

Model-based Development for seL4 Microkit/Rust with Integrated Formal Methods using HAMR

seL4 Summit – Sept 3, 2025

Kansas State University

John Hatcliff

Robby

Jason Belt

Aarhus University

Stefan Hallerstede

With collaborators at ...

Collins Aerospace

Dornerworks

UNSW

Proofcraft

Carnegie Mellow Univ.

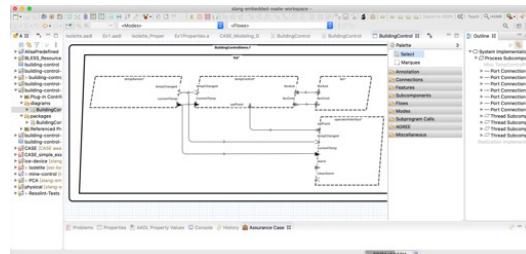
Univ. of Kansas

HAMR - SysMLv2/AADL to Rust + sel4

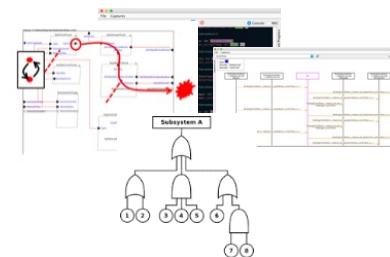
HAMR

HAMR – tool chain for [H]igh [A]ssurance [M]odeling and [R]apid engineering for embedded systems

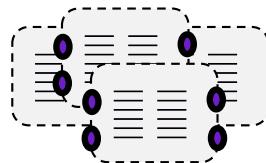
Modeling, analysis, and verification in the **SysMLv2** and **AADL** modeling languages



Model-level behavior specifications (e.g., contracts) and analyses



Component development, automated testing, and verification in multiple languages



- C
- Rust with Verus verification
- Slang (developed at Kansas State)
 - safety-critical subset of Scala
 - contract-based verification
 - transpiles to C and Rust

Deployments aligned with AADL run-time on multiple platforms

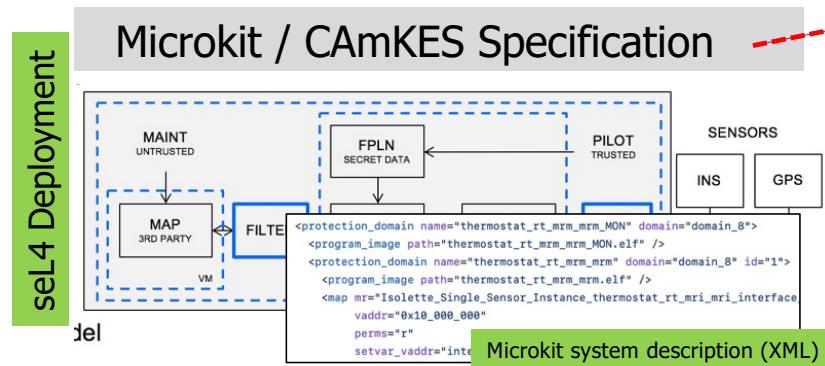
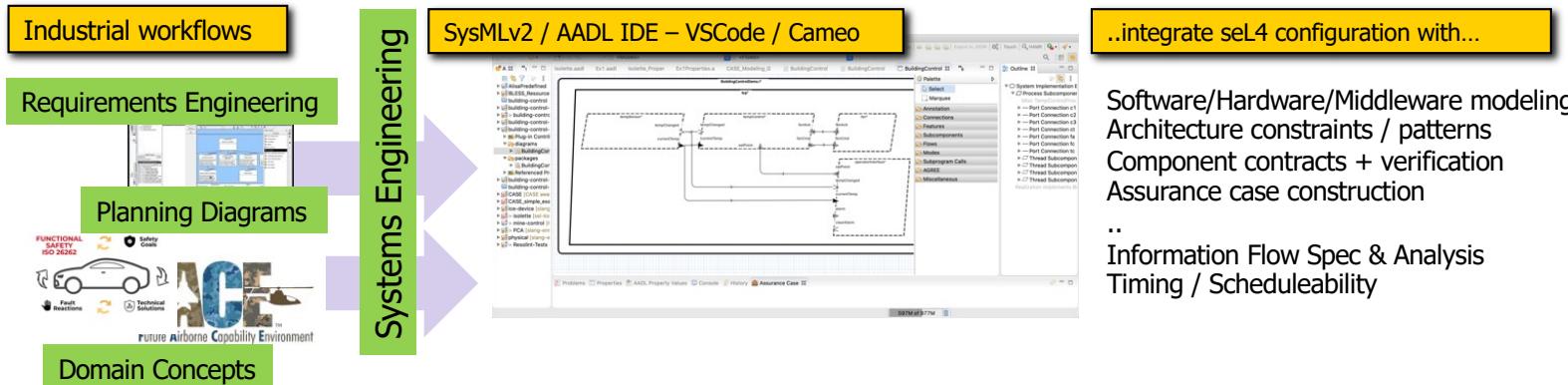


CAmkES & microKit

HAMR - SysMLv2/AADL to Rust + seL4

Potential Benefits to seL4 Application Developers

A **systems engineering environment** based on standardized modeling languages (SysMLv2, AADL) with accompanying analysis, verification, and assurance case tools

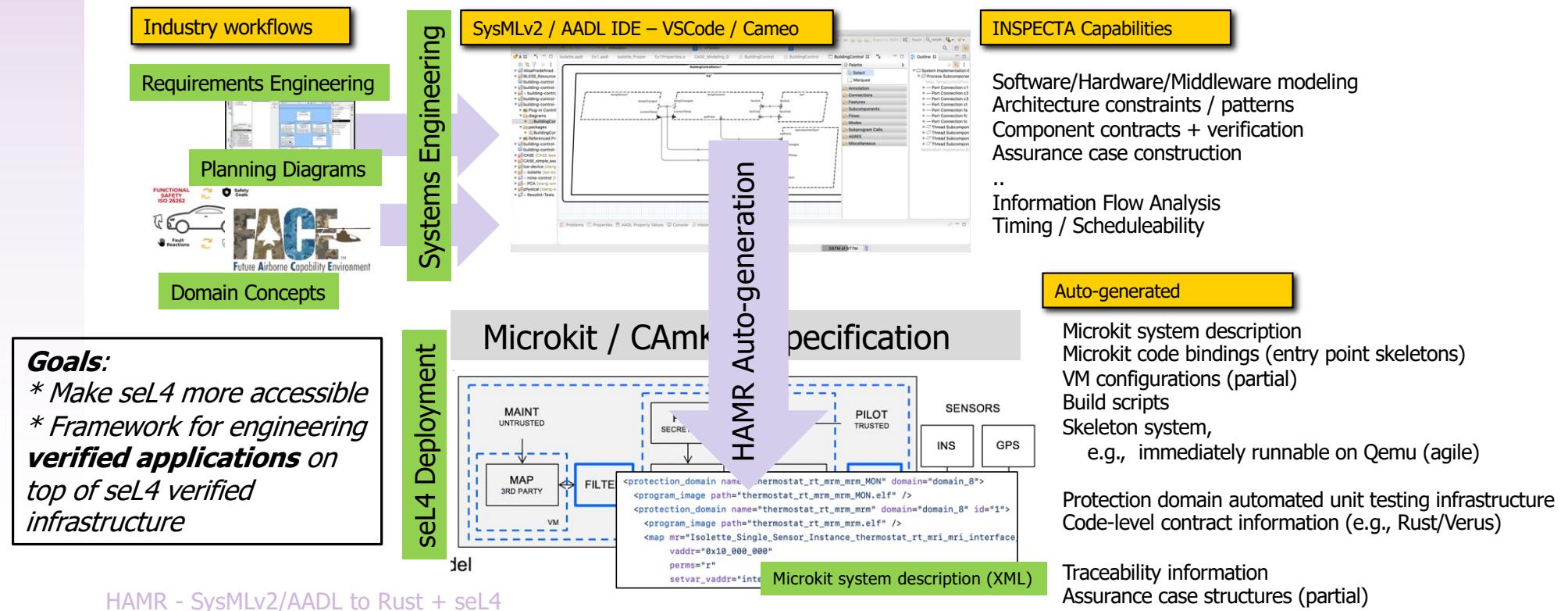


Existing seL4 ecosystem tools for component-oriented specification of seL4 capabilities, partitioning, and inter-partition communication

HAMR - SysMLv2/AADL to Rust + seL4

Potential Benefits to seL4 Application Developers

A **systems engineering environment** based on standardized modeling languages (SysMLv2, AADL) with accompanying analysis, verification, and assurance case tools



Context and Target Applications

On the DARPA PROVERS program, HAMR is being used to develop an experimental version of the mission computer for the Collins "Launched Effects" platform (final development will emphasize HAMR SysMLv2 to Rust)



Collins Aerospace



Launched Effects product line
= tube-launched, expendable UAVs

Launched Effects Mission Computer



**..increase security
and modularity**

..decrease costs for
development and
assurance

Video: <https://youtu.be/SwPJHmZQMaM?si=NwTdb3VFpV-MxSre>

DARPA PROVERS

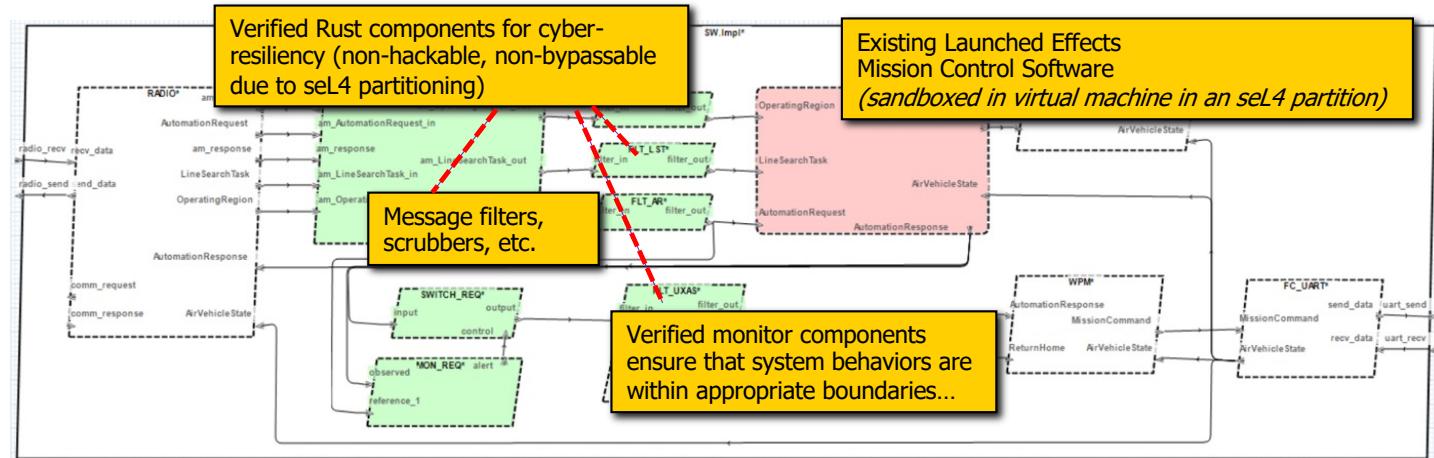
= "integrating pipelines of formal methods in
defense industry development processes"



HAMR - SysMLv2/AADL to Rust + seL4

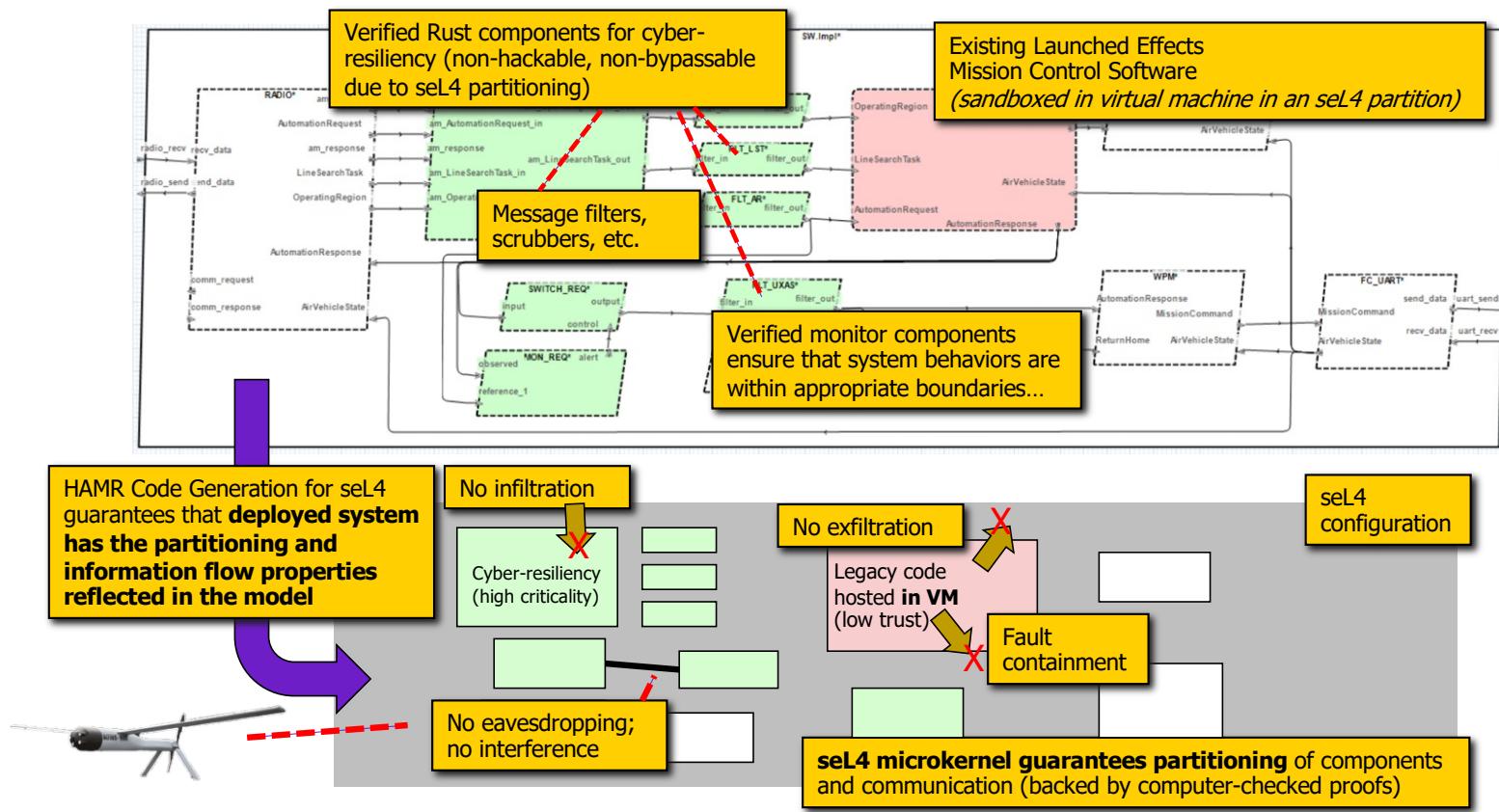
Characteristics of Supported Systems

Use HAMR SysMLv2/AADL modeling to specify partitioning, communication architecture of improved system



Characteristics of Supported Systems

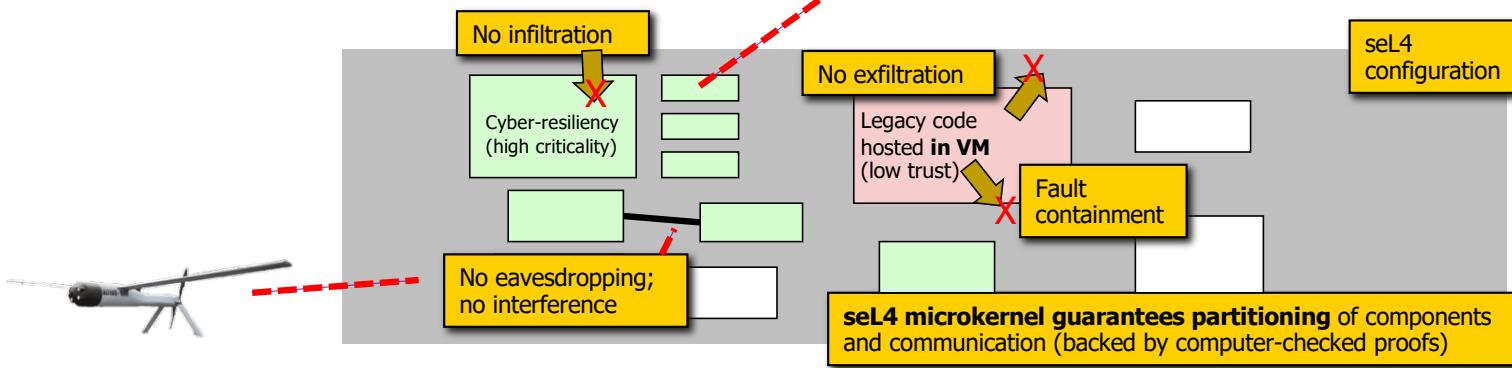
Use HAMR SysMLv2/AADL modeling to specify partitioning, communication architecture of improved system



HAMR - SysMLv2/AADL to Rust + seL4

Verified Component Code

PROVERS program
emphasis on memory
safe languages...



