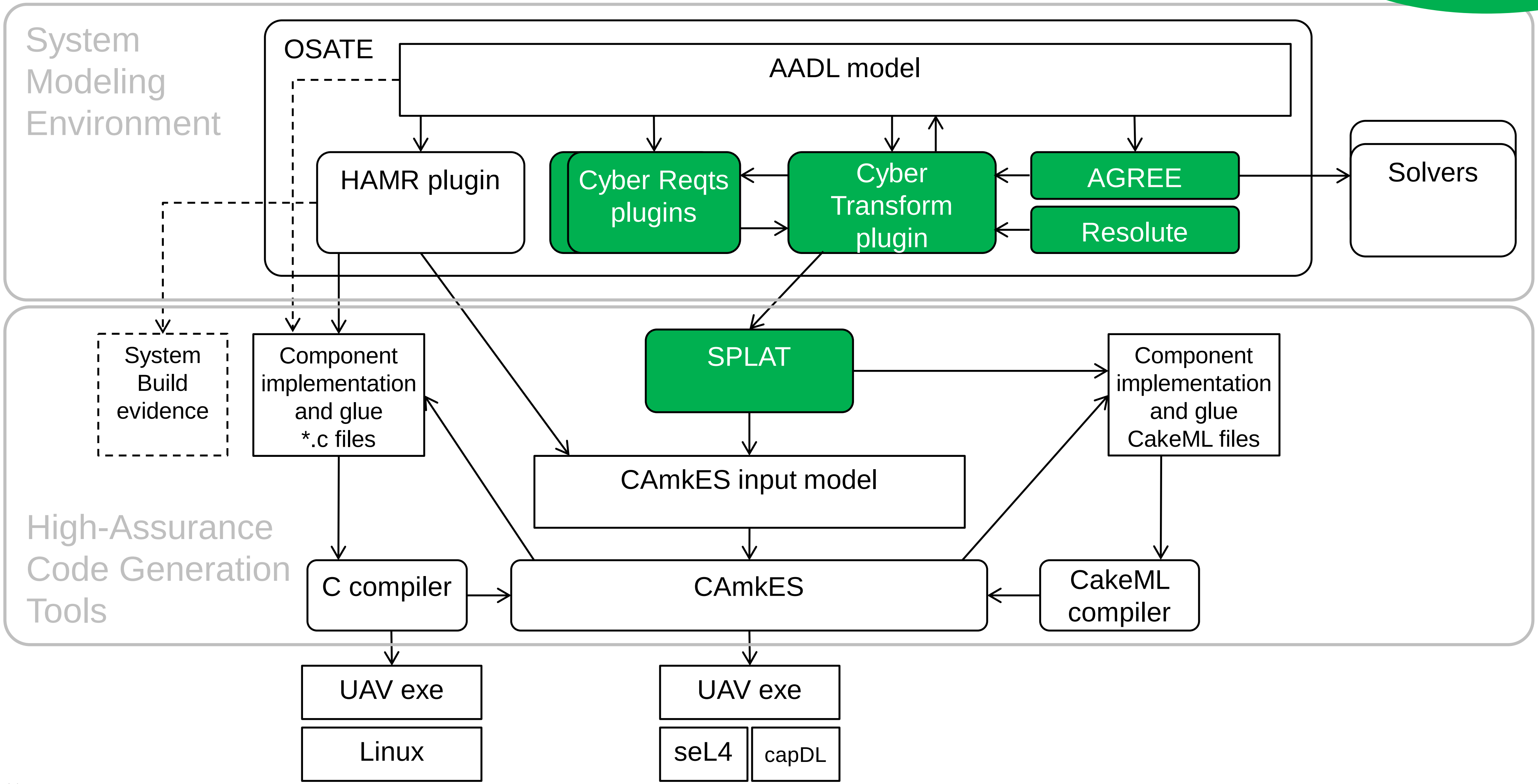
BRIEFCASE TOOL ARCHITECTURE



Collins Aerospace

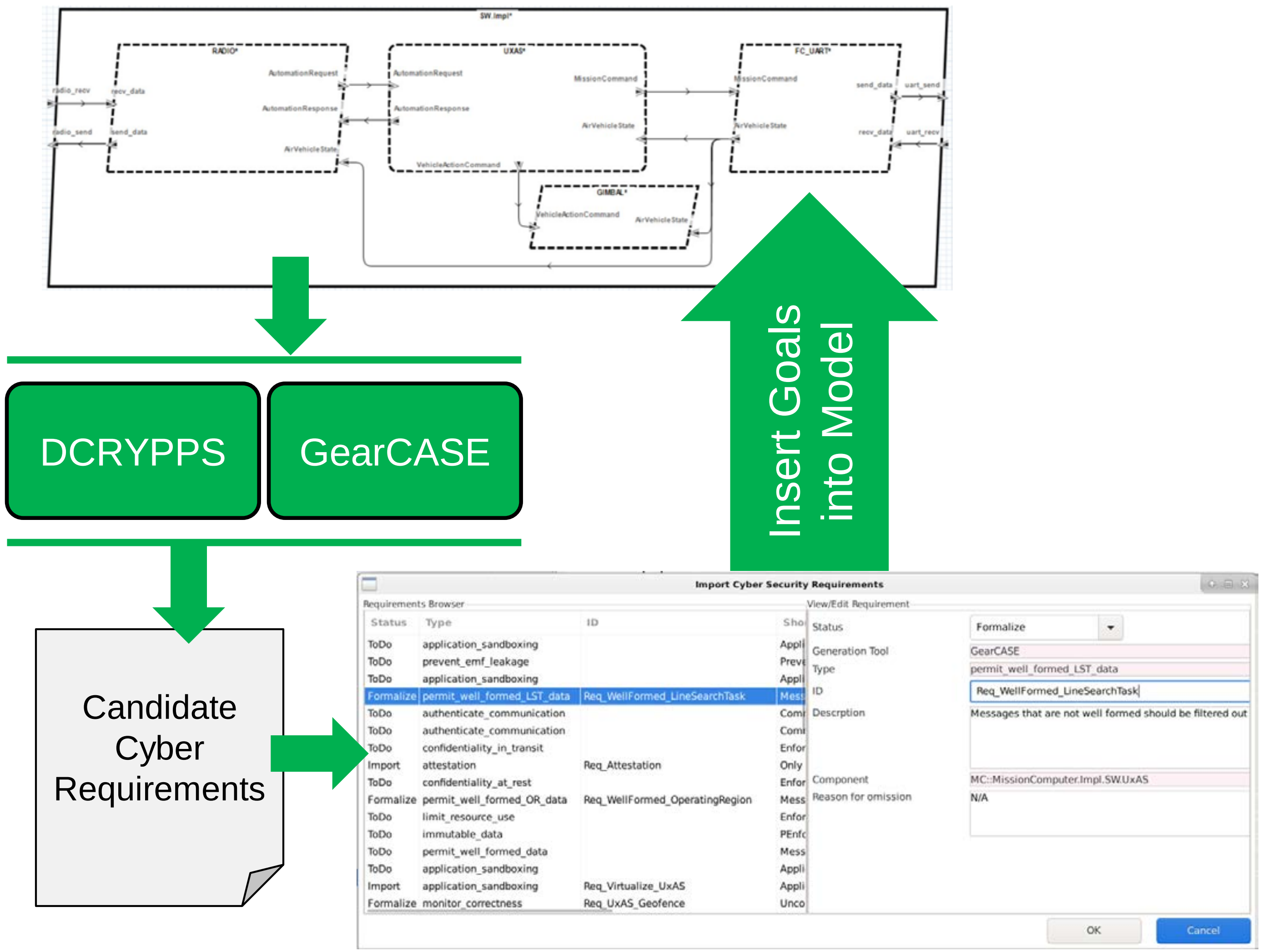
ARCHITECTURAL TRANSFORMATIONS FOR CYBER-RESILIENCY

1. CY-RES



1. GENERATE / IMPORT CYBER REQUIREMENTS

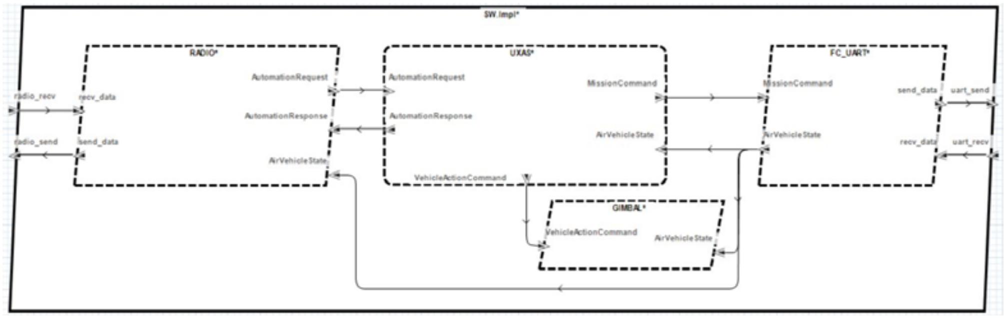
- Choose one of the Cyber Requirements generation tools
  - CRA GearCASE plugin
  - Vanderbilt/DOLL DCRYPPS plugin
- Initial model data is exported to selected tool
- Requirements import wizard manages the generated requirements
  - Select action
  - Naming/tagging
  - Associate with formal properties
- **Requirements inserted into model as Resolute goals (GSN)**
  - We will build an assurance argument to satisfy these goals