HAMR - SysMLv2/AADL to Rust + seL4

Other Summit INSPECTA-Related Talks

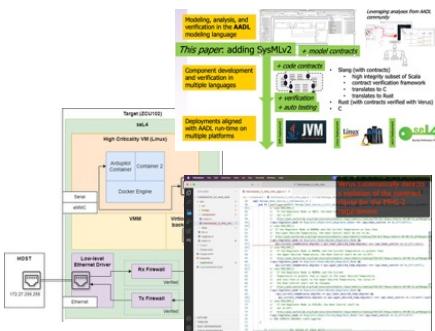


Darren Cofer (Principal Investigator) –
Application to Collins Launched Effects

“Integration of seL4 in a Flight
Vehicle Mission System”

Gerwin Klein –
Automating seL4 kernel correctness proofs
for new platforms

“The next 700 verified seL4
platforms”



Robert VanVossen –
Rust contract-based development, testing,
and verification of firewall components

“Rust-based drivers and verified rust
applications on seL4”

Gernot Heiser –
Verified infrastructure and services

“Trustworthy Systems R&D Update”

Junming Zhao –
Verified infrastructure and services

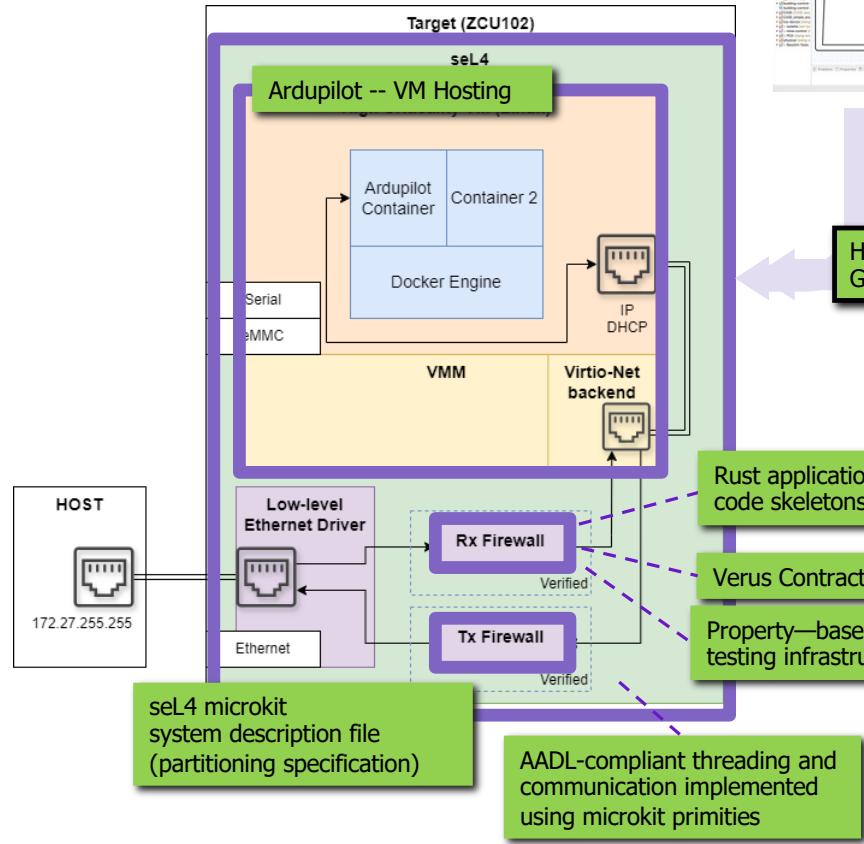
“Verifying Device Drivers with Pancake”

Robert Vanvossen Talk

Dornerworks

"Rust-based drivers and verified rust applications on seL4" -- Thursday

INSPECTA – Public Demonstrator Example



HAMR - SysMLv2/AADL to Rust + seL4



HAMR Generation

+ Behavior Contracts
(formalized
Firewall Rules)

Verus Verification that Firewall implementation conforms to Requirements

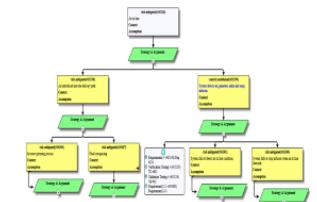
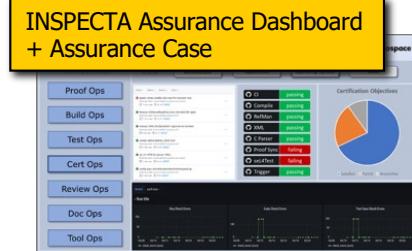
Automated property-based testing that Firewall implementation conforms to Requirements



Recent demo!

Automated..

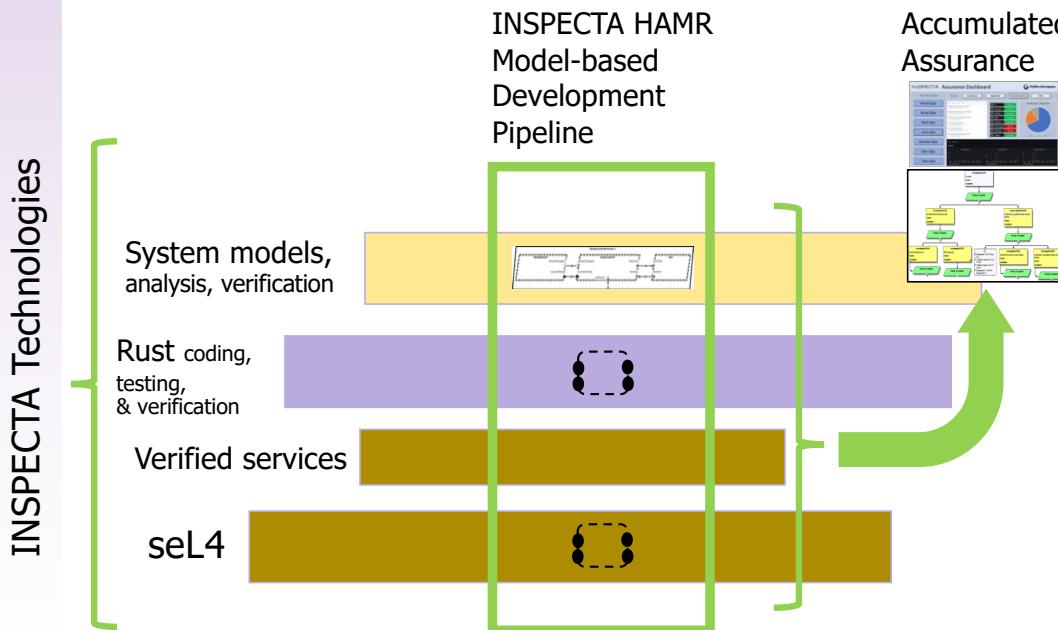
- metrics generation
- traceability info
- attestation (w/ KU)
- assurance case evidence



...Dornerworks found a *10x reduction in development time*.

INSPECTA “PROVERS Pipeline” Scope

A primary goal of PROVERS is to demonstrate “pipelines” of formal methods capabilities. Designing and managing the INSPECTA “pipeline” entails a lot of extra work...



To fully demonstrate pipeline concepts within program timeline, the scope of the pipeline needs to be narrower than that of the individual technologies

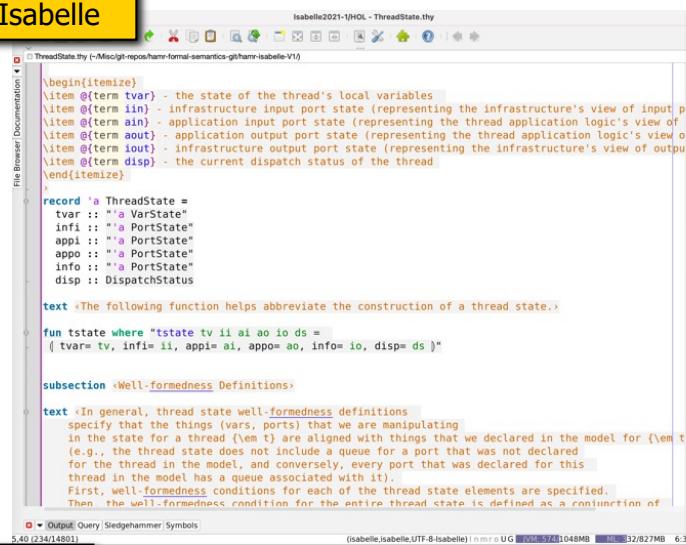
HAMR - SysMLv2/AADL to Rust + seL4

- Interactions across the pipelines stages are organized a **core** set of computational and data **abstractions** that are amenable to **formal verification**
- **Semantics** of these abstractions must be maintained and **traceable across the stages**
- Claims, contributions of stages, and assurance evidence must be accumulated across the stages

HAMR Formal Semantics for INSPECTA Pipeline

150+ page literate-style Isabelle/HOL theories for AADL/SysMLv2 HAMR execution model (guides our design of our contracts and verification/testing framework)

Isabelle



```
record 'a ThreadState =
  tvar :: 'a VarState
  infi :: 'a PortState
  appi :: 'a PortState
  appo :: 'a PortState
  info :: 'a PortState
  disp :: DispatchStatus

text <The following function helps abbreviate the construction of a thread state.>

fun tstate where "tstate tv ii ai ao io ds =
  (tvar= tv, infi= ii, appi= ai, appo= ao, info= io, disp= ds)"

subsection <Well-formedness Definitions>

text <In general, thread state well-formedness definitions specify that the things (vars, ports) that we are manipulating in the state are aligned with things that we declared in the model for t. (e.g., the thread state does not include a queue for a port that was not declared for the thread in the model, and conversely, every port that was declared for this thread in the model has a queue associated with it). First, well-formedness conditions for each of the thread state elements are specified. Then, the well-formedness condition for the entire thread state is defined as a conjunction of these properties.>
```

PROVERS

- Enhanced and scope expanded
- Prove soundness of contract framework
- Extend formalization downwards towards seL4 proof-base

Latex/PDF generated from Isabelle

Joint work with
Stefan Hallerstede
(U. Aarhus)

```
record 'a ThreadState =
  tvar :: 'a VarState
  infi :: 'a PortState
  appi :: 'a PortState
  appo :: 'a PortState
  info :: 'a PortState
  disp :: DispatchStatus

The following function helps abbreviate the construction of a thread state.

fun tstate where tstate tv ii ai ao io ds =
  (tvar= tv, infi= ii, appi= ai, appo= ao, info= io, disp= ds)

2.4.2 Well-formedness Definitions

In general, thread state well-formedness definitions specify that the things (vars, ports) that we are manipulating in the state are aligned with things that we declared in the model for t. (e.g., the thread state does not include a queue for a port that was not declared for the thread in the model, and conversely, every port that was declared for this thread in the model has a queue associated with it). First, well-formedness conditions for each of the thread state elements are specified. Then, the well-formedness condition for the entire thread state is defined as a conjunction of these properties.

Well-formed Thread State Elements

definition wf_ThreadState-tvar:: Model ⇒ ComplId ⇒ ('a VarState) ⇒ bool where
wf_ThreadState-tvar m c v ≡ wf-VarState vs {v . isVarOfCID m c v}

The infi component of a ThreadState (input infrastructure port map) is well formed when the domain of the infi port map is equal to the set of input ports for the thread declared in the model. Intuitively, each of the declared "in" ports for the thread (according to the model) is associated with a infrastructure message queue, (and there are no "extra" ports in the map).

definition wf_ThreadState-infi:: Model ⇒ ComplId ⇒ ('a PortState) ⇒ bool where
wf_ThreadState-infi m c ps ≡ wf-PortState ps {p . isInCIDPID m c p}

The definitions below for other port-state elements are similar.

definition wf_ThreadState-appi:: Model ⇒ ComplId ⇒ ('a PortState) ⇒ bool where
wf_ThreadState-appi m c ps ≡ wf-PortState ps {p . isInCIDPID m c p}

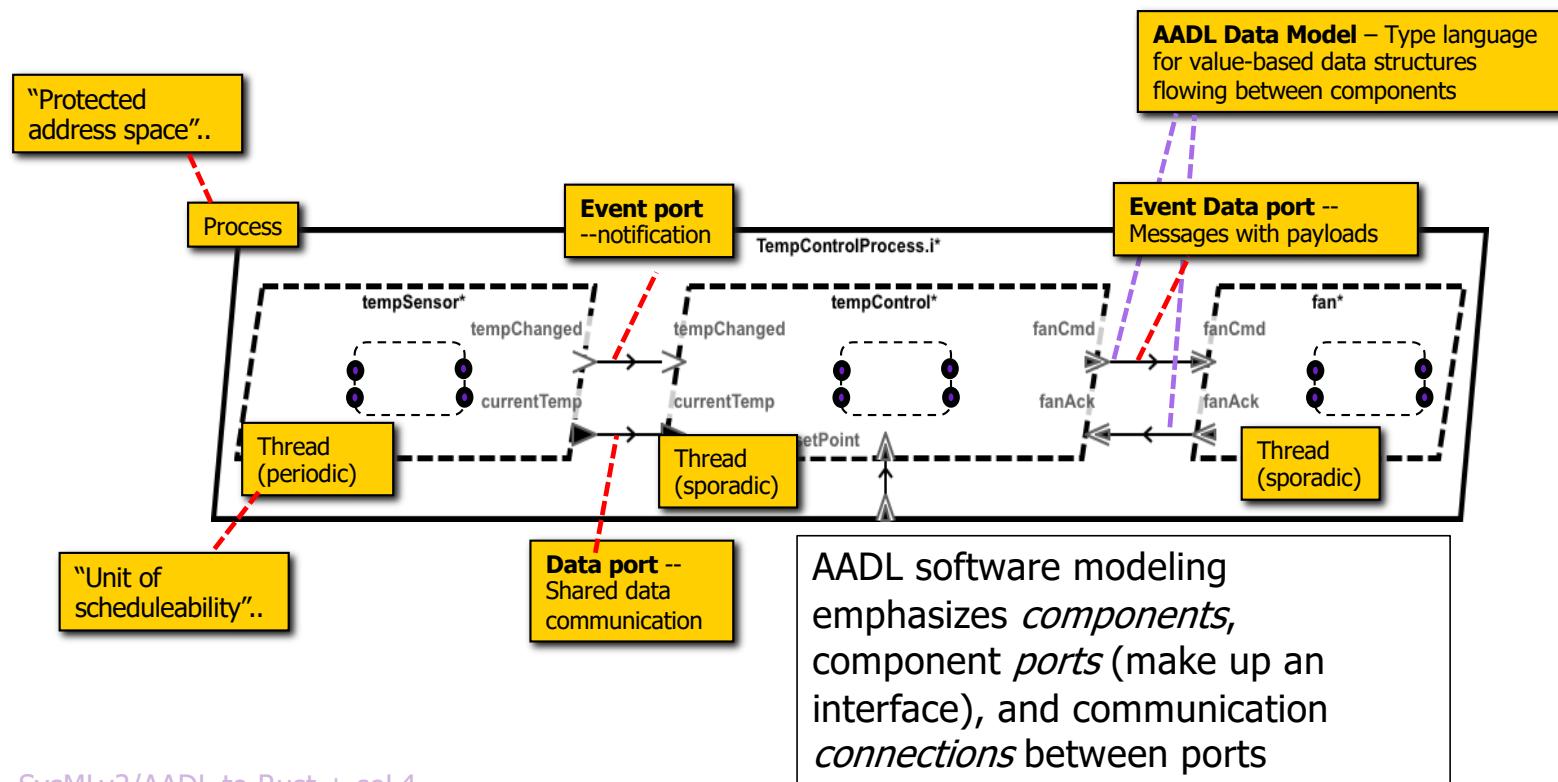
definition wf_ThreadState-appo:: Model ⇒ ComplId ⇒ ('a PortState) ⇒ bool where
wf_ThreadState-appo m c ps ≡ wf-PortState ps {p . isOutCIDPID m c p}

definition wf_ThreadState-info:: Model ⇒ ComplId ⇒ ('a PortState) ⇒ bool where
wf_ThreadState-info m c ps ≡ wf-PortState ps {p . isOutCIDPID m c p}
```

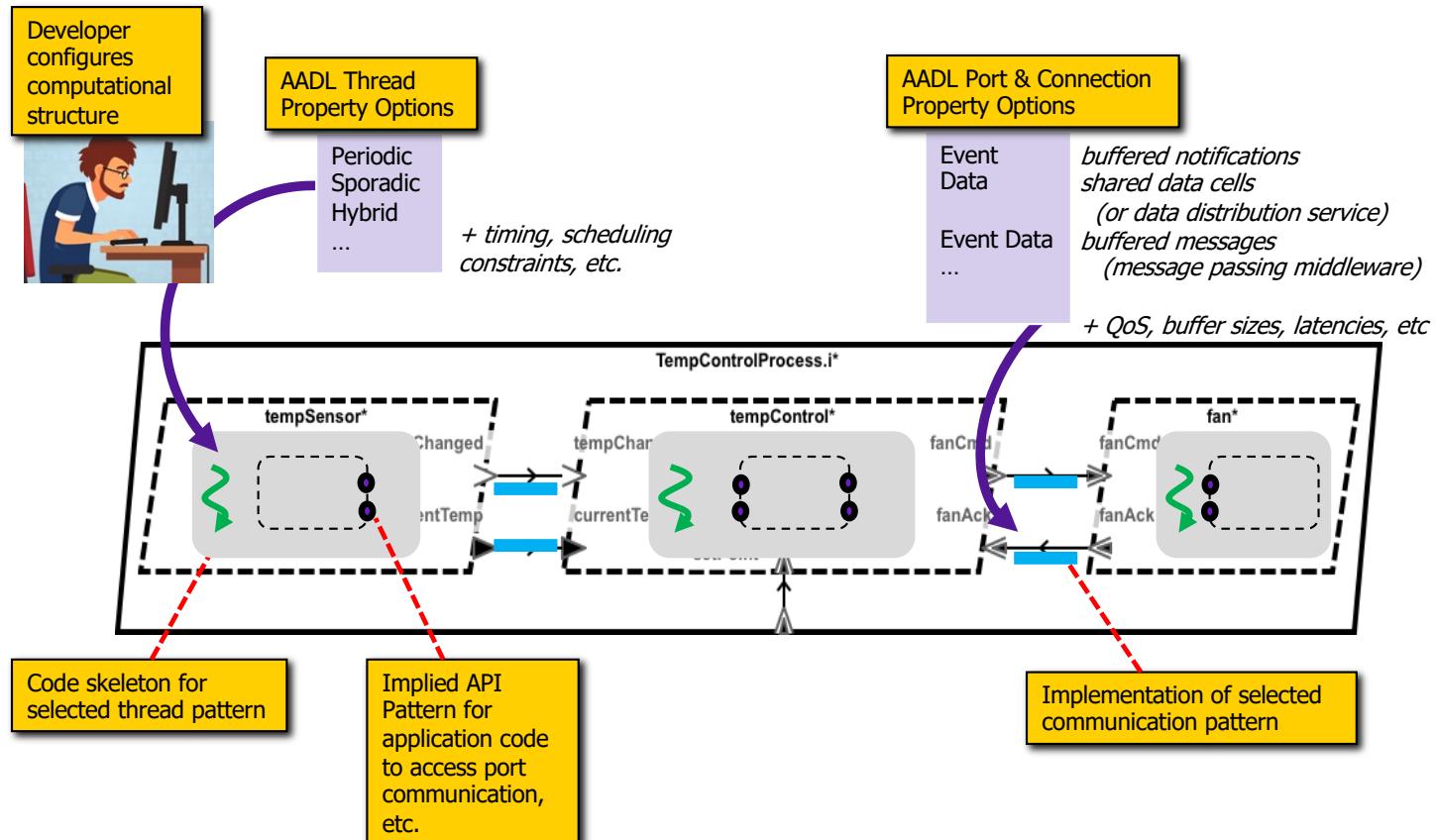
Note limited scope: HAMR subset of AADL/SysMLv2; run-time semantics; connection to code generator by manual inspection

AADL Modeling Concepts

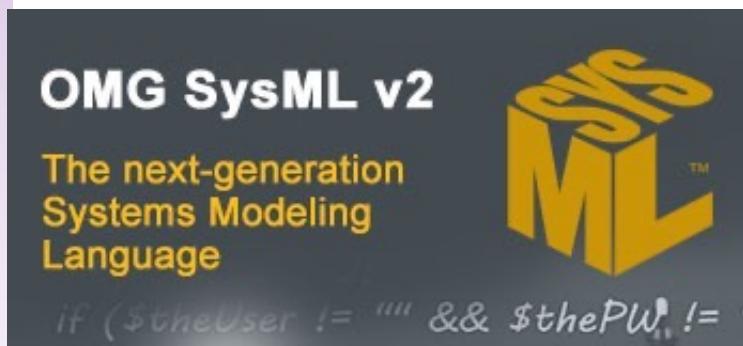
Each AADL modeling element is classified according to its role in embedded system architecture...



AADL Modeling Concepts



AADL to SysMLv2



Why might SysMLv2 provide a alternate vehicle for rigorous model-based development, including AADL concepts?

- Like AADL, has both a graphical view and textual view
- Many AADL modeling elements have analogues in SysMLv2
 - E.g., components, ports, connections, developer-defined attributes
- Aims to provide a stronger “semantics” for system engineering compared to UML, SysMLv1
- Re-engineered from the ground up
 - No backwards compatibility with SysMLv1 except through translation
 - Not built as a profile of UML
- Will have wide-ranging commercial tool support as well as open source implementations

Standardization Effort - Migrating AADL to SysMLv2



About the SMC



The OMG Systems Modeling Community gathers people interested in advancing SysMLv2

Different membership structure

See <https://www.omg.org/communities/>

RTESC Workgroup – entity responsible for integrating AADL concepts into SysMLv2

Charter: "Develop domain libraries w/ KerML & SysMLv2 to support the precise modeling of Real-Time Embedded Safety-Critical Systems. Integrate capabilities from domain-specific models like SAE AADL, OpenGroup FACE, OMG MARTE, & AutoSAR"

Lead: Gene Shreve (i3-Corp), Jerome Hugues (CMU/SEI)

- Working with OMG RTECS working group to prototype AADL concepts in SysMLv2
- We are one of the most active participants working on building end-to-end tools for formal methods and code generation
- "Trail blazers" on integrating formal contract languages in SysMLv2 IDE

VSCode SysMLv2 HAMR Front End

We developed a VSCode SysMLv2 HAMR front-end based on the SysIDE VSCode plug-in

SysMLv2 component interfaces

Code-level artifacts in same IDE: Integration of MicroKit-based Slang, Rust and C development for seL4

AADL Library Properties as SysMLv2 attributes

SysMLv2 encodings of datatypes specified using AADL Data Modeling Language

Formal behavior specifications in GUMBO contract language

Verification results for model-level contracts

```
part def Manage_Heat_Source_i_ >> Thread {
    attribute ::> Dispatch_Protocol = Supported_Dispatch_Protocols::Periodic;
    attribute ::> Period = 1000 [millisecond];
    attribute Domain: CASE_Scheduling::Domain = 9;

    // ===== INPUTS =====
    // current temperature (from temp sensor)
    in port current_tempStatus : DataPort { in ::> type : Isolette_Data_Model::TempWStatus_i; }
    // lowest and upper bound of desired temperature range
    in port lower_desired_temp : DataPort { in ::> type : Isolette_Data_Model::Temp_i; }
    in port upper_desired_temp : DataPort { in ::> type : Isolette_Data_Model::Temp_i; }
    // subsystem mode
    in port regulator_mode : DataPort { in ::> type : Isolette_Data_Model::Regulator_Mode; }

    // ===== OUTPUTS =====
    // command to turn heater on/off (actuation command)
    out port heat_control : DataPort { out ::> type : Isolette_Data_Model::On_Off; }

language "GUMBO" /*
    // indicate that the component maintains an internal state (variables) that influence its behavior
    state
        lastCmd: Isolette_Data_Model::On_Off;

    // ===== Initialize Entry Point Behavior Constraints =====
    initialize
        guarantee
            initlastCmd: lastCmd == Isolette_Data_Model::On_Off.Off;
            guarantee REQ_MHS_1 "If the Regulator Mode is INIT, the Heat Control shall be
                |set to Off
                |http://pub.santoslab.org/high-assurance/module-requirements/readings/FAA-DOT-
                heat_control == Isolette_Data_Model::On_Off.Off;
*/
```

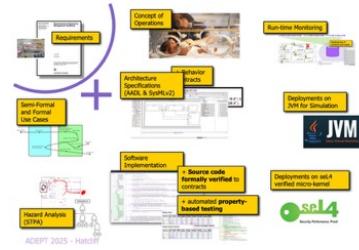
SysMLv2/AADL for HAMR with seL4 Microkit

Artifacts & Workflow -- Detailed Technical Report

Isolette – Infant Incubator



- 9 Real-time Tasks
- ~40 component-level requirements
- Interesting modal behavior



End-to-end Artifacts

- ConOps
- Use Cases
- Requirements
- Models
- Contracts
- Testing
- Verification
- Assurance Case

HAMR - SysMLv2/AADL to Rust + seL4

The Isolette System: Illustrating End-to-End Artifacts for Rigorous Model-based Engineering

(Collins Aerospace INSPECTA Technical Report)

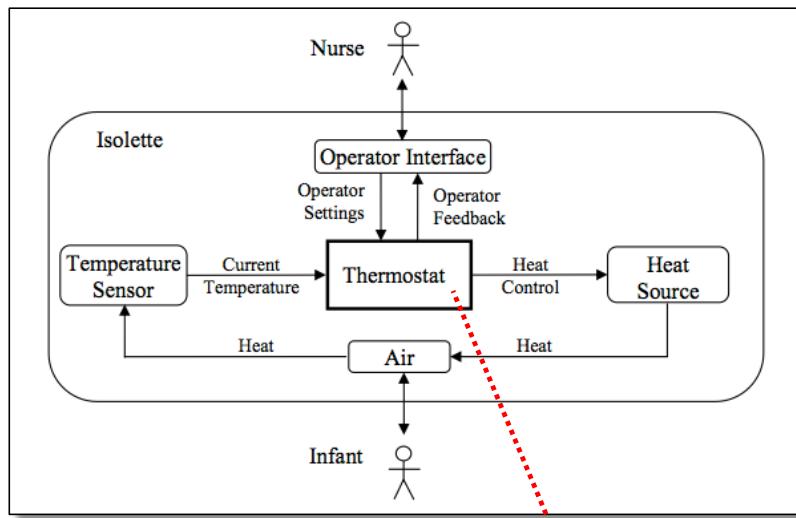
John Hatchiff and Jason Belt

The Isolette System: Illustrating End-to-End Artifacts for Rigorous Model-based Engineering (Collins Aerospace INSPECTA Technical Report)	1
John Hatchiff and Jason Belt	1
1 Introduction	1
1.1 Background, Scope, and Research Context	3
1.2 Document Objectives	3
1.3 Example Choice Justification and Limitations	3
1.4 Resources	4
2 HAMR Approach to Formal Methods-Integrated Model-Based Development	5
2.1 Motivation for Scoping/Research/Development Approach	5
2.2 HAMR Scoping for Code Generation and Formal Methods Integration	5
2.3 HAMR Computational Paradigm for Formal Methods Integrated Components	7
3 Artifact Overview	10
4 Requirements	11
4.1 Informal Design	12
4.2 INSPECTA Development Tasks for Requirements and Initial Design	13
5 Models	15
5.1 AADL Model	15
5.2 SysMLv2 AADL Profile Translation	19
6 Model-level Behavioral Specifications	19
6.1 AADL	19
6.2 SysMLv2	23
7 Component Development	23
7.1 Thread Crate Structure	23
7.2 Code Generation with Embedded Contracts	25
7.3 Coding Application Logic	26
7.4 Thread API: Port Access and Integration Constraints	26
8 Component Testing	27
8.1 Manual Unit Testing Framework	28
8.2 Contract Checking via GUMBOX	30
8.3 Toward Property-Based Testing	33
9 Component Verification	33
10 System Scheduling	34
11 System Simulation and Visualization	36
12 Next Steps in the Development of This Report	36
A Component Development: AADL	37
A.1 Code Generation with Embedded Contracts	37
A.2 Coding Application Logic	38
B Component Testing - Slang	39
B.1 Manual Unit Testing Framework	39
B.2 Property-based Testing Framework Overview	41
B.3 GUMBOX Artifacts Illustrated	42
B.4 Assurance Artifacts	45
C Component Verification - Slang	47
D System Simulation and Visualization - Slang	49

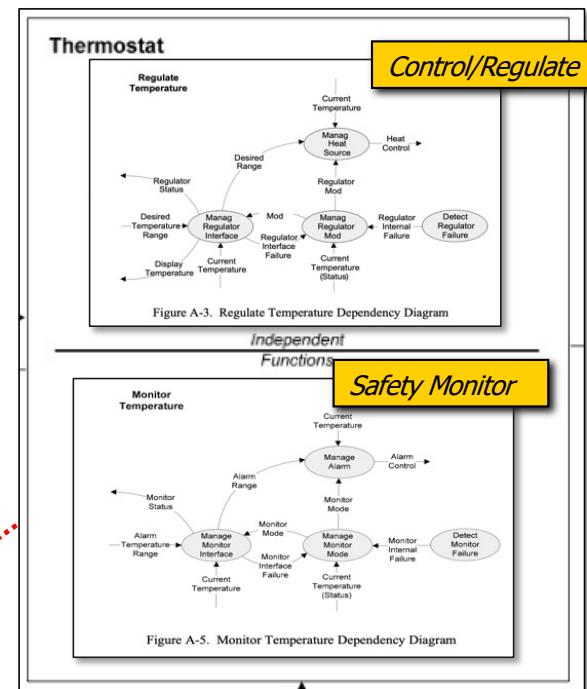
50+ page report w/
Git repo and videos

REMH - Informal Designs

The FAA REMH decomposes the Isolette into a control system and safety monitor subsystem with three tasks each

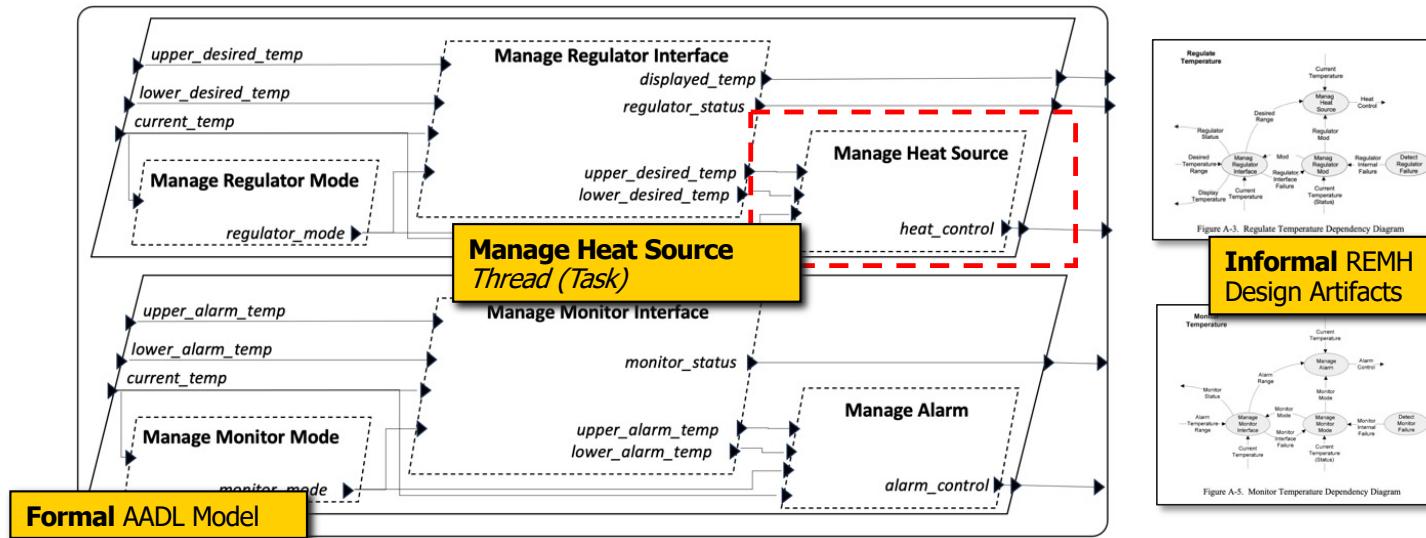


Thermostat decomposed into Regulate Temperature and Monitor Temperature functions.