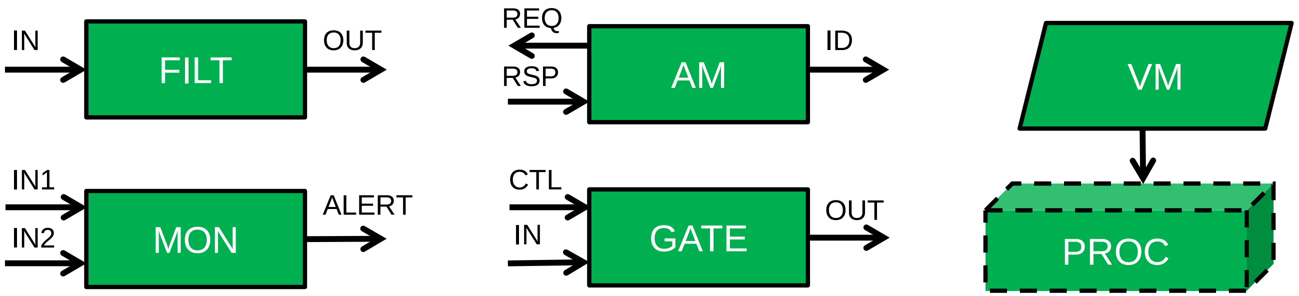


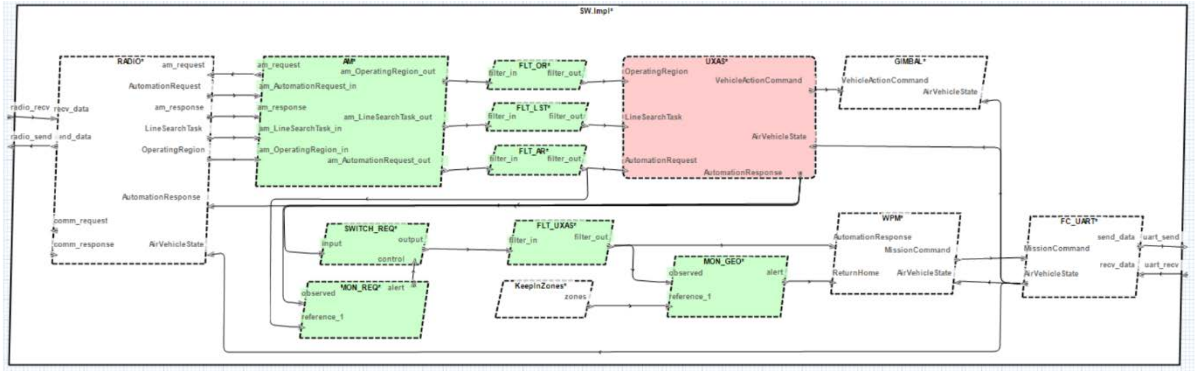
2. APPLY CYBER TRANSFORMATIONS

- Cyber requirements tools provide model context and sometimes suggested mitigation
- System engineer selects from available cyber-resiliency transformations
  - Filter
  - Monitor
  - Gate (controlled by monitor)
  - Attestation
  - Virtualization
  - seL4 build prep
- Wizard interface collects needed configuration data
- **Tool automatically transforms AADL model**

- Also adds Resolute assurance case strategy to show how the associated goal (requirement) is satisfied

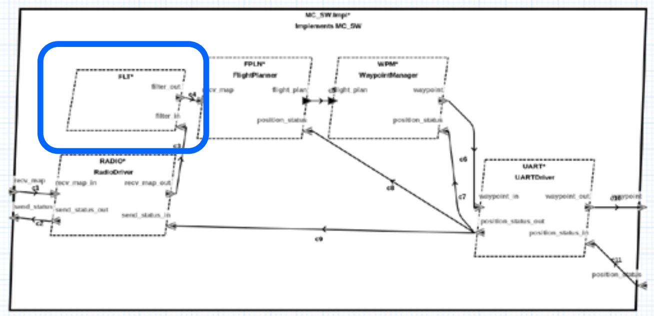






Collins Aerospace

2A. INSERT ASSURANCE CASE STRATEGY



- **Resolute links cyber transform to goal as a GSN strategy**
- Checks for violations/changes that impact correctness
- Collects evidence and generates assurance case

```
thread Filter
    features
        filter_in: in event data port
        filter_out: out event data po
    properties
        CASE::COMP_TYPE => FILTER;
        CASE::COMP_IMPL => "CakeML";
        CASE::COMP_SPEC => "(\\i{-90,
    annex agree {**
        guarantee "The Flight Planner
```

```
package CASE_Requirements
private

    annex Resolute {**

        goal Req_WellFormed(comp_
            ** "[permit_well_form
            context Generated_By
            context Generated_On
            context Req_Component
            context Formalized :
            agree_property_checke
```

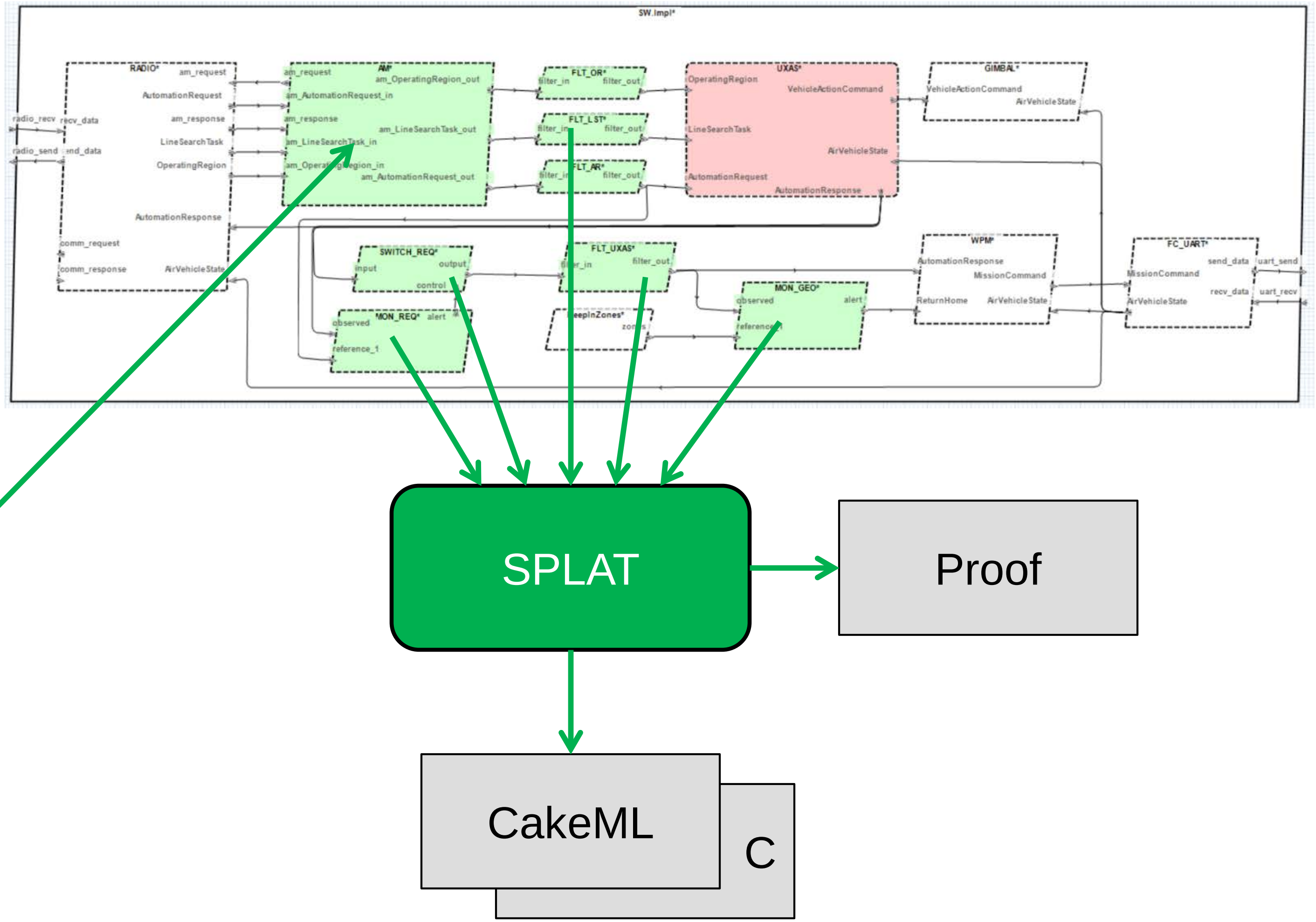
```
annex Resolute {**

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

    -- Top-level claim for proper insertion of a filter
    goal add_filter(comp_context : component, filter : component, conn : connection, msg_type : data) <=
        ** "Filter " filter " is properly added to component " comp_context **
        filter_exists(filter, comp_context, conn) and component_not_bypassed(filter, comp_context, msg_type) and component_implemented(filter)

    -- Top-level claim for proper insertion of attestation manager
    goal add_attestation_manager(comm_driver : component, attestation_manager : component, attestation_gate : component) <=
        ** "Attestation Manager added for communications driver " comm_driver **
        attestation_manager_exists(comm_driver, attestation_manager) and attestation_manager_not_bypassed(comm_driver, attestation_manager, attes
```

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]

3. GENERATE HIGH ASSURANCE COMPONENTS

- Some of the cyber transforms insert new high-assurance components into the model
- The behavior of the component (its contract) is specified in AGREE
- **SPLAT generates component implementations from their specifications**
- SPLAT also generates a proof showing that the component implements its specification