Using AADL to Represent Design

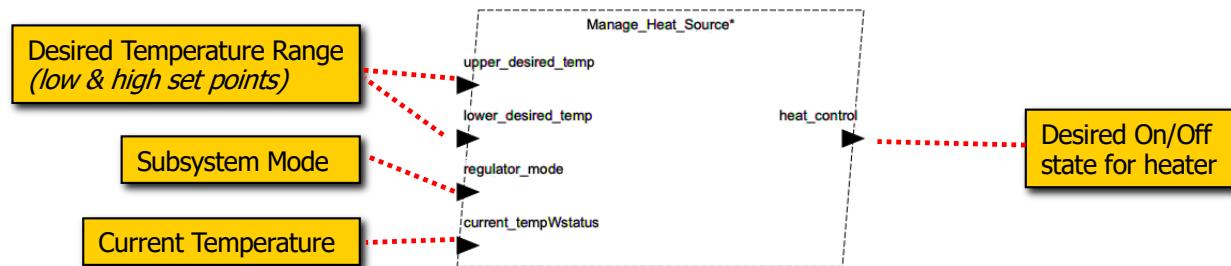
AADL Model is a straightforward rendering of the design diagrams in the FAA REMH



*This example (software aspects) is worked **completely end-to-end** from requirements, to contracts, to automatically tested and verified application code, to deployment on seL4, Linux, JVM, JavaScript.
All artifacts are publicly available.*

Manage Heat Source Thread

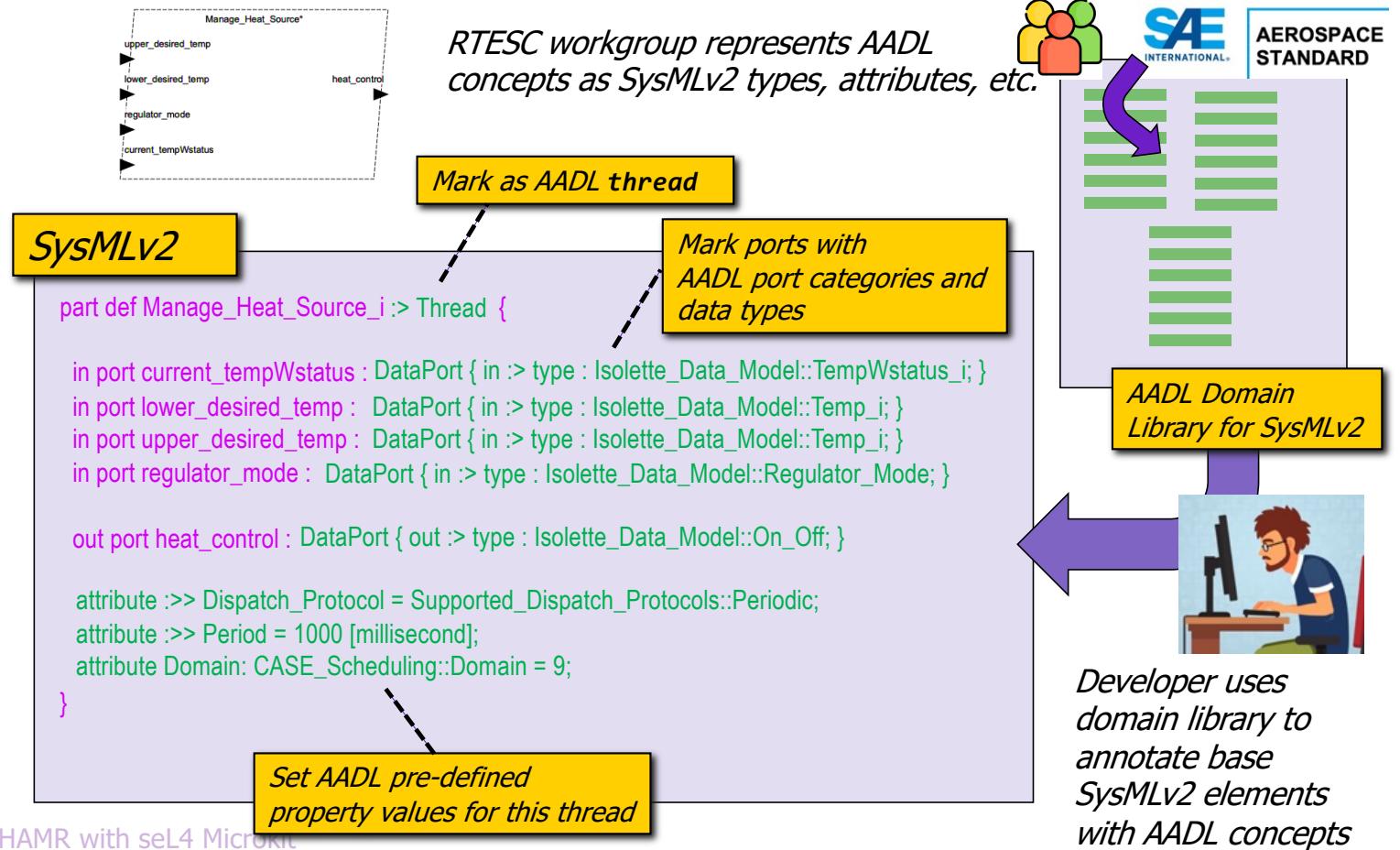
AADL Interface for Manage Heat Source Thread



```
thread Manage_Heat_Source
  features
    -- ====== INPUTS ======
    -- ("Current Temperature") - current temperature (from temp sensor)
    current_tempWstatus: in data port Isolette_Data_Model::TempWstatus.impl;
    -- ("Desired Range") - lowest and upper bound of desired temperature range
    lower_desired_temp: in data port Isolette_Data_Model::Temp.impl;
    upper_desired_temp: in data port Isolette_Data_Model::Temp.impl;
    -- ("Regulator Mode") - subsystem mode
    regulator_mode: in data port Isolette_Data_Model::Regulator_Mode;

    -- ====== OUTPUTS ======
    -- ("Heat Control") - command to turn heater on/off (actuation command)
    heat_control: out data port Isolette_Data_Model::On_Off;
```

SysMLv2 + AADL Modeling Concepts



AADL / SysMLv2 Component Types Side-by-Side

AADL

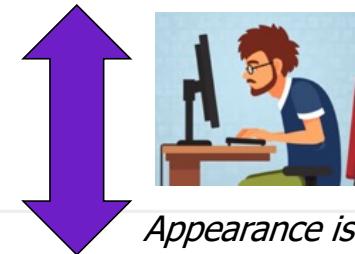
```
thread Manage_Heat_Source
  features
    current_tempWstatus: in data port Isolette_Data_Model::TempWstatus.impl;
    lower_desired_temp: in data port Isolette_Data_Model::Temp.impl;
    upper_desired_temp: in data port Isolette_Data_Model::Temp.impl;
    regulator_mode: in data port Isolette_Data_Model::Regulator_Mode;
    heat_control: out data port Isolette_Data_Model::On_Off;

  properties
    Dispatch_Protocol => Periodic;
    Period => Isolette_Properties::ThreadPeriod;
```

SysMLv2

```
part def Manage_Heat_Source_i :> Thread {
  in port current_tempWstatus : DataPort { in :> type : Isolette_Data_Model::TempWstatus_i; }
  in port lower_desired_temp : DataPort { in :> type : Isolette_Data_Model::Temp_i; }
  in port upper_desired_temp : DataPort { in :> type : Isolette_Data_Model::Temp_i; }
  in port regulator_mode : DataPort { in :> type : Isolette_Data_Model::Regulator_Mode; }
  out port heat_control : DataPort { out :> type : Isolette_Data_Model::On_Off; }

  attribute :>> Dispatch_Protocol = Supported_Dispatch_Proocols::Periodic;
  attribute :>> Period = 1000 [millisecond];
  attribute Domain: CASE_Scheduling::Domain = 9;
```



Appearance is similar

Challenges

Challenges in migrating AADL Formal Methods to SysMLv2

- SysMLv2 has no “annex mechanism”; need to figure out how to represent AADL Annexes
 - ~~behavior contracts, architectural constraints language, hazard analysis~~
- Representation of AADL Properties
 - model configuration parameters
- Formal semantics of run-time behavior
 - Development of SysMLv2 “semantics” and “formal methods” is spread across several OMG working groups and is struggling to focus
 - SysMLv2 is big and general, so it is hard for committees to develop a precise semantics that satisfies their committee mandate

Natural Language Requirements for Thread

FAA REMH requirements for **Manage Heat Source** task

Requirements for control laws of this task...

REQ-MHS-1: If the Regulator Mode is INIT, the Heat Control shall be set to Off.

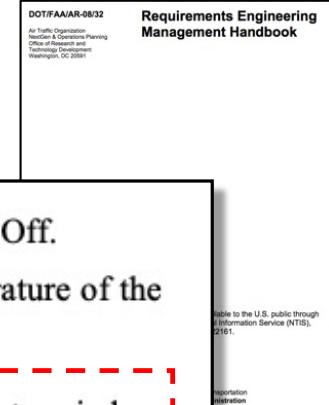
Rationale: A regulator that is initializing cannot regulate the Current Temperature of the Isolette and the Heat Control should be turned off.

REQ-MHS-2: If the Regulator Mode is NORMAL and the Current Temperature is less than the Lower Desired Temperature, the Heat Control shall be set to On.

REQ-MHS-3: If the Regulator Mode is NORMAL and the Current Temperature is greater than the Upper Desired Temperature, the Heat Control shall be set to Off.

REQ-MHS-4: If the Regulator Mode is NORMAL and the Current Temperature is greater than or equal to the Lower Desired Temperature and less than or equal to the Upper Desired Temperature, the value of the Heat Control shall not be changed.

REQ-MHS-5: If the Regulator Mode is FAILED, the Heat Control shall be set to Off.



Component Requirements to Contracts

GUMBO contracts are written together with the thread interface in the VSCode SysIDE plug-in **using a customized editor extension that we developed to support contracts**

The screenshot shows a VSCode interface with several tabs open:

- `Regulate.sysml`: Shows a component named `Regulate` with code for managing heat sources and defining regulator modes.
- `Operator_Interface.sysml`: Shows a component named `Operator_Interface`.
- `Monitor.sysml`: Shows a component named `Monitor`.
- `Isolette.sysml`: Shows a component named `Isolette`.
- `thermostat_rt_mhs_mhs_user.c`: A C file containing system requirements.

A yellow box labeled "Component interface" highlights the code in the `Regulate.sysml` tab, specifically the thread definition and port declarations.

A yellow box labeled "Component contract" highlights the code in the `thermostat_rt_mhs_mhs_user.c` tab, specifically the system requirements (REQ-MHS-1 through REQ-MHS-5).

A yellow box labeled "Heat Controller Task natural language functional requirements (control laws)" contains the following text:

REQ-MHS-1: If the Regulator Mode is INIT, the Heat Control shall be turned off.
Rationale: A regulator that is initializing cannot regulate Isolette and the Heat Control should be turned off.

REQ-MHS-2: If the Regulator Mode is NORMAL, and the Current Temperature is less than the Lower Desired Temperature, the Heat Control shall be turned on.
Rationale: If the Regulator Mode is NORMAL, and the Current Temperature is less than the Lower Desired Temperature, the Heat Control should be turned on.

REQ-MHS-3: If the Regulator Mode is NORMAL, and the Current Temperature is greater than or equal to the Upper Desired Temperature, the value of the Heat Control shall not be changed.
Rationale: If the Regulator Mode is NORMAL, and the Current Temperature is greater than or equal to the Upper Desired Temperature, the value of the Heat Control shall not be changed.

REQ-MHS-4: If the Regulator Mode is NORMAL, and the Current Temperature is greater than or equal to the Lower Desired Temperature and less than or equal to the Upper Desired Temperature, the value of the Heat Control shall not be changed.
Rationale: If the Regulator Mode is NORMAL, and the Current Temperature is greater than or equal to the Lower Desired Temperature and less than or equal to the Upper Desired Temperature, the value of the Heat Control shall not be changed.

REQ-MHS-5: If the Regulator Mode is FAILED, the Heat Control shall be set to Off.
Rationale: If the Regulator Mode is FAILED, the Heat Control shall be set to Off.

A purple arrow points from the "Component contract" box to the "Heat Controller Task natural language functional requirements" box.

A small image of a developer at a computer is shown with the text "Developer formalizes requirements".

Contracts incorporated via SysMLv2 language construct (essentially, a comment that HAMR parses, highlights syntax, etc.)



HAMR - SysMLv2/AADL to Rust + seL4

Component Requirements to Contracts

Example: One contract from *heater control laws* in **Manage Heat Source** Thread (a periodic component), with traceability to natural language requirements.

Mode condition
(..if the mode is Normal)

...

```
case REQ_MHS_2 "If the Regulator Mode is NORMAL and the Current Temperature is less than
    the Lower Desired Temperature, the Heat Control shall be set to On.":
    assume (regulator_mode == Isolette_Data_Model::Regulator_Mode.Normal_Regulator_Mode)
    & (current_tempWstatus.value < lower_desired_temp.value);
    guarantee heat_control == Isolette_Data_Model::On_Off.On;
```

...

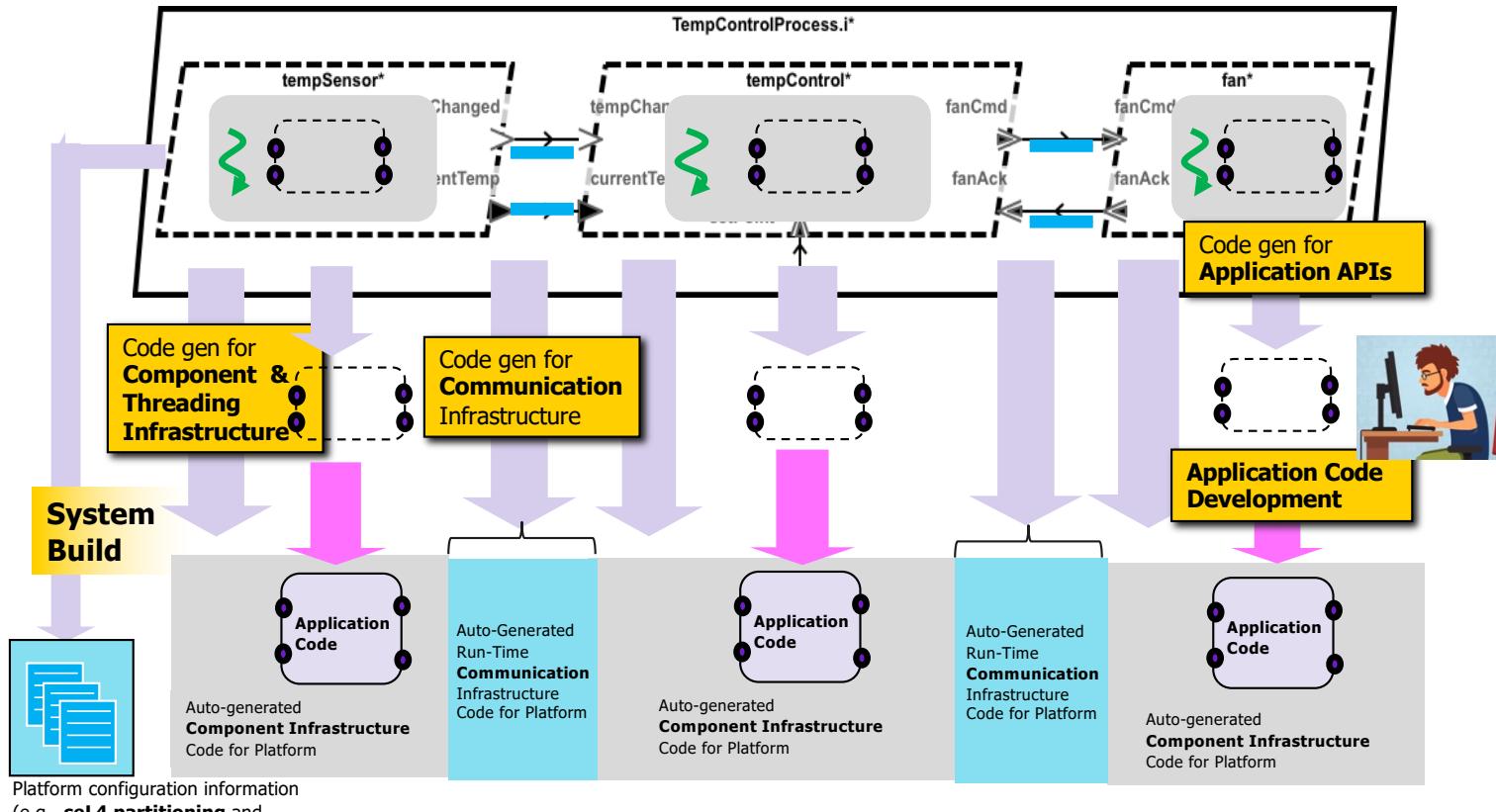
Compare current temperature to desired range
(..if temperature is below the target range)

Set the desired state of the heater
(...turn heater On, to warm up the Isolette)



OSATE AADL Editor

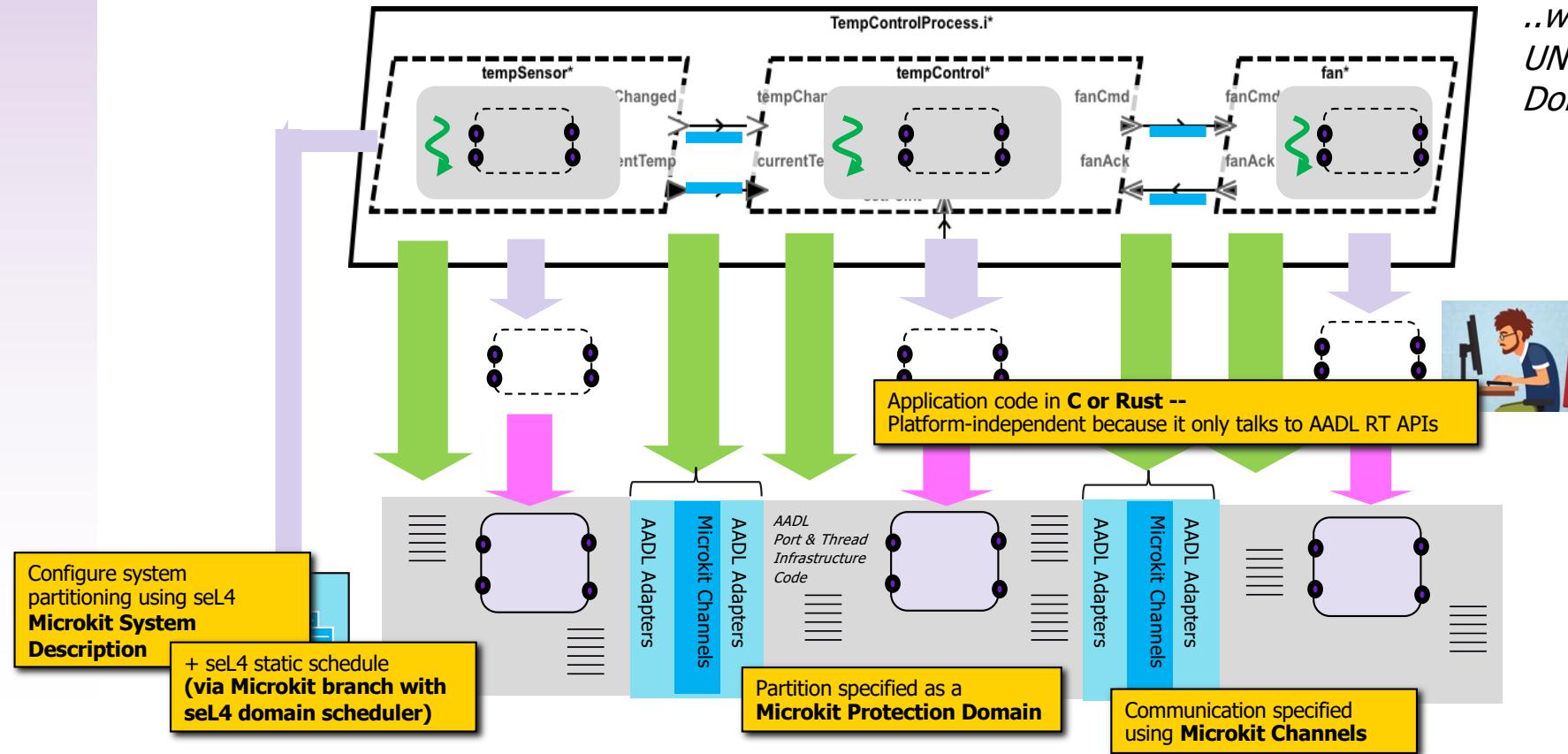
HAMR Code Generation



HAMR - SysMLv2/AADL to Rust + seL4

HAMR Code Generation

For seL4, the process is instantiated like this...



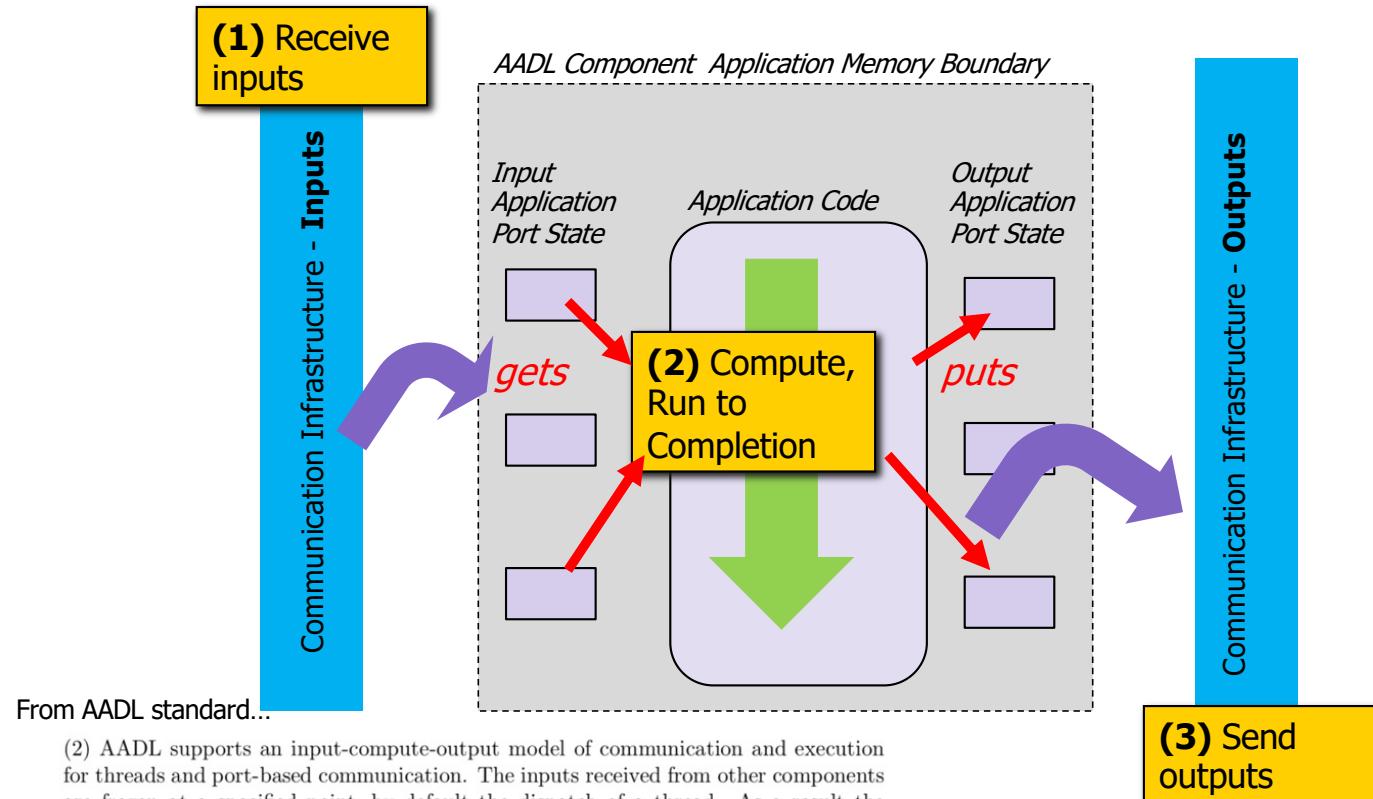
SysMLv2/AADL for HAMR with seL4 Microkit

AADL Port and Thread Execution Semantics



"Analyzable Real-Time Systems"
Burns & Wellings

On each dispatch, AADL threads follow a well-known **input-compute-output** pattern for real-time tasks that simplifies analysis and verification...



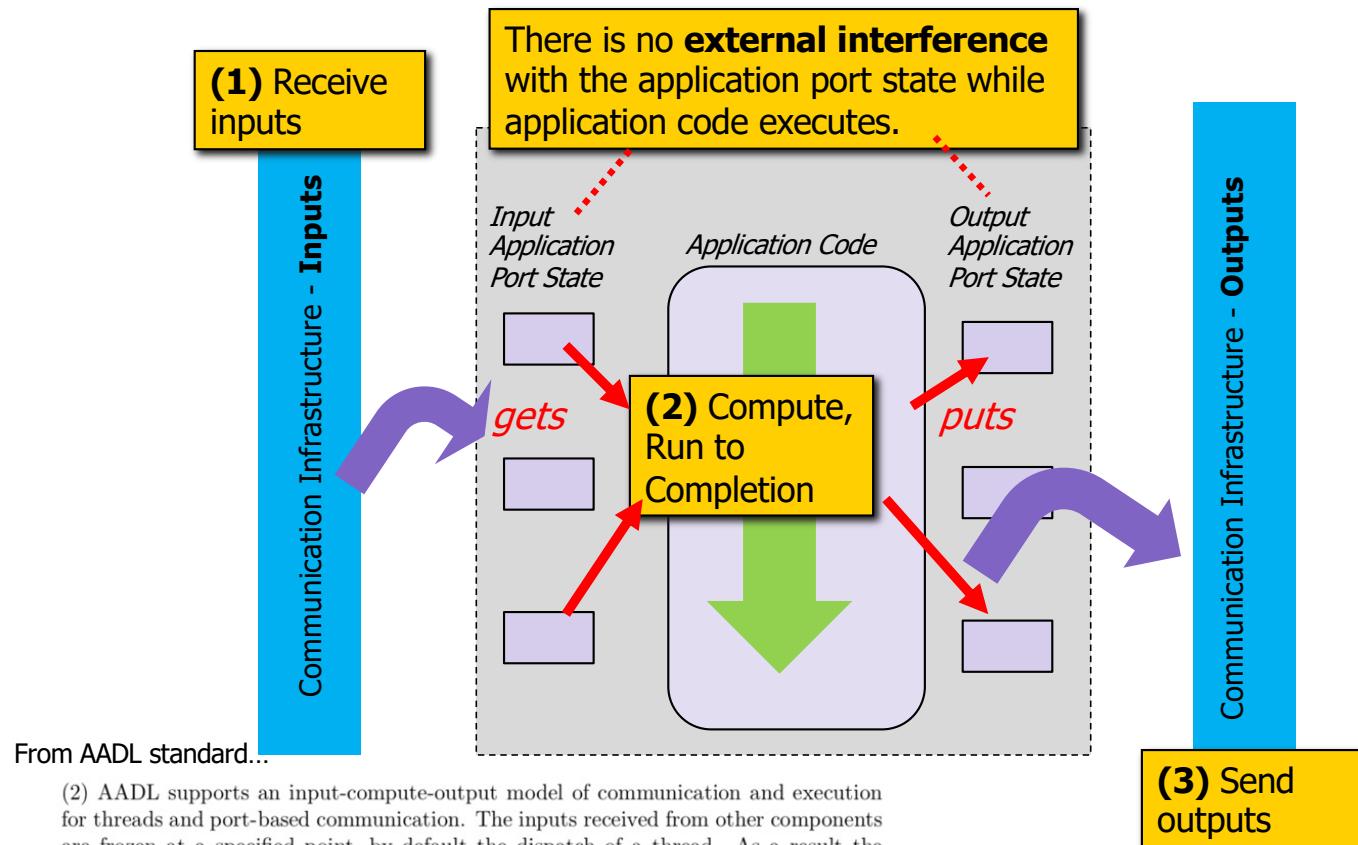
(2) AADL supports an input-compute-output model of communication and execution for threads and port-based communication. The inputs received from other components are frozen at a specified point, by default the dispatch of a thread. As a result the

AADL Port and Thread Execution Semantics



"Analyzable Real-Time Systems"
Burns & Wellings

AADL tasking and port semantics ensures no interference with other threads or communication layer

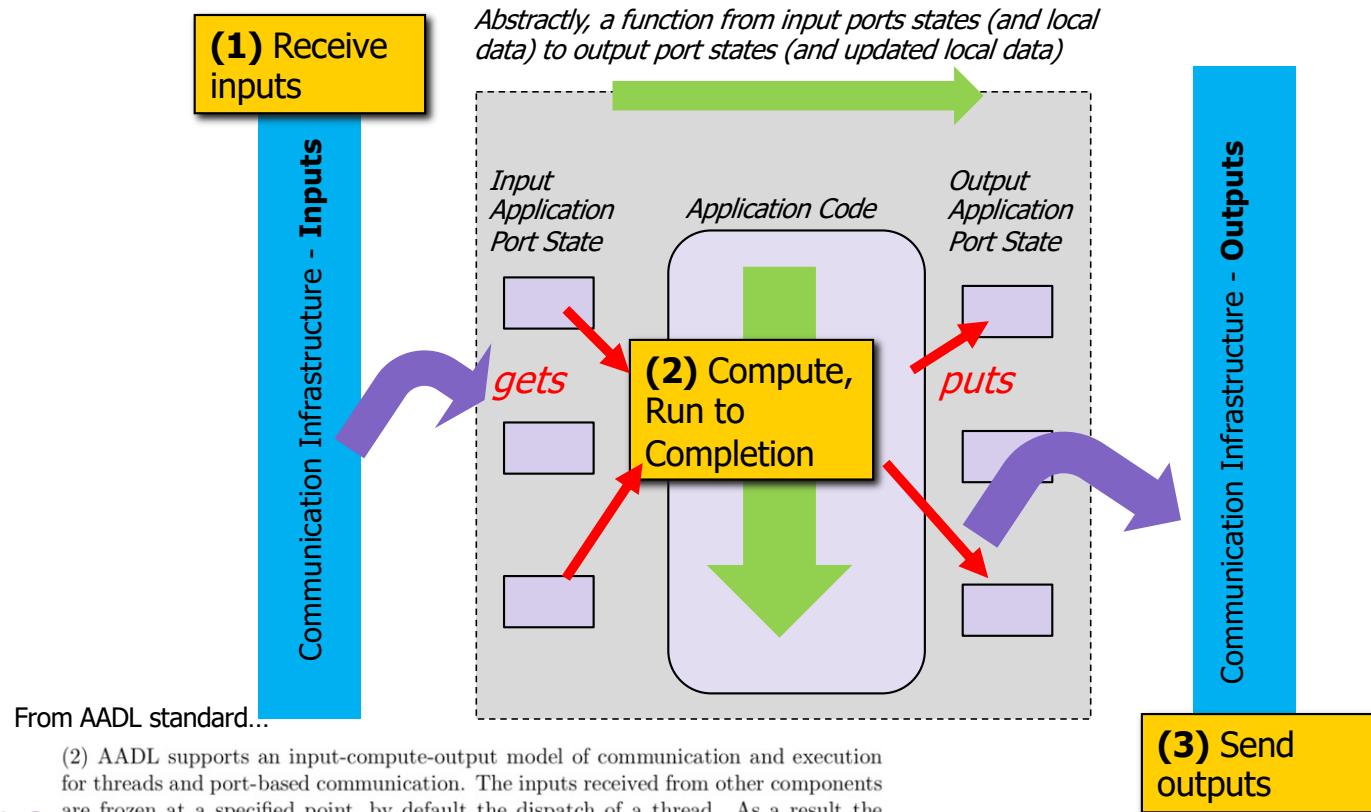


(2) AADL supports an input-compute-output model of communication and execution for threads and port-based communication. The inputs received from other components are frozen at a specified point, by default the dispatch of a thread. As a result the