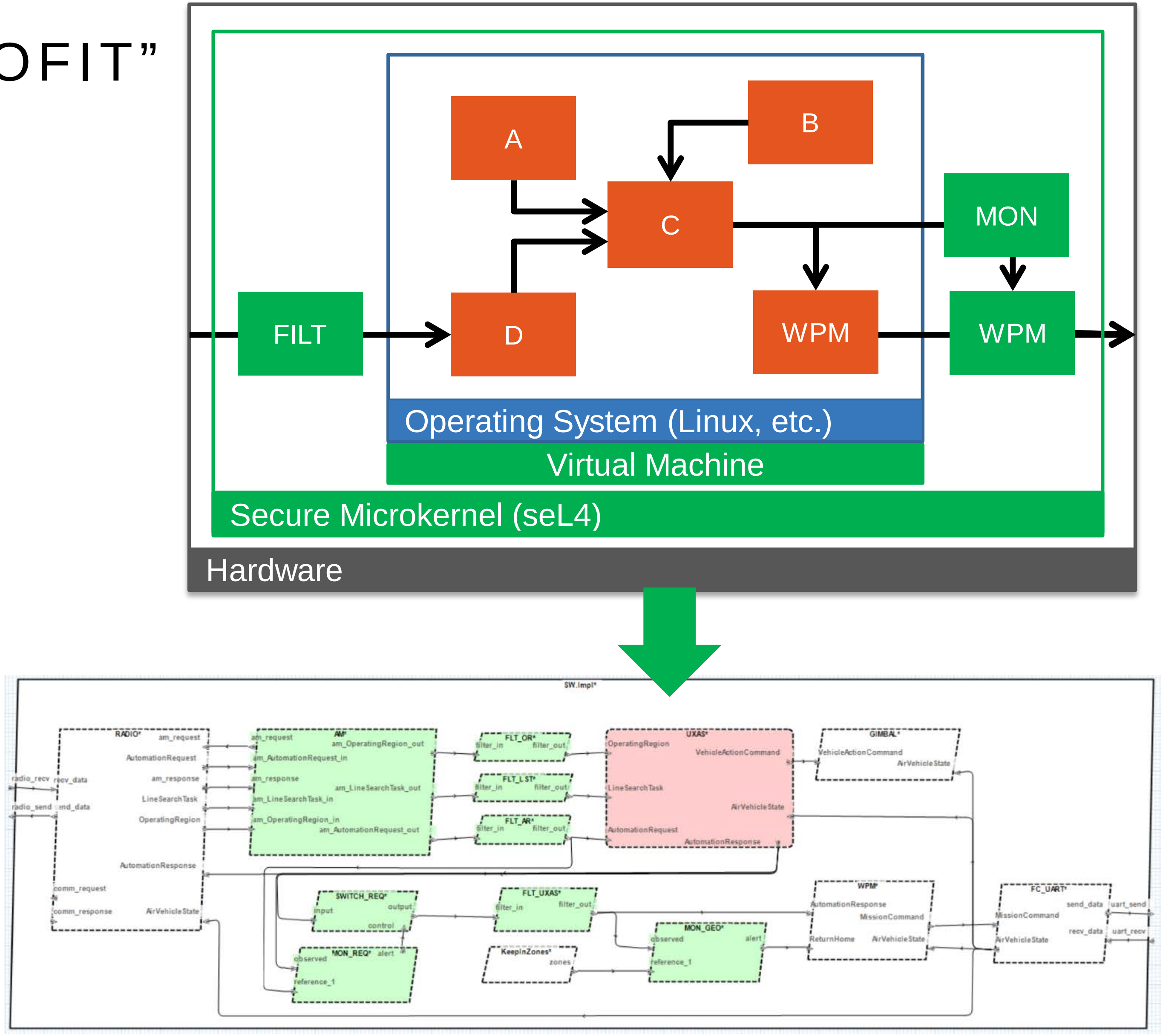


- Other components (such as the Attestation Manager) are pre-built pre-verified libraries
- Their implementations are essentially library functions that are added to the build, possibly with some configuration data from the model

USE CASE : “CYBER RETROFIT”

- How can we deal with legacy code/systems?
- Example: UxAS
  - Open source, less trusted
  - Not designed for cybersecurity
  - Written in C++

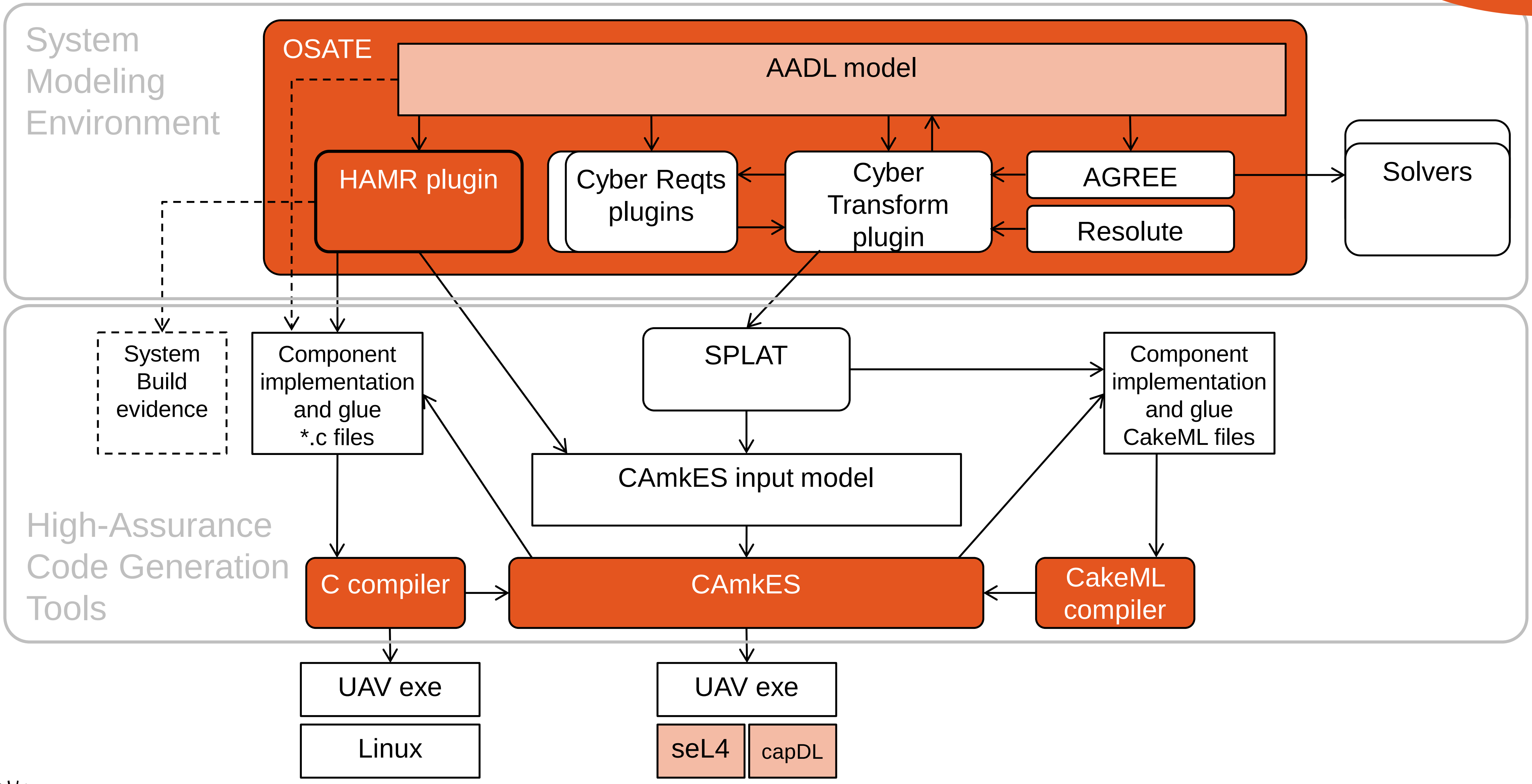
1. Virtualize legacy system
   - Support for multiple VMs, if needed
2. Host on seL4
3. Extract/harden critical components
4. Filter inputs
5. Monitor outputs



Collins Aerospace

MODEL-BASED SYSTEMS ENGINEERING FOR CYBERSECURITY

2. MBSEC



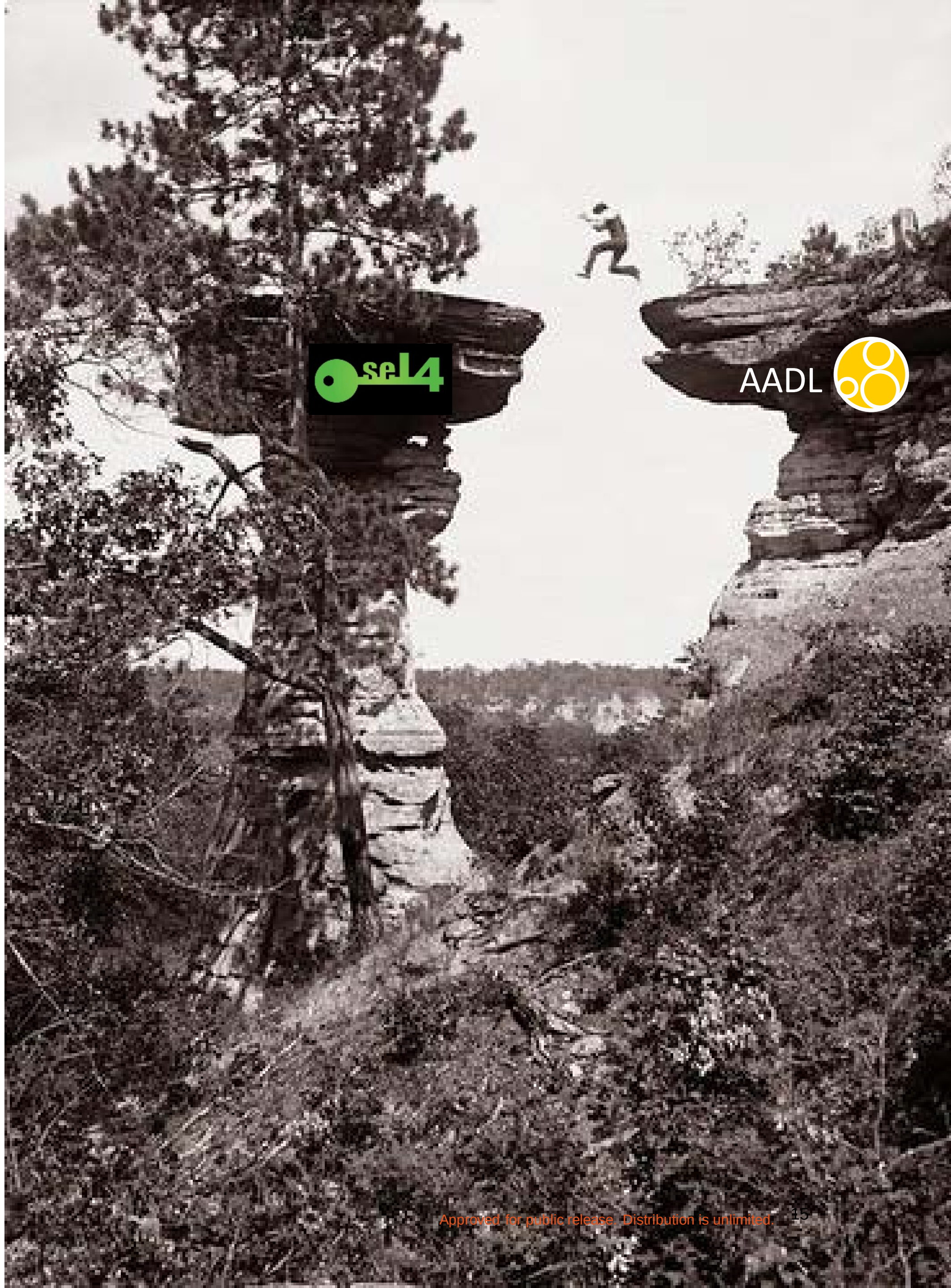
Collins Aerospace

MBSEC

MODEL-BASED SYSTEMS ENGINEERING FOR CYBERSECURITY

- **The model IS the system**
  - It is not just a picture for communication, or a way to capture requirements, or a means to perform analyses
  - It is all of those things, but it is also how we will actually build the system, and do so in a way that ensures compliance with the design captured in the model
- Rationale for AADL
  - Provides the necessary precision and semantics for building real-time distributed embedded software systems
- MBSE for seL4
  - Furthermore, we are bridging the gap between systems engineers and the formally verified **seL4 microkernel**
  - Ensure usability by systems engineers and promote successful transition
- HAMR
  - High Assurance Modeling for Rapid Engineering
  - Code generation from AADL models

Collins Aerospace

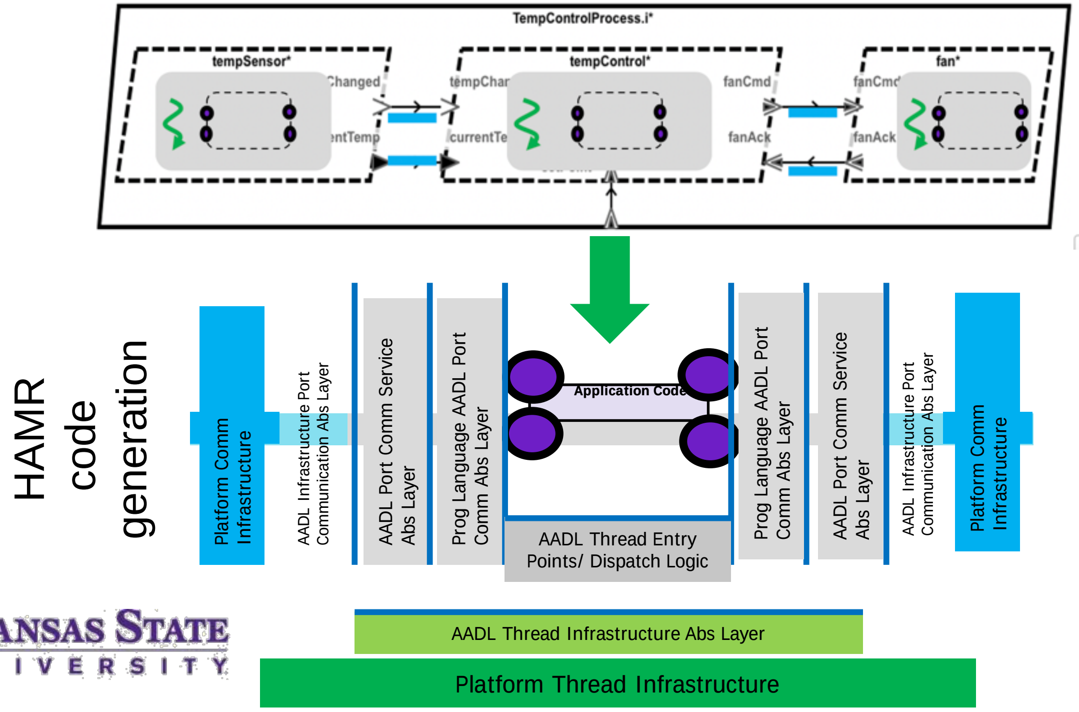


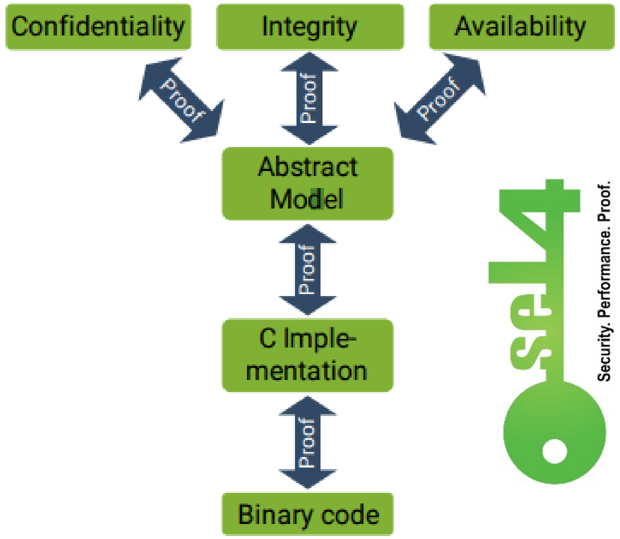
SOFTWARE INFRASTRUCTURE

HAMR AND SEL4

- HAMR is a multi-stage translation architecture to address CASE goals of component migration between platforms and information flow control
- Semantic consistency from model to execution
- Ensures model-level analysis applies to deployed code
- Same computational model across different platforms
- Build for multiple target platforms:
    - seL4 / Linux / Virtual Machine
    - Build for workstation / emulator / embedded platform

- seL4 microkernel guarantees partitioning of components and communication, backed by computer-checked proofs
- seL4 guarantees no infiltration, exfiltration, eavesdropping, interference, and provides fault containment for untrusted code



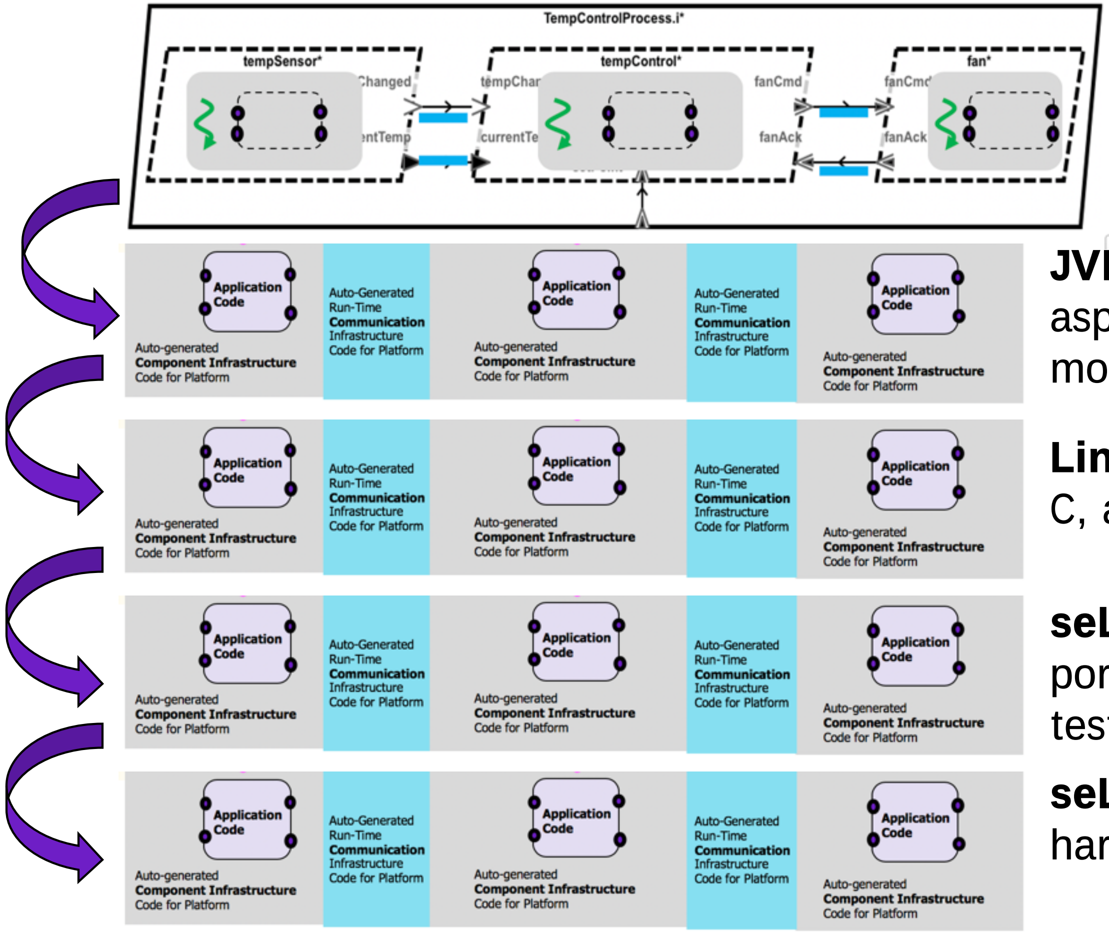


seL4 is…

- An operating system microkernel
- A hypervisor
- Proved correct
- Provably secure
- Fast

HAMR SUPPORTS MULTIPLE LANGUAGE/ PLATFORM COMBINATIONS

The flexibility of being able to easily shift between different platforms was quite useful as the team experimented with building the Phase 2 Experimental Platform assessment deliverable.