AADL Port and Thread Execution Semantics



"Analyzable Real-Time Systems"
Burns & Wellings

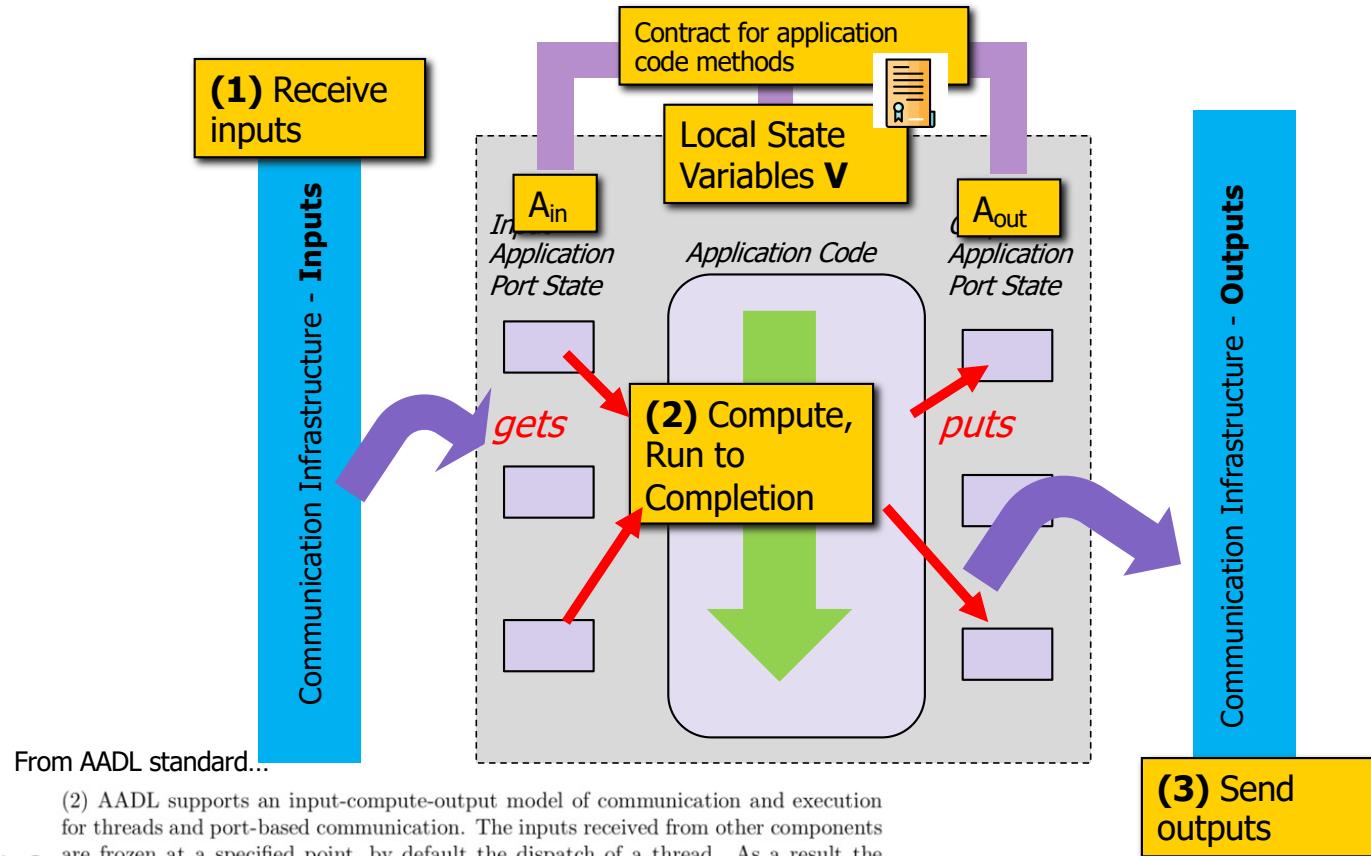


(2) AADL supports an input-compute-output model of communication and execution for threads and port-based communication. The inputs received from other components are frozen at a specified point, by default the dispatch of a thread. As a result the

AADL Port and Thread Execution Semantics

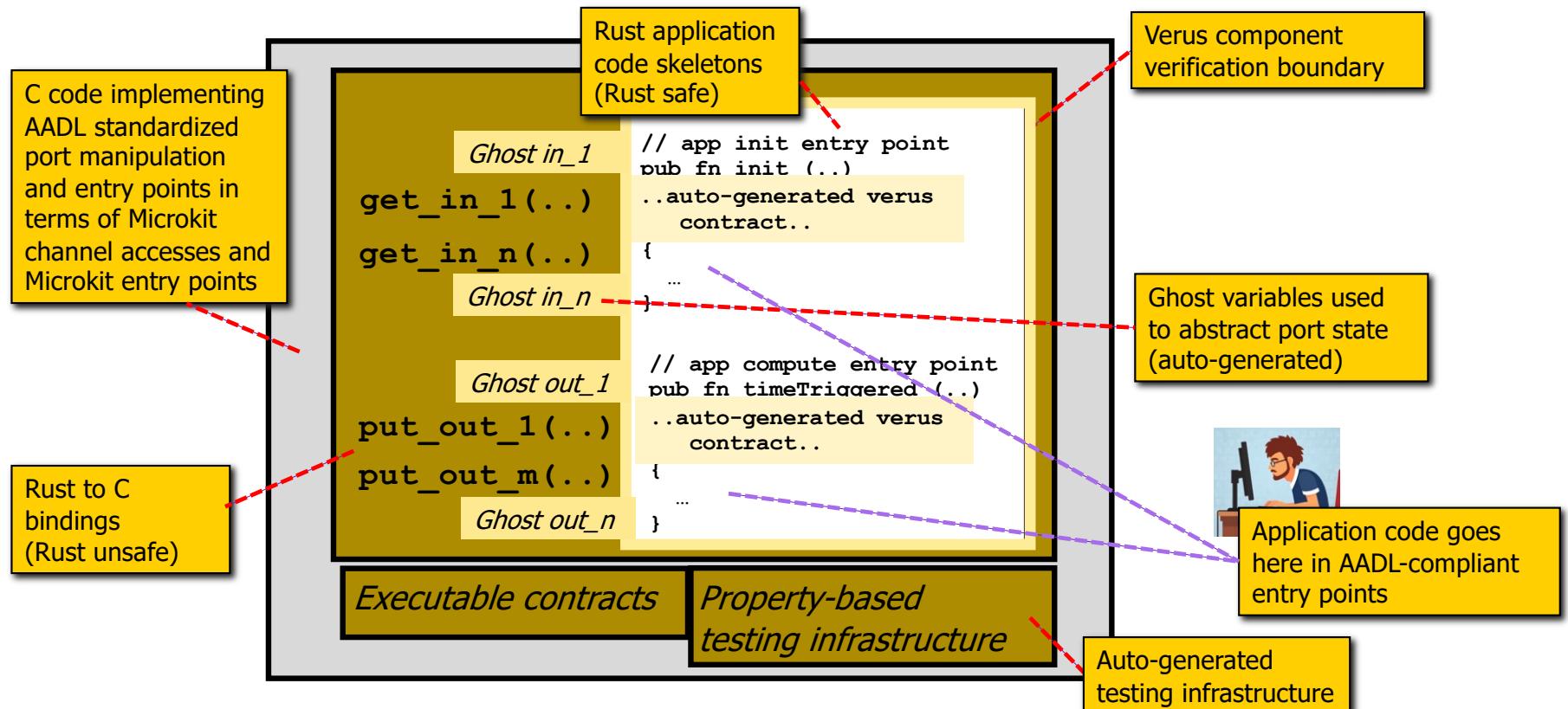


"Analyzable Real-Time Systems"
Burns & Wellings

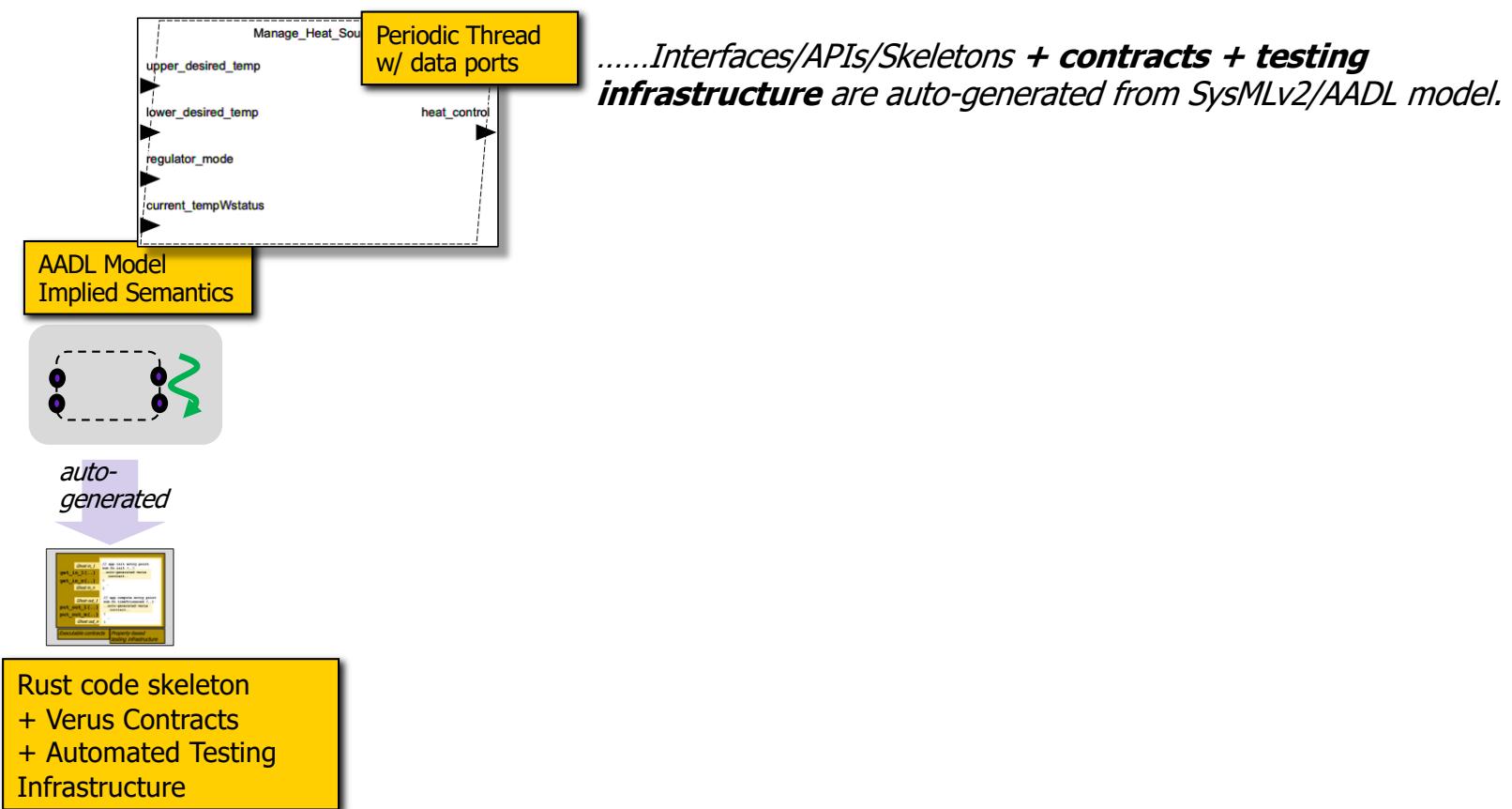


Outline of Protection Domain Structure

For each SysMLv2/AADL *periodic Thread* component, HAMR generates the following Microkit PD code...



Auto-generated Skeleton, Contracts, Testing



HAMR - SysMLv2/AADL to Rust + seL4

Auto-generated Skeleton, Contracts, Testing

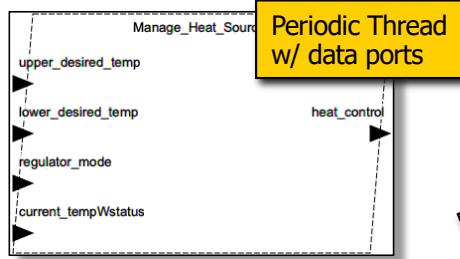
The diagram illustrates the workflow for generating Rust code from an AADL model. It starts with an **AADL Model** (Implied Semantics) which is converted into **Application Code in Rust**. This code is then integrated into a **Rust VSCode editor**.

Annotations highlight specific parts of the process:

- Periodic Thread w/ data ports**: Refers to the generated thread structure.
- Component contract (small excerpt)**: Shows a snippet of the generated component contract code.
- Skeleton for application code entry point**: Points to the generated entry point skeleton.
- Verus error indicates that contract is not yet satisfied**: Points to an error message in the VSCode editor.
- postcondition not satisfied**: Points to a specific error message in the VSCode editor's Problems panel.

Text overlay:*Interfaces/APIs/Skeletons + contracts + testing infrastructure* are auto-generated from SysMLv2/AADL model.

Verus Contract Auto-Generated From Model Contract



auto-generated



Verification of Rust application code against contracts using Verus (excerpts)

```
pub fn timeTriggered<API: thermostat_rt_mhs_mhs_Full_Api>(  
    &mut self,  
    api: &mut thermostat_rt_mhs_mhs_Application_Api<API>)  
requires  
    // BEGIN MARKER TIME TRIGGERED REQUIRES  
    // assume lower_is_lower_temp  
    old(api).lower_desired_temp.degrees <= old(api).upper_desired_temp.degrees  
    // END MARKER TIME TRIGGERED REQUIRES  
ensures  
    // BEGIN MARKER TIME TRIGGERED ENSURES  
    // guarantee lastCmd  
  
// case REQ_MHS_2  
//   If the Regulator Mode is NORMAL and the Current Temperature is less than  
//   the Lower Desired Temperature, the Heat Control shall be set to On.  
((old(api).regulator_mode == Regulator_Mode::Normal_Regulator_Mode) &&  
 | (old(api).current_tempWstatus.degrees < old(api).lower_desired_temp.degrees)) ==>  
 (api.heat_control == On_Off::Onn),  
  
//   If the Regulator Mode is NORMAL and the Current Temperature is less than  
//   the Lower Desired Temperature, the Heat Control shall be set to On.  
((old(api).regulator_mode == Regulator_Mode::Normal_Regulator_Mode) &&  
 | (old(api).current_tempWstatus.degrees < old(api).lower_desired_temp.degrees)) ==>  
 (api.heat_control == On_Off::Onn),
```

Coding and Background Verification

The diagram illustrates the development process for a real-time system. It starts with an **AADL Model Implied Semantics** (represented by a state transition diagram), which is **auto-generated** from an **Application Code in Rust** (shown in a code editor). The application code is annotated with **Periodic Thread w/ data ports**, indicating its execution context. The developer adds **application code to contract-annotated skeleton**, specifically implementing the `timeTriggered` method. The code editor shows the implementation of the `thermostat_rt_mhs_mhs` trait, which includes logic for regulator mode and heat control, along with comments for input port values, computation logic, and output port setting.

...Developer **adds application code** to contract-annotated skeleton, and **verification/testing tools** check conformance to contracts.

AADL Model Implied Semantics

auto-generated

Application Code in Rust

Periodic Thread w/ data ports

heat_control

impl thermostat_rt_mhs_mhs {
 pub fn timeTriggered<API: thermostat_rt_mhs_mhs_Full_Api>(
 old(api).regulator_mode == Isolette_Data_Model::Regulator_Mode::Failed_Regulation
 | (api.heat_control == Isolette_Data_Model::On_Off::Off)
 // END_MARKER_TIME_TRIGGERED_ENSURES

 // ----- Get values of input ports -----
 let lower: Temp_i = api.get_lower_desired_temp();
 let upper: Temp_i = api.get_upper_desired_temp();
 let regulator_mode: Regulator_Mode = api.get_regulator_mode();
 let currentTemp: TempWstatus_i = api.get_current_tempWstatus();

 //===== compute / control logic ======

 // current command defaults to value of last command (REQ-MHS-4)
 let mut currentCmd: On_Off = self.lastCmd;

 match regulator_mode { ...

 // ----- Set values of output ports -----
 api.put_heat_control(currentCmd);
 self.lastCmd = currentCmd
 } fn timeTriggered

HAMR - SysMLv2/AADL to Rust + SEL4

Coding and Background Verification

The diagram illustrates the development process for a thermostatic system, showing the relationship between the AADL model, generated code, and verification results.

AADL Model Implied Semantics: Shows a component interface with ports: `upper_desired_temp`, `lower_desired_temp`, `regulator_mode`, and `current_tempWstatus`. A red dashed circle highlights the `Get` operation on the `regulator_mode` port. Another red dashed circle highlights the `Put` operation on the `heat_control` port.

Application Code in Rust: A screenshot of a code editor showing the implementation of the `thermostat_rt_mhs_mhs` component. The code uses auto-generated APIs to handle input ports and control logic.