AADL / OSATE – design model, types, perform analyses

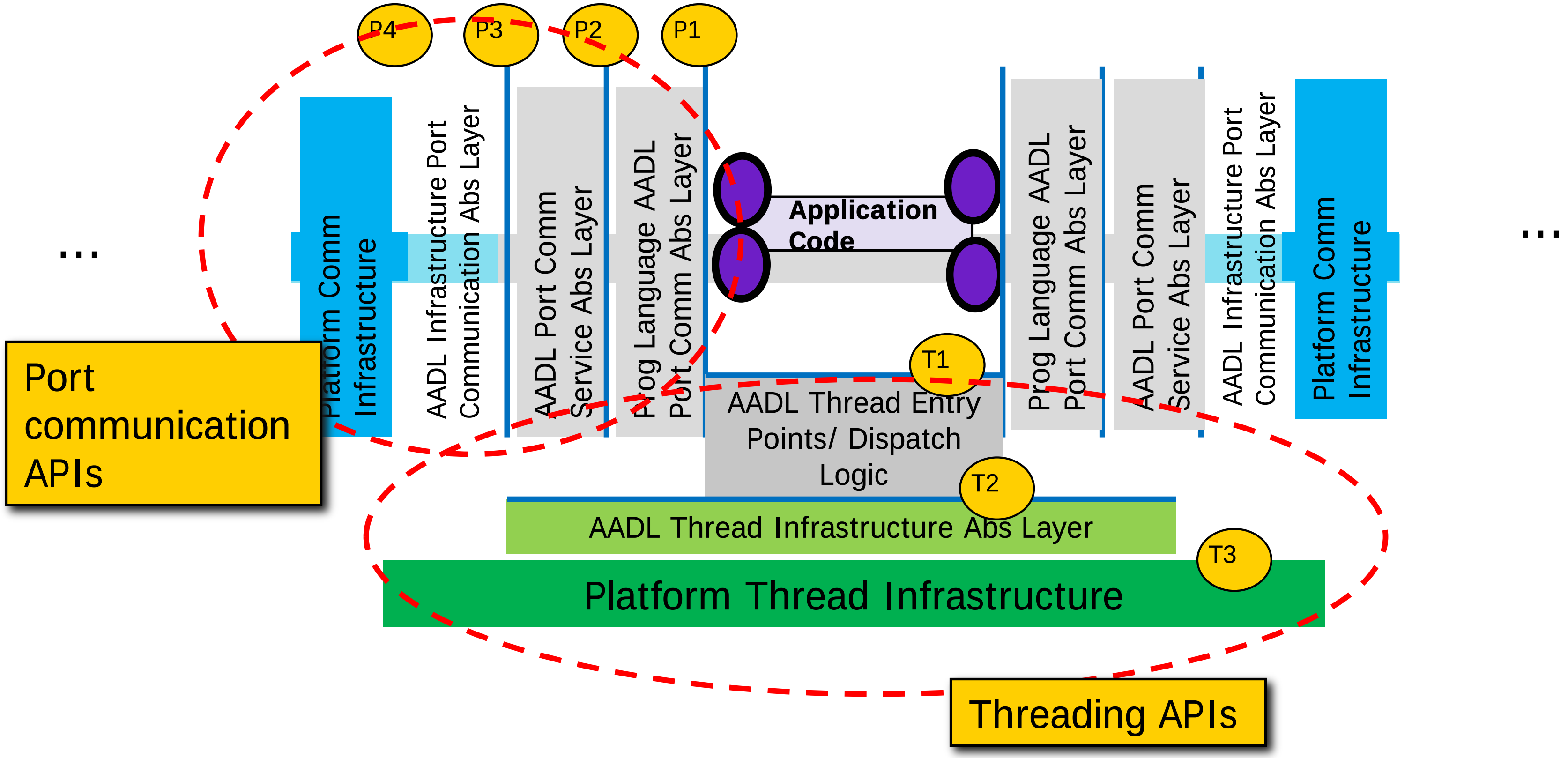
JVM / Slang – data types, port constraints, basic aspects of application logic, initial unit testing – some mocked up components, many useful visualizations

Linux C – compile Slang to C, or manually code C, and debug C implementation, VMs mocked up

seL4 C / Qemu – C application code easily ports to seL4 native components, add in VMs, test/simulate/debug in Qemu

seL4 C / board – seL4 build shifted to actual hardware for final testing and deployment

HAMR ABSTRACTION LAYERS



APPLICATION LOGIC

Standardized entry point for HAMR / AADL periodic threads (skeleton auto-generated)

```
Unit hamr_SW_WaypointPlanManagerService_thr_Impl_Impl_timeTriggered_(
  STACK_FRAME
  hamr_SW_WaypointPlanManagerService_thr_Impl_Impl this) {
    bool dataReceived = false;
    size_t numBits = 0;
    uint8_t payload[MAX_PAYLOAD];

    dataReceived =
    api_get_ReturnHome__hamr_SW_WaypointPlanManagerService_thr_Impl_Impl(this);
    if (dataReceived) {
      returnHome = true;
    }

    if (returnHome || automationResponse != NULL) {
      dataReceived =
      api_get_AirVehicleState__hamr_SW_WaypointPlanManagerService_thr_Impl_Impl(
        this, &numBits, payload);
      if (dataReceived) {
        air_vehicle_state_in_event_data_receive_handler(this, payload);
      }
    }
}
```

Use of auto-generated API for input event port

Use of auto-generated API for input event data port

Note: *Same C APIs are used for seL4 native, seL4 VM (as an option), and Linux. These mirror the structure of Slang and CakeML APIs.*

RESOLINT: LINTER TOOL FOR AADL MODELS

- Define *rules* in Resolute that correspond to modeling guidelines

- Group rules into *rulesets* corresponding to organizational process, customer requirements, certification guidelines, and tool constraints

- Automatically check compliance with modeling guidelines in OSATE

Modeling Guidelines

rule **dispatch_protocol_specified**	Threads should have the dispatch_protocol property specified	⚠
rule **valid_dispatch_protocol**	Threads can only specify a dispatch_protocol property of *periodic* or *sporadic*	⊗

```
dispatch_protocol_specified() <=
    ** "Threads should have the Dispatch_Protocol property specified" **
    forall (t : thread) .
        lint_check(t, has_property(t, Dispatch_Protocol))

valid_dispatch_protocol() <=
    ** "Threads can only specify a dispatch_protocol property of periodic or sporadic" **
    forall (t : thread) . lint_check(t, has_property(t, Dispatch_Protocol) =>
        (property(t, Dispatch_Protocol) = "Sporadic" or property(t, Dispatch_Protocol) = "Periodic"))
```

```
package System_Build

public

    annex resolute {**

        ruleset HAMR {
            info (print("Linting HAMR ruleset"))

            error (one_process())

            warning (dispatch_protocol_specified())
            error (valid_dispatch_protocol())

            error (bounded_integers())
            error (bounded_floats())

            warning (data_type_specified())
            warning (subcomponent_type_specified())

            error (array_dimension())
            error (one_dimensional_arrays())
```