- Periodic Thread w/ data ports:** An annotation pointing to the `heat_control` port in the AADL model.
- ...Developer uses auto-generate APIs to **get** and **put** data on component ports:** An annotation pointing to the `Get` and `Put` operations in the AADL model.
- Reading a value from the `regulator_mode` input data port using auto-generated API:** An annotation pointing to the `api.get_regulator_mode()` call in the Rust code.
- Putting a value from the `heat_control` output data port using auto-generated API:** An annotation pointing to the `api.put_heat_control(currentCmd)` call in the Rust code.
- Verus indicates that contract is satisfied:** An annotation pointing to the Verus analysis results at the bottom left.
- No problems have been detected in the workspace:** An annotation pointing to the Verus analysis results at the bottom right.

Bottom Navigation: Includes links for `verus-analyzer` and `verus-analyzer`, and status information: Jason Belt (1 month ago), Ln 114, Col 1, Spaces: 2, UTF-8, LF, {&} Rust.

Demo

Verification of application code against contracts using Verus verification tool...

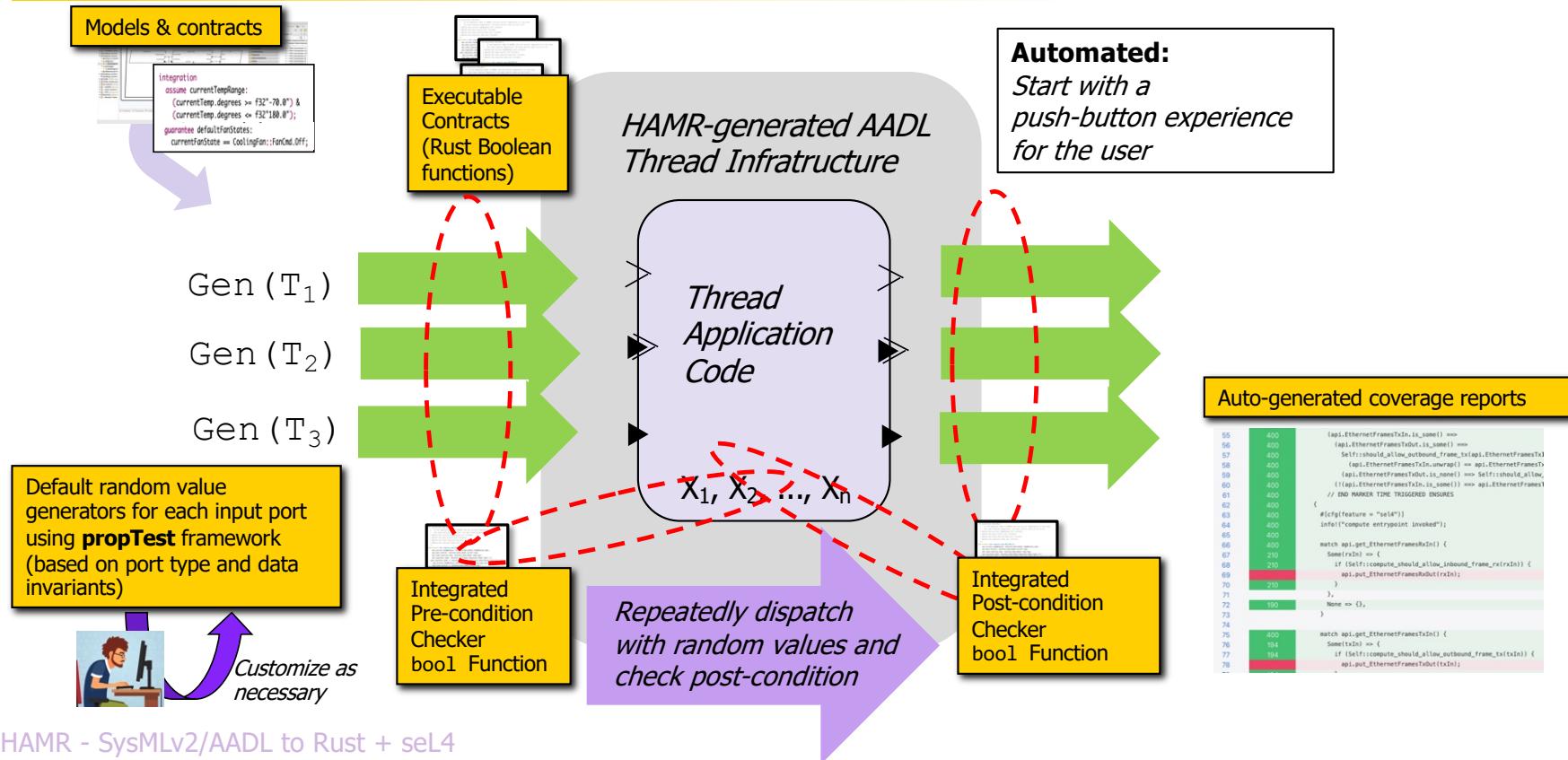
The screenshot shows a VSCode interface with the following details:

- File Path:** thermostat_rt_mhs_mhs
- Code Editor Content:** A Rust file named `thermostat_rt_mhs_mhs_app.rs`. The code implements a component for managing heat sources in a thermostat. It includes several match statements based on regulator modes (INIT, NORMAL, FAILED) and temperature status (lower desired temp, upper desired temp). The code is annotated with Verus contracts.
- Annotations:** A large red callout box highlights a specific contract violation:

```
Verus automatically detects  
a violation of the contract  
clause for the MHS-2  
requirement
```
- Tool Status Bar:** Shows "HAIIR - SysMLV2/AADL to Rust + SEL4".
- Bottom Status Bar:** Includes icons for main, Git Graph, rust-analyzer, verus-analyzer, and file status (Not Committed Yet).

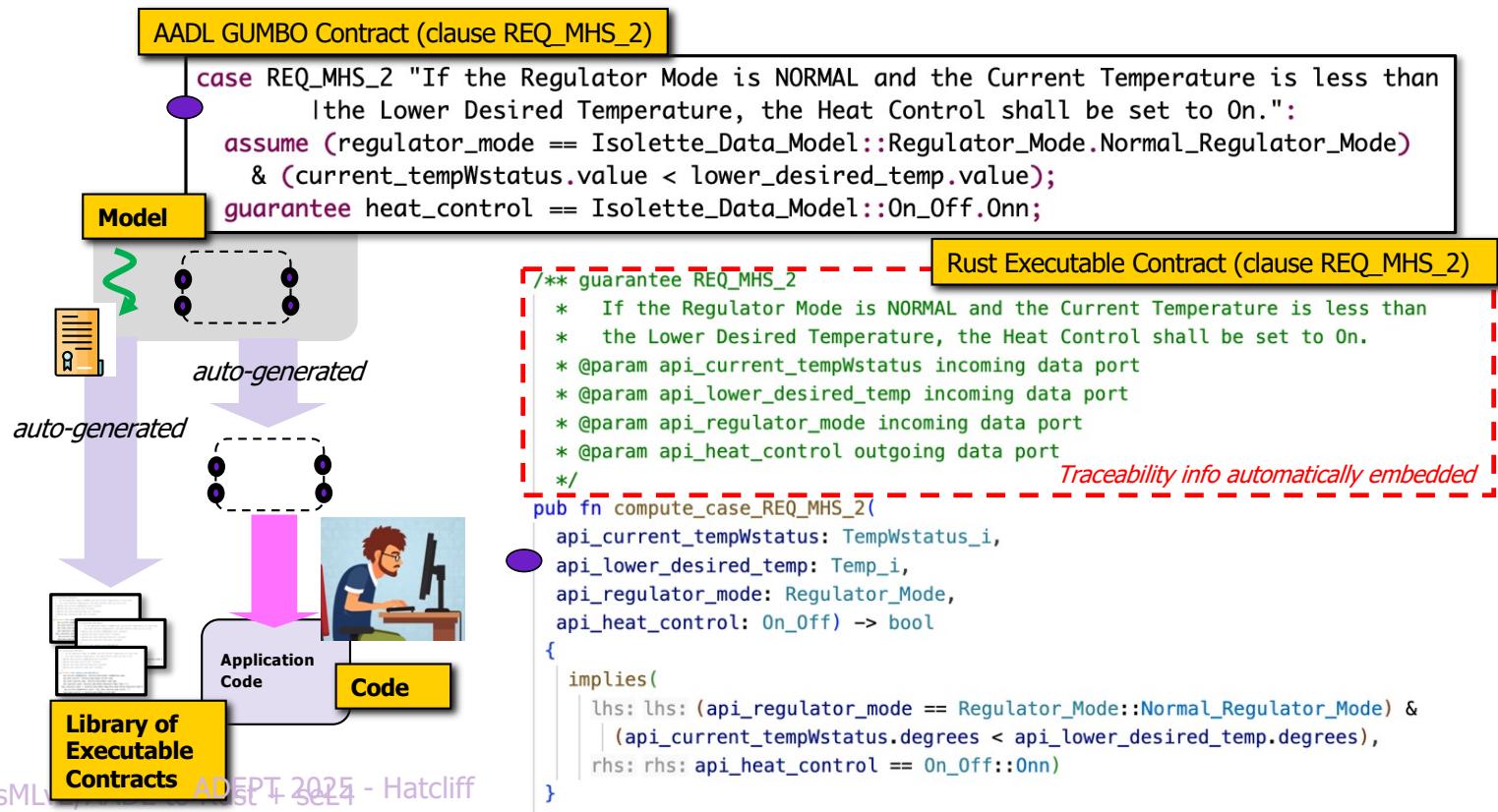
Automated Testing to Contracts

For every thread component, HAMR auto-generates property-based testing infrastructure for inserting values into component input ports and for checking values of output ports.



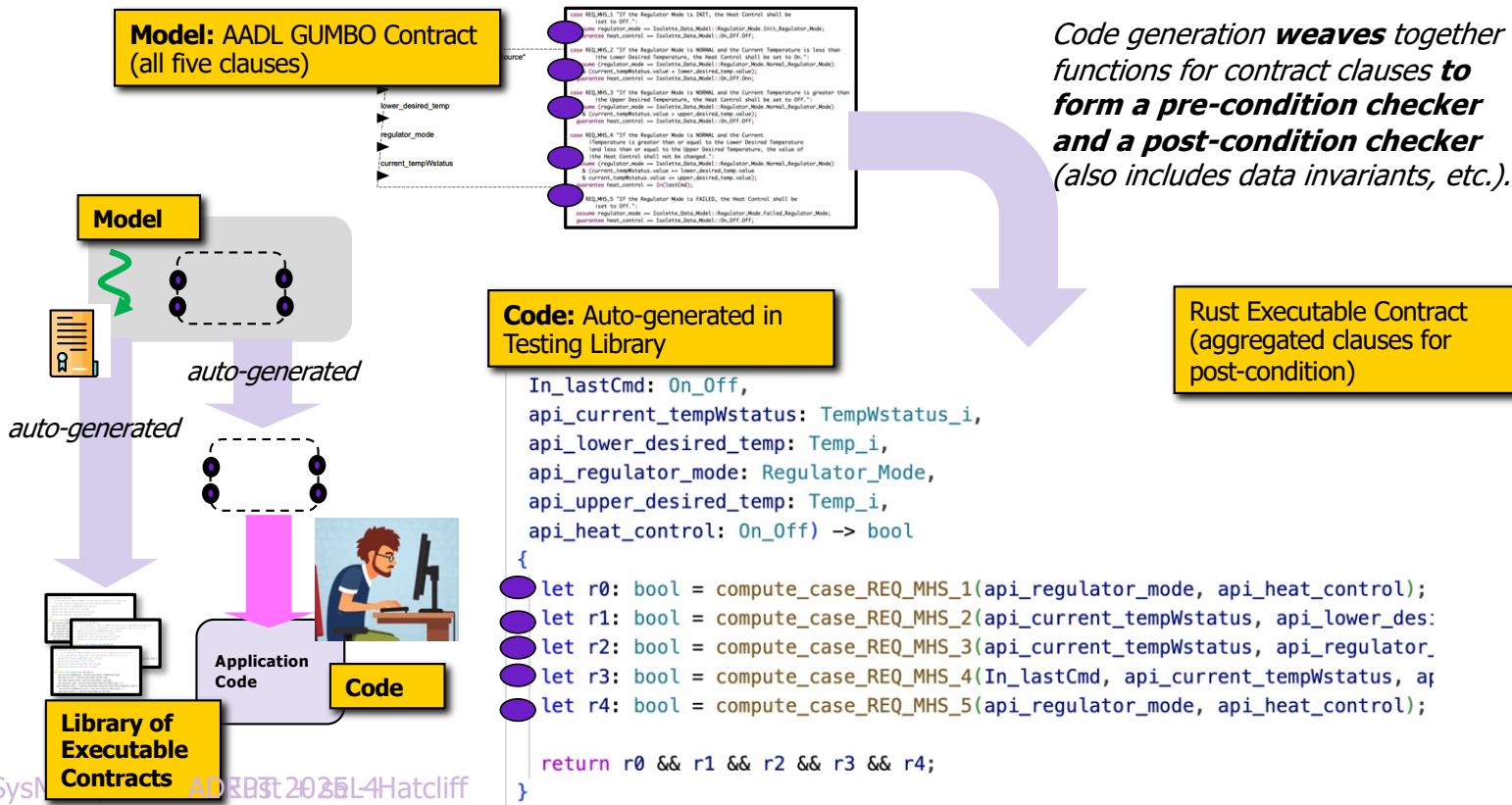
HAMR-generated *Executable Contracts*

Each clause in **model-level** GUMBO contracts is translated to a **code-level** Boolean function in Rust that works on the appropriate port/thread state elements



HAMR-generated *Executable Contracts*

Complete set of **Model-level** GUMBO contract clauses are translated to a hierarchy of executable Boolean functions in Rust (**code-level**) to form executable pre/post conditions and test oracles.



HAMR-generated Randomizing Test Runner

HAMR automatically generates test runner infrastructure with default random value generators for each input port. The executable contract is automatically used behind the scenes as a test oracle.

```
testComputeCB_macro! {  
    ▶ Run Test | Debug  
    prop_testComputeCB_default, // test name  
    config: ProptestConfig { // proptest configuration, built by overriding fields from default config  
        cases: numValidComputeTestCases,  
        max_global_rejects: numValidComputeTestCases * computeRejectRatio,  
        verbose: verbosity,  
        ..ProptestConfig::default()  
    },  
    // auto-generated strategies for generating each component input  
    api_current_tempWstatus: test_api::Isolette_Data_Model_TempWstatus_i_strategy_default(),  
    api_lower_desired_temp: test_api::Isolette_Data_Model_Temp_i_strategy_default(),  
    api_regulator_mode: test_api::Isolette_Data_Model_Regulator_Mode_strategy_default(),  
    api_upper_desired_temp: test_api::Isolette_Data_Model_Temp_i_strategy_default()  
}
```

Press this button and you automatically get 1000's of random tests against the component contract

auto-generated configurations for propTest framework

auto-generated propTest random value generators for each input port



HAMR-generated Randomizing Test Runner

Default random generators are often easy to customize to increase coverage, reduce #'s of discarded tests, obtain tests for specific features, etc.

```
testComputeCB_macro! {  
    ▶ Run Test | Debug  
    prop_testComputeCB_default, // test name  
    config: ProptestConfig { // proptest configuration, built by overriding fields from default config  
        cases: numValidComputeTestCases,  
        max_global_rejects: numValidComputeTestCases * computeRejectRatio,  
        verbose: verbosity,  
        ..ProptestConfig::default()  
    },  
}
```

Customizing a numeric generator to a particular range

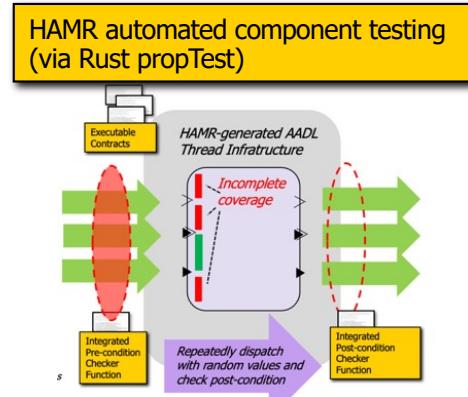
```
// developer-customized strategies for generating each component input  
api_current_tempWstatus: test_api::Isolette_Data_Model_TempWstatus_i_strategy_cust(  
    95..=103,  
    test_api::Isolette_Data_Model_ValueStatus_strategy_default()),  
api_lower_desired_temp: test_api::Isolette_Data_Model_Temp_i_strategy_cust(94..=105),  
api_regulator_mode: test_api::Isolette_Data_Model_Regulator_Mode_strategy_default(),  
api_upper_desired_temp: test_api::Isolette_Data_Model_Temp_i_strategy_cust(94..=105)
```



Developer-customized generators

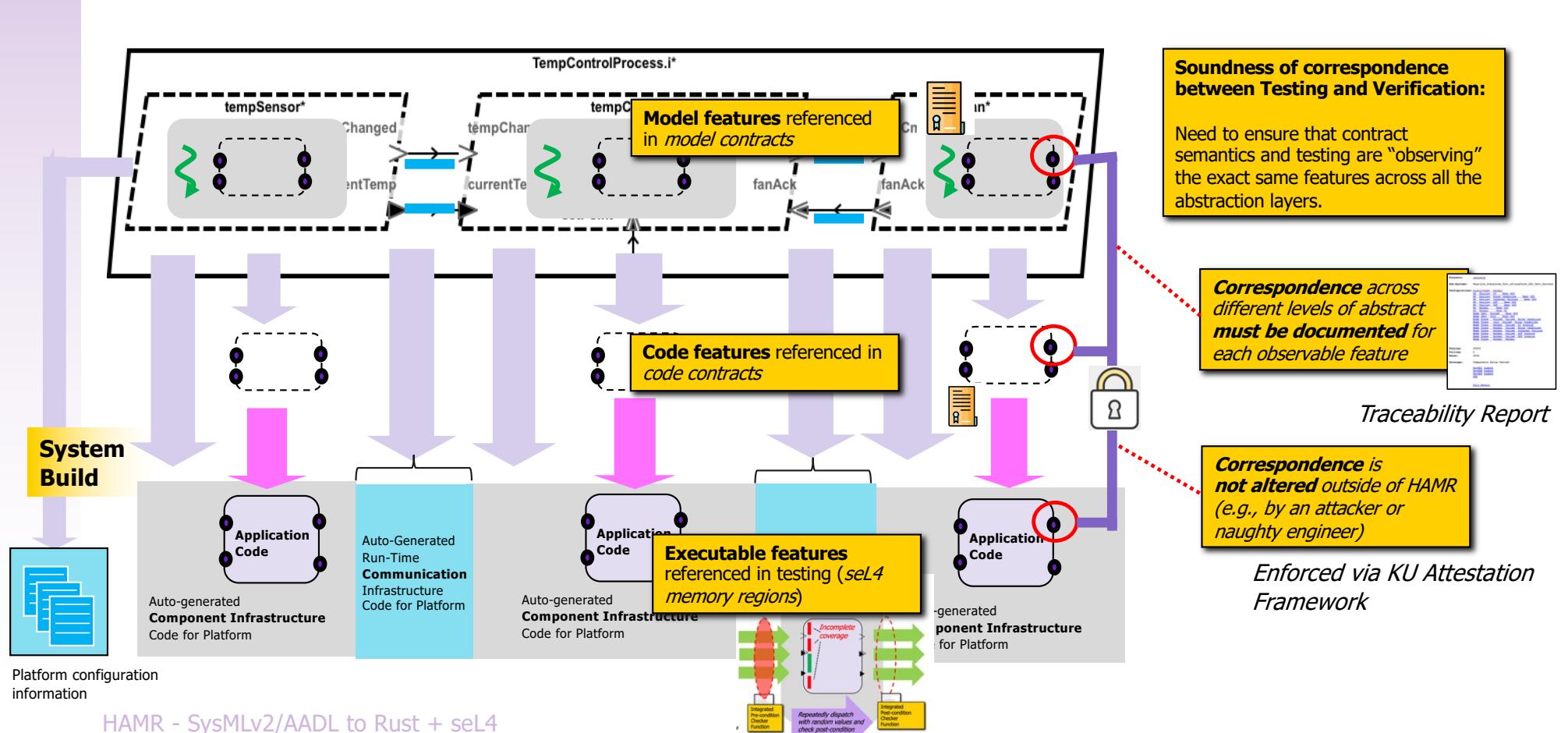
Benefits - Integrated Testing / Verification

- Immediately launch 1000's of default tests, check conformance to contracts
 - Debug contracts; gradually move to Verus
 - When Verus verification fails, generate concrete failing tests that can be given to developers to run through debugger
 - When Verus/SMT cannot handle certain language features; use testing for lower-confidence assurance
 - Maybe be a step taken before handing off certain VCs from Verus to Lean
 - Testing and verification *derived from the exact same GUMBO contracts*



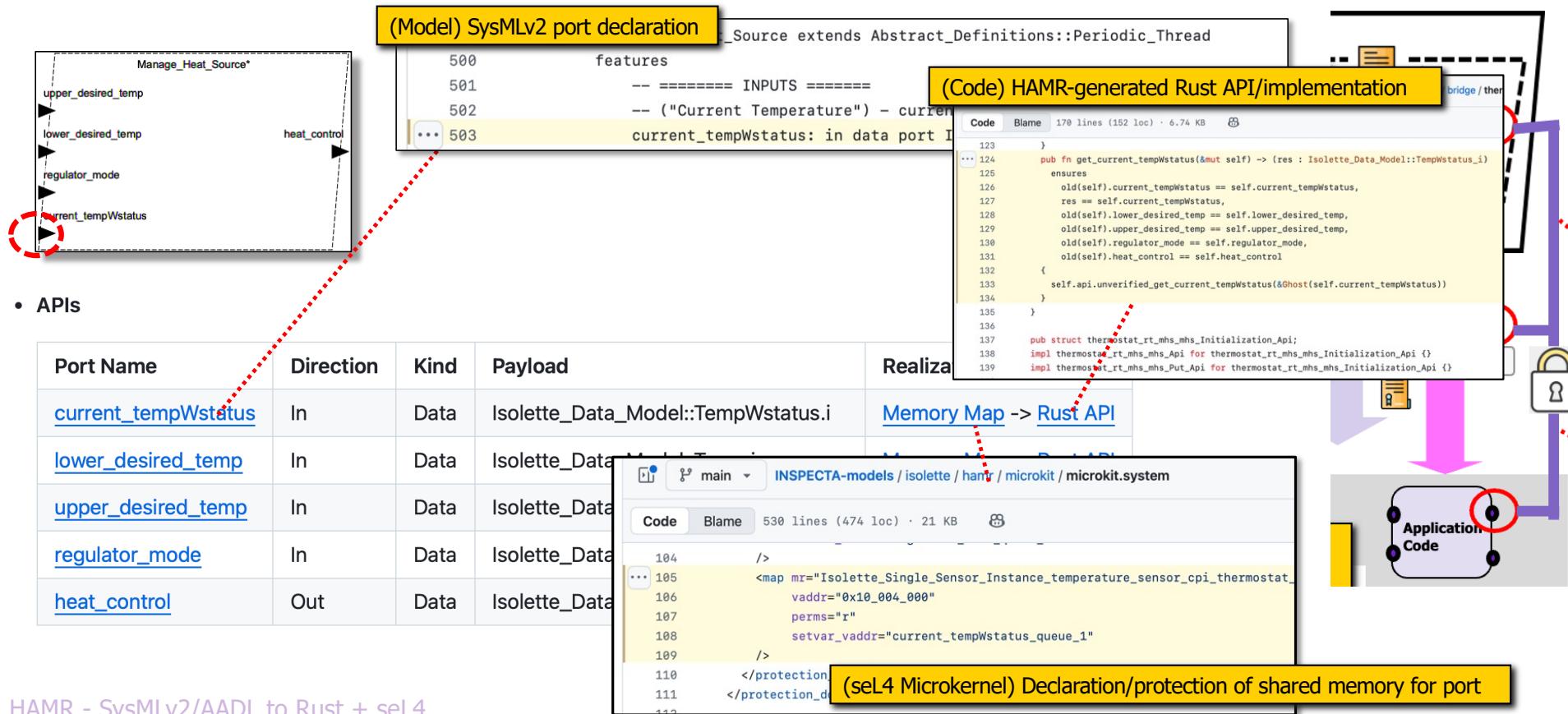
HAMR - SysMLv2/AADL to Rust + seL4

HAMR Observations/Traceability Framework



Auto-generated System Feature Traceability for Manage Heat Source Port (GitHub Markdown)

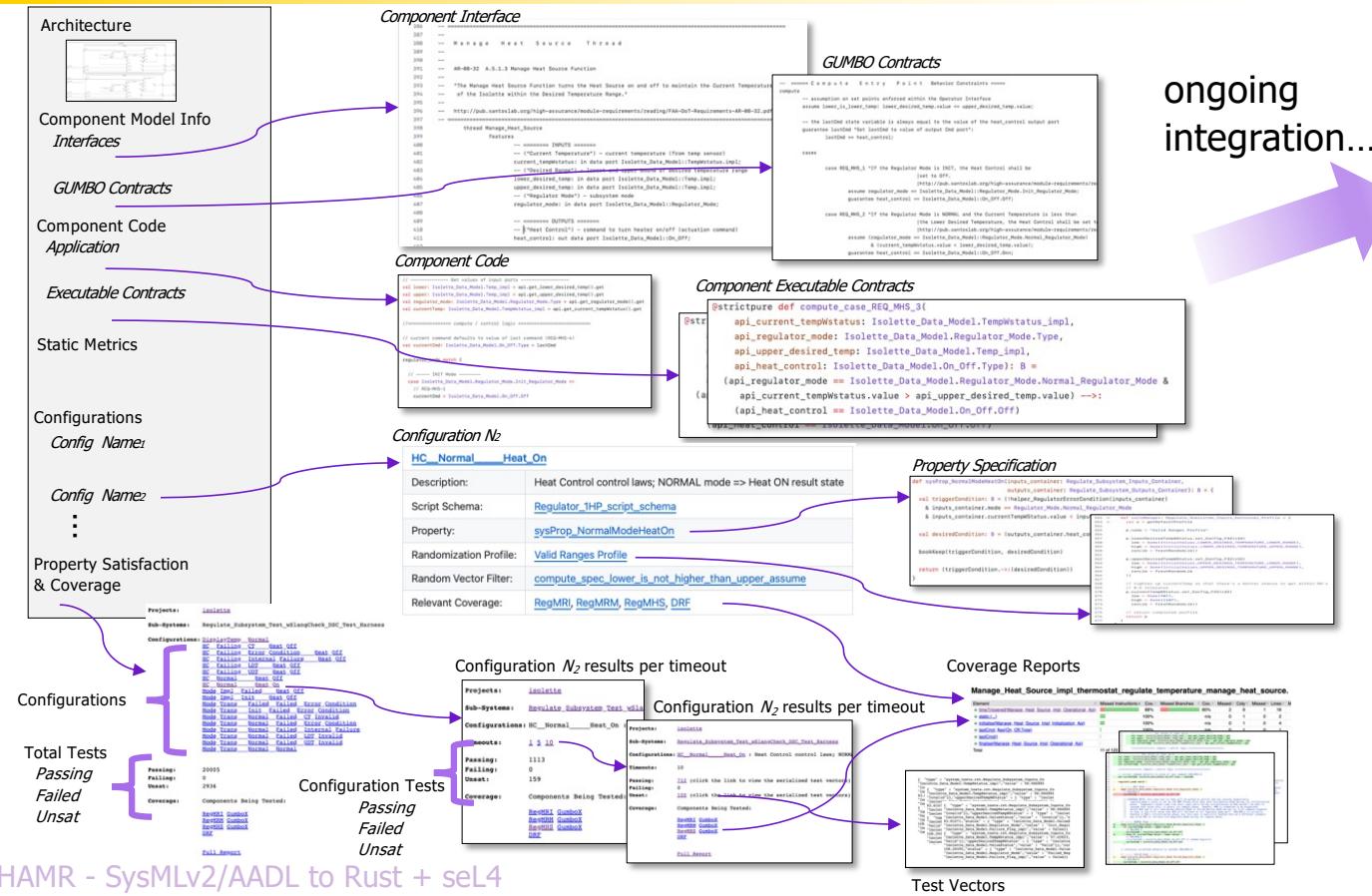
HAMR auto-generates traceability reports, e.g., for a port – relationships between **model**, **code**, **kernel** artifacts



HAMR - SysMLv2/AADL to Rust + seL4

Assurance and Traceability Reports

HAMR auto-generates a variety of assurance and traceability reports

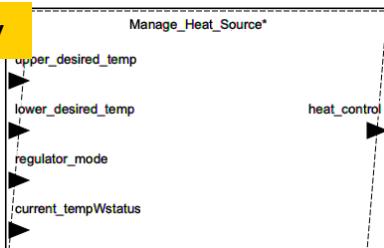


Auto-generated Contract Traceability for Manage Heat Source Requirement (GitHub Markdown)

Contract Clause traceability

GUMBO

State Variables		
lastCmd	GUMBO	Verus



Initialize

guarantee initlastCmd	GUMBO	Verus	GUMBOX
guarantee REQ_MHS_1	GUMBO	Verus	GUMBOX

Compute

assume lower_is_lower_temp	GUMBO	Verus	GUMBOX
guarantee lastCmd	GUMBO	Verus	GUMBOX
case REQ_MHS_1	GUMBO	Verus	GUMBOX
case REQ_MHS_2	GUMBO	Verus	GUMBOX
case REQ_MHS_3	GUMBO	Verus	GUMBOX
case REQ_MHS_4	GUMBO	Verus	GUMBOX
case REQ_MHS_5	GUMBO	Verus	GUMBOX

Req ID

(Model) GUMBO contract clause for the requirement

```

Code Blame 623 lines (535 loc) · 31.7 KB
739
548 case REQ_MHS_1 "If the Regulator Mode is INIT, the Heat Control shall be
549     set to Off.
550     [http://pub.santoslab.org/high-assurance/module-requirements/reading/FAA-Dot-Requirements-AR-88-32.pdf#page=118 :]
551     assume regulator_mode == Isolette_Data_Model::Regulator_Mode.Init_Regulator_Mode;
552     guarantee heat_control == Isolette_Data_Model::On_Off.Off;
553
554 case REQ_MHS_2 "If the Regulator Mode is NORMAL and the Current Temperature is less than
555     the Lower Desired Temperature, the Heat Control shall be set to On.
556     [http://pub.santoslab.org/high-assurance/module-requirements/reading/FAA-Dot-Requirements-AR-88-32.pdf#page=118 :]
557     assume (regulator_mode == Isolette_Data_Model::Regulator_Mode.Normal_Regulator_Mode)
558     & (current_temperature_degrees < lower_desired_temp_degrees);
559     guarantee heat_control == Isolette_Data_Model::On_Off.On;
560
561 case REQ_MHS_3 "If the Regulator Mode is NORMAL and the Current Temperature is greater than
562     the Upper Desired Temperature, the Heat Control shall be set to Off.
563     [http://pub.santoslab.org/high-assurance/module-requirements/reading/FAA-Dot-Requirements-AR-88-32.pdf#page=118 :]
564     assume (regulator_mode == Isolette_Data_Model::Regulator_Mode.Normal_Regulator_Mode)
565
566 ...
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
837
838
839
839
840
841
842
843
844
845
845
846
847
847
848
849
849
850
851
852
853
854
855
856
856
857
858
858
859
859
860
861
862
863
864
865
865
866
867
867
868
868
869
869
870
871
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
881
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1630
1631
1631
1632
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1640
1641
1641
1642
1642
1643
1643
1644
1644
1645
1645
1646
1646
1647
1647
1648
1648
1649
1649
1650
1650
1651
1651
1652
1652
1653
1653
1654
1654
1655
1655
1656
1656
1657
1657
1658
1658
1659
1659
1660
1660
1661
1661
1662
1662
1663
1663
1664
1664
1665
1665
1666
1666
1667
1667
1668
1668
1669
1669
1670
1670
1671
1671
1672
1672
1673
1673
1674
1674
1675
1675
1676
1676
1677
1677
1678
1678
1679
1679
1680
1680
1681
1681
1682
1682
1683
1683
1684
1684
1685
1685
1686
1686
1687
1687
1688
1688
1689
1689
1690
1690
1691
1691
1692
1692
1693
1693
1694
1694
1695
1695
1696
1696
1697
1697
1698
1698
1699
16
```

Conclusions -- Themes

- Raising the abstraction level
 - “**patterns** for Microkit protection domains” **for application components**
 - choosing patterns to be amenable to component/system assurance
 - representing patterns/abstractions in **standardized modeling languages**
 - separating developer view of the pattern (higher-level) from seL4/microkit realization (lower-level)
- Auto-generation support for development tasks
 - build scripts, VM configuration, testing, logging
- Leveraging specifications for both verification and testing
- Integrating activities from broader industry ecosystems, especially those related to assurance activities

Plans - Next Six Months

- System Reasoning
 - Formal system specifications
 - System Testing / Run-time Monitoring
 - System Verification
- Continued evolution of Microkit target
 - More systematic scheduling and communication, support for LionsOS concepts
- Efficiency improvements for seL4 microkit and hardening of infrastructure code
- Build-out for assurance framework, traceability, attestation