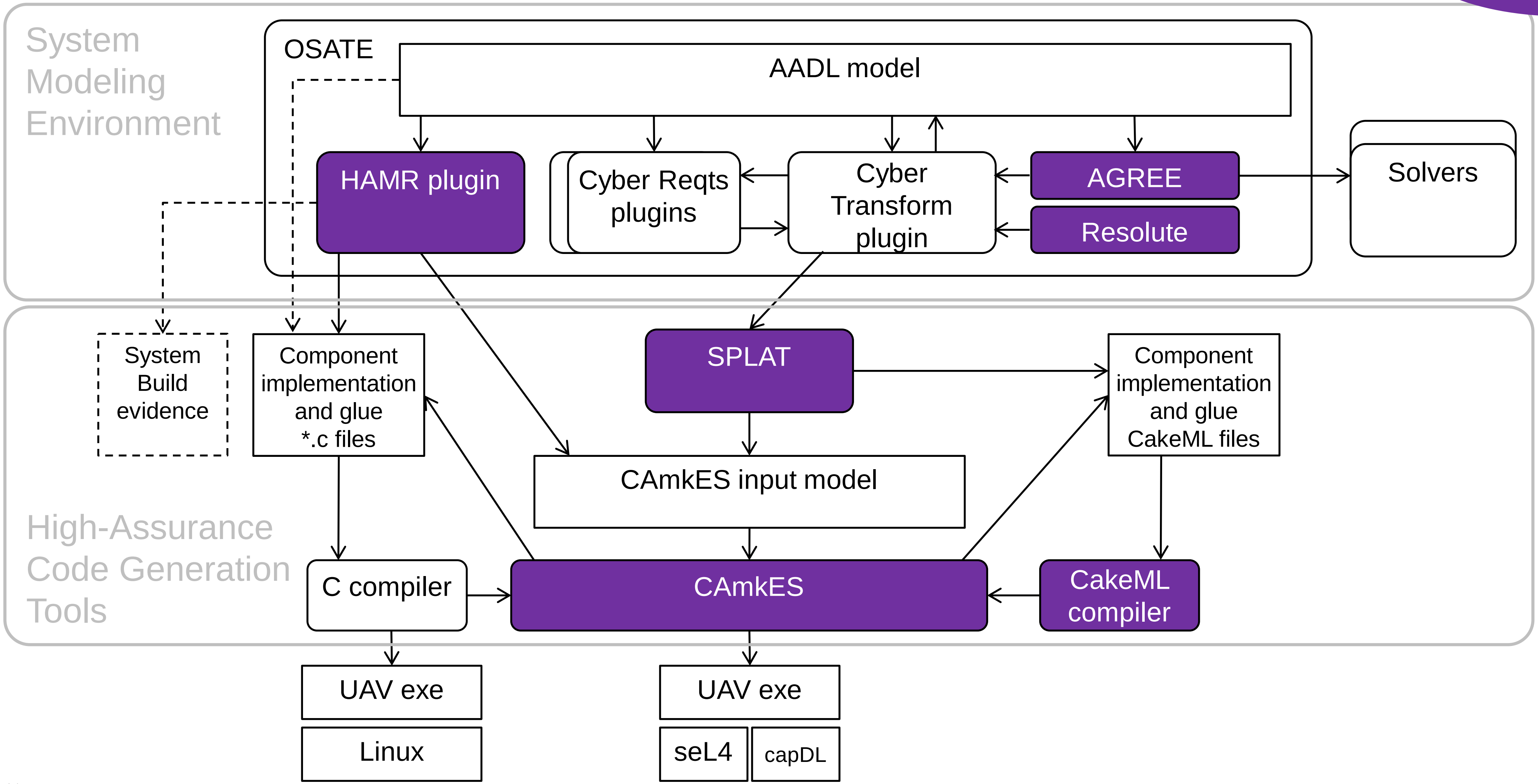
Problems | Properties | Console

2 errors, 54 warnings, 1 other

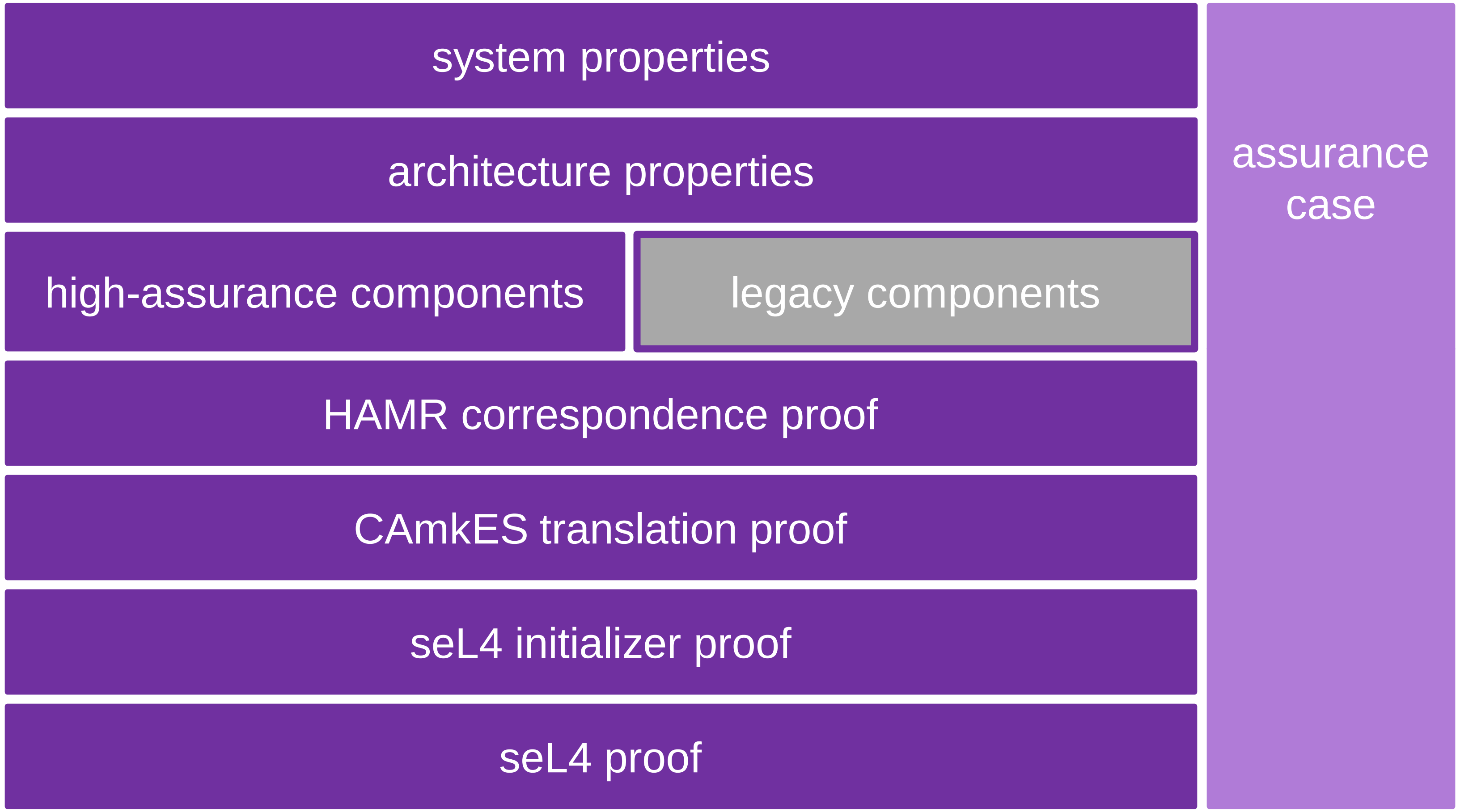
Description
Errors (2 items)
Components bound to a virtual processor may only communicate with components bound to other processors via event data ports
Only one processor-bound process can contain thread or thread-group subcomponents
Warnings (54 items)

END-TO-END INTEGRATED FORMAL VERIFICATION

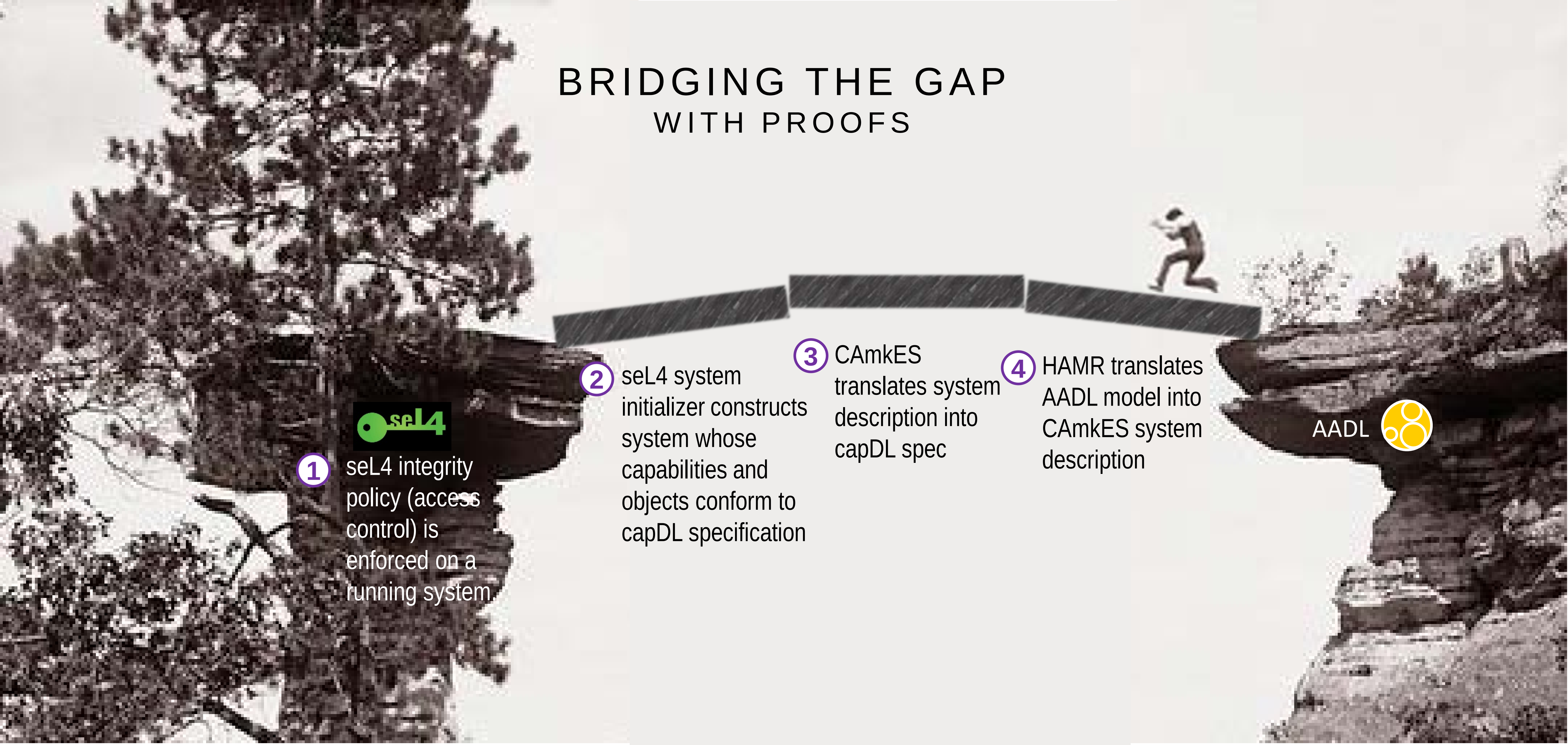
3. PROOF



END-TO-END INTEGRATED FORMAL VERIFICATION



BRIDGING THE GAP
WITH PROOFS



1 seL4 integrity policy (access control) is enforced on a running system.

2 seL4 system initializer constructs system whose capabilities and objects conform to capDL specification

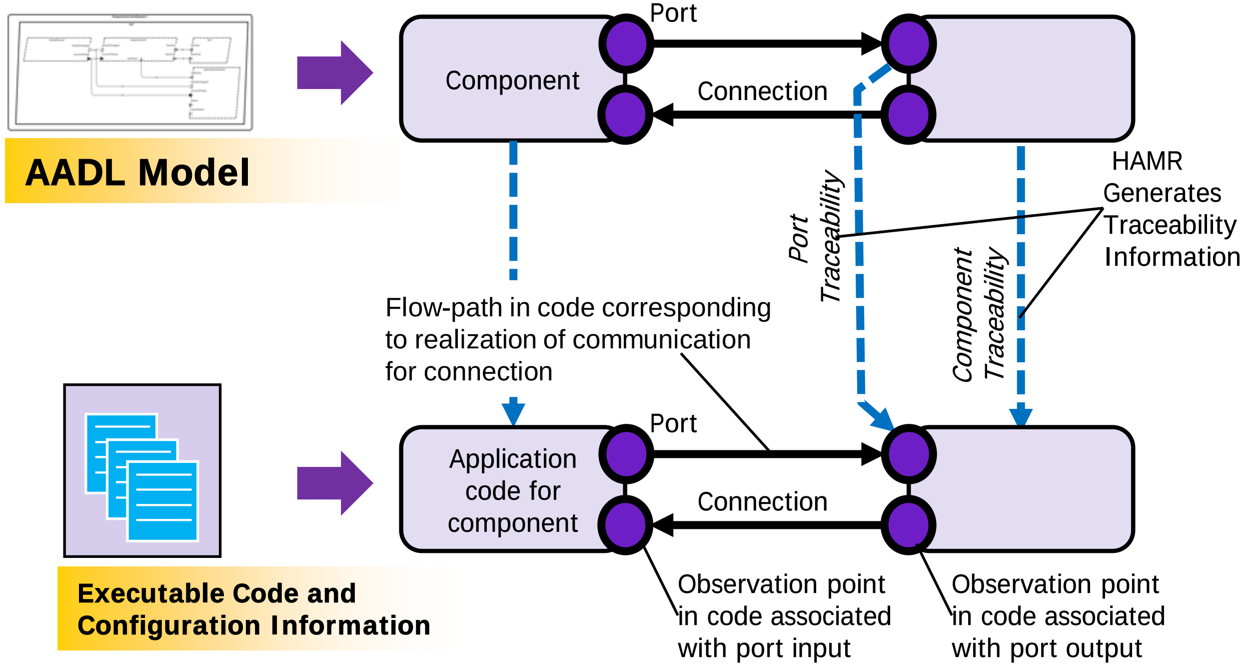
3 CAmkES translates system description into capDL spec

4 HAMR translates AADL model into CAmkES system description

HAMR CORRESPONDENCE PROOF CONCEPT 4

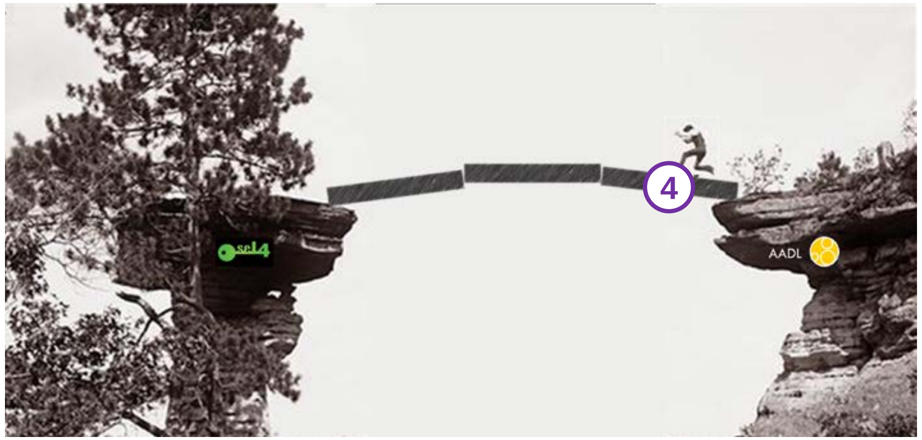
HAMR generates a topological structure using Alloy relations

HAMR generates a topological structure using Alloy relations



FlowPreservation (formal spec in Alloy): For every connection between two components in AADL, there is a flow path in the source code between code artifacts associated with the ports.

NoNewFlows (formal spec in Alloy): For every flow path between two components in the source code, there is a connection in the AADL model between corresponding ports.

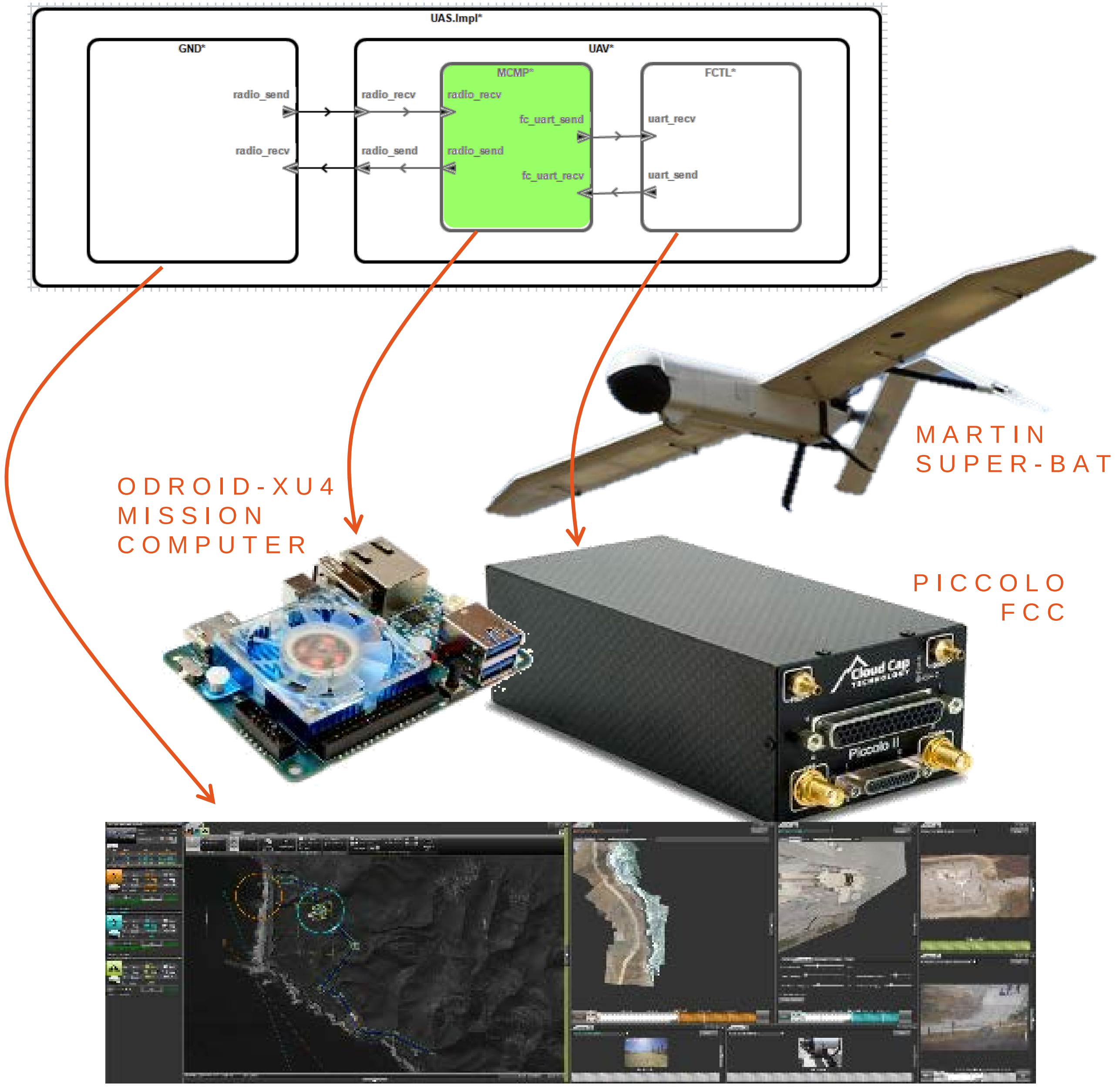


PHASE 2 UAV DEMONSTRATION

AFRL UXAS AUTONOMOUS MISSION PLANNER

- Ground Station sends automation requests to UAV
- UAV Mission Computer processes requests and generates flight plans
- UAV Flight Control Computer computes guidance commands to follow current segment of flight plan

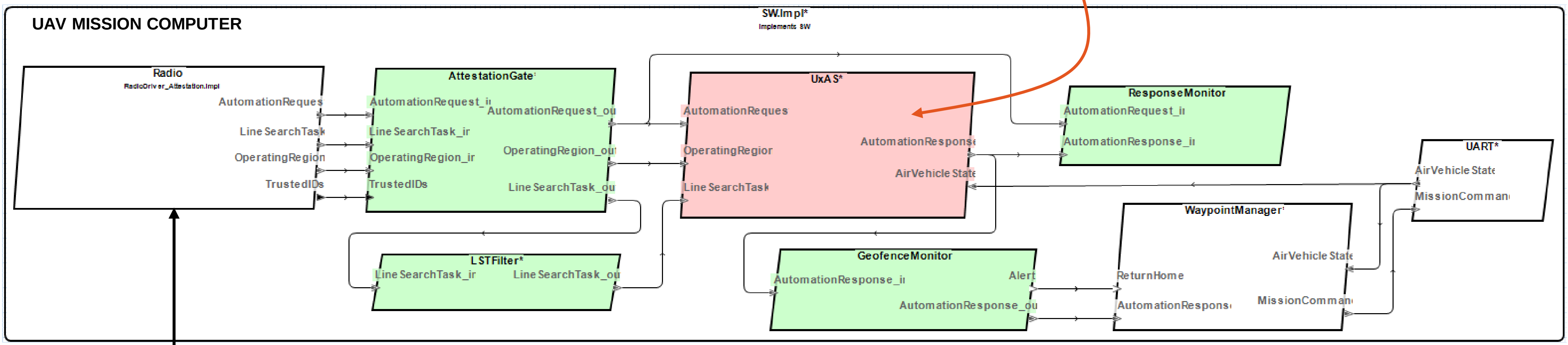
- CASE evaluation team developed cyber attacks on baseline platform
- Collins team hardened platform using BriefCASE tools
- Evaluation team attacks ineffective against hardened platform



GROUND STATION

ATTACKS / MITIGATION

2 Trojans added to UxAS PlanBuilderService Replaces AutomationResponse with plan that violates KeepOutZone



1 Malformed ground station messages

GND STN

3 Code added to ground station software to generate malicious requests

All UAV software runs on formally verified seL4 secure kernel

UxAS software isolated in Linux virtual machine (cyber retrofit)

Specific attacks from evaluation team mitigated:

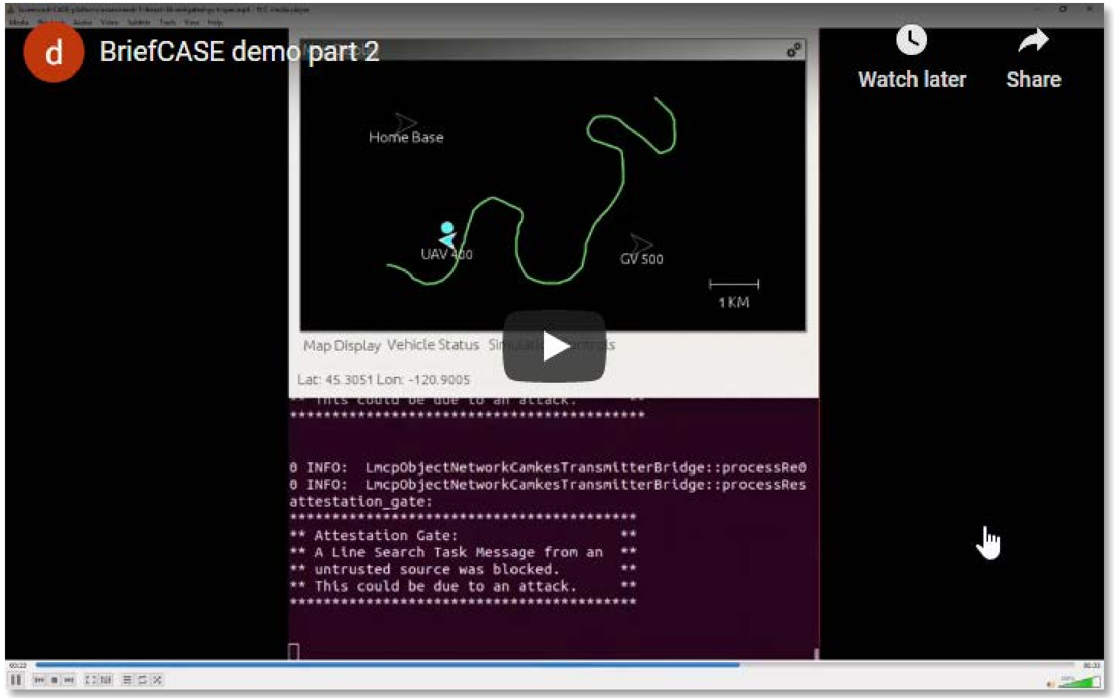
1. Malformed messages blocked by high-assurance filter
2. High-assurance geofence monitor detects plan that violates KeepOutZone and sends alert to WaypointManager to trigger return to base

   Response monitor detects crashed UxAS planner and alerts operator
3. Remote attestation measures ground station software and detects modified code

CASE PHASE 2 DEMO VIDEOS

HTTP://LOONWERKS.COM/PROJECTS/CASE

This video provides a demonstration of the BriefCASE tool environment, showing how to use the tools to address multiple cyber-resiliency requirements for a UAV mission computing system (22:35).

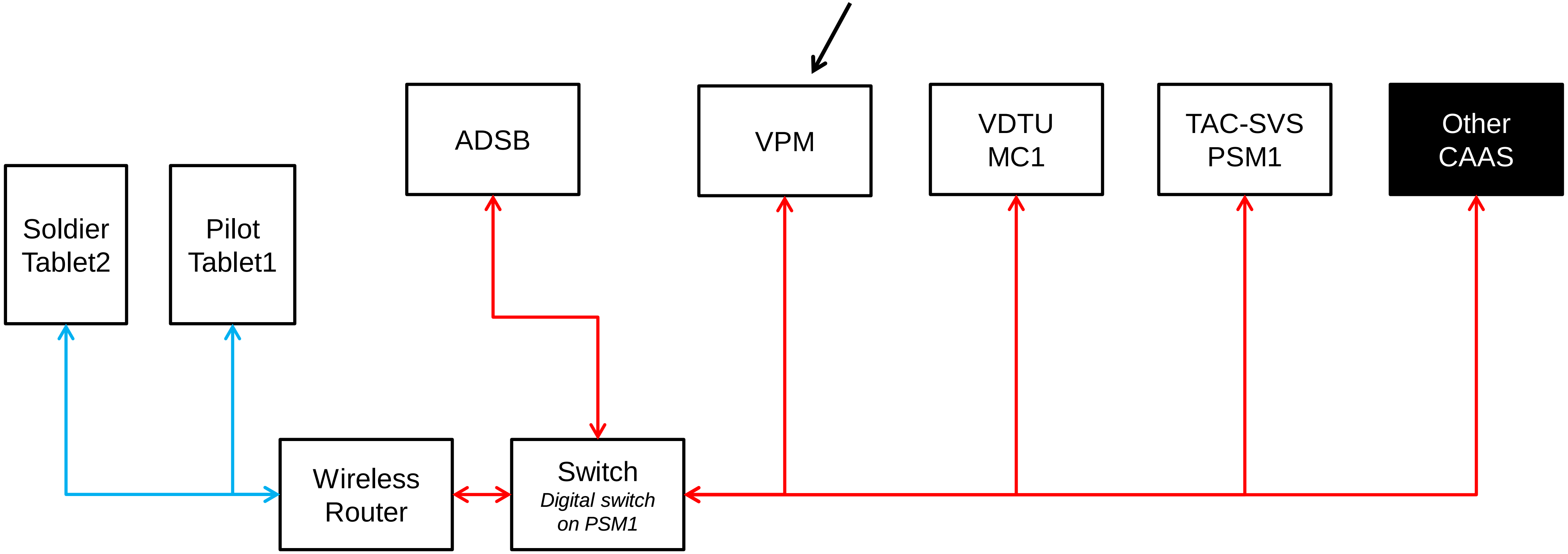


In part two, we run the hardened UAV mission computing system built in the first video and test it against several cyber attacks to show the effectiveness of the approach (10:13).

CASE PHASE 3 DEMONSTRATION

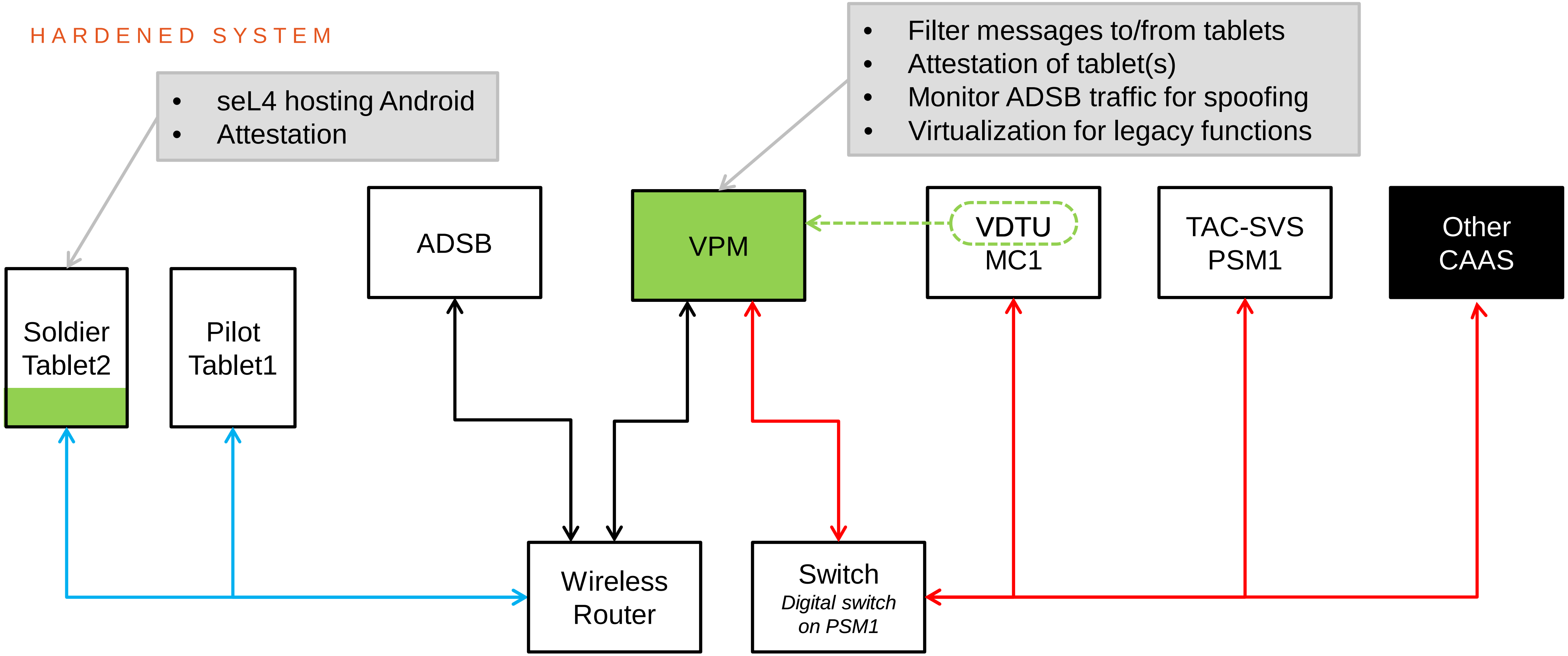
BASELINE SYSTEM (PHYSICAL)

- Collins Common Avionics Architecture System (CAAS)
- Goal : Extend (securely) to add wireless connectivity
- Idea : Repurpose one of the mission processing modules (VPM) to act as guard using BriefCASE AADL tools



CASE PHASE 3 DEMONSTRATION

HARDENED SYSTEM



CONCLUSION

- **Developer assistance to implement cyber-resiliency**
  - Automated architecture transforms for threat mitigation
  - High assurance components generated from specifications
  - Techniques to deal with legacy code (cyber retrofit)
- **MBSE environment for high-assurance cyber-resilient system development**
  - Build system directly from detailed, verified AADL model
  - Makes the security guarantees of seL4 accessible to system developers
  - Ability to target different platforms to facilitate incremental debugging/development
- **Integration of formal verification/proof**
  - Formal verification of functional and cyber-resiliency properties, information flow properties, component proofs
  - Code generation equivalence to model, seL4 build preserves properties
  - Integrate evidence as an assurance case demonstrating how/why requirements are satisfied

Open-source tools : GitHub.com/Loonwerks/formal-methods-workbench
Demo video : Loonwerk.com/projects/case.html

CY-RES
Architectural Transformations for Cyber-Resiliency

MBSEC
Model-Based Systems Engineering for Cybersecurity

PROOF
End-to-End Integrated Formal Verification