Expanding AADL Code Generation & Formal Methods Tooling to SysMLv2

TCCOE 2024 – May 10, 2024

Kansas State University

John Hatcliff
Robby
Jason Belt

Collins Aerospace DARPA PROVERS INSPECTA

- *Aarhus University*
- *CMU*
- *ProofCraft*
- *UNSW*

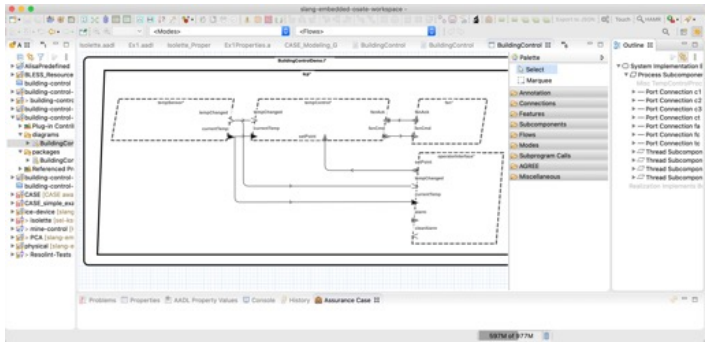
Galois

- *Todd Carpenter, Danielle Stewart*

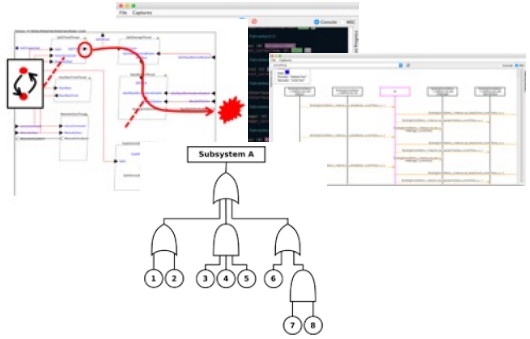
HAMR

HAMR – tool chain for [H]igh [A]ssurance [M]odeling and [R]apid engineering for embedded systems (developed by Kansas State University and Galois)

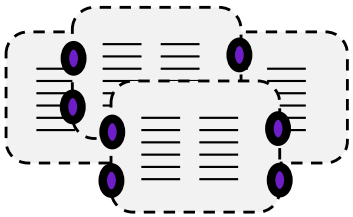
Modeling, analysis, and verification in the **AADL** modeling language



Leveraging analyses from AADL community



Component development and verification in multiple languages



- C
- Slang (developed at Kansas State)
  - high integrity subset of Scala
  - contract verification framework
  - translates to C

Deployments aligned with AADL run-time on multiple platforms

JVM Deployment

Linux Deployment

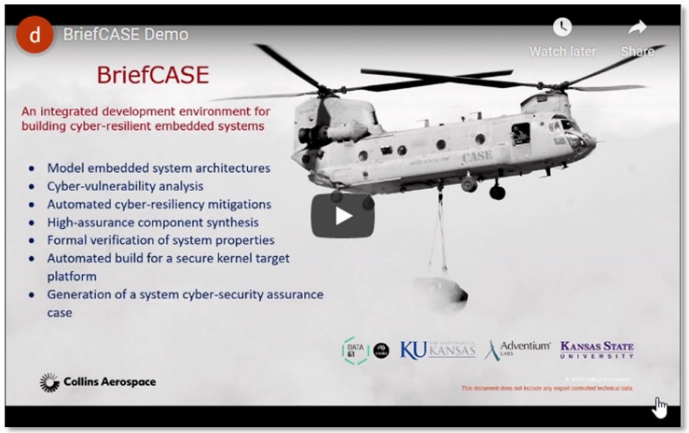
seL4 Deployment

verified micro-kernel

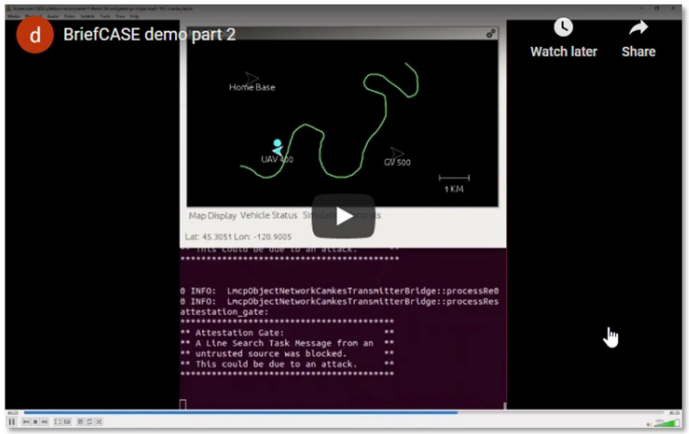
HAMR with seL4 on DARPA CASE

Collins Aerospace CASE project web site – includes videos of the Phase II end-to-end demonstration

http://loonwerks.com/projects/case.html



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).

Detailed journal paper on HAMR with seL4 backend applied on DARPA CASE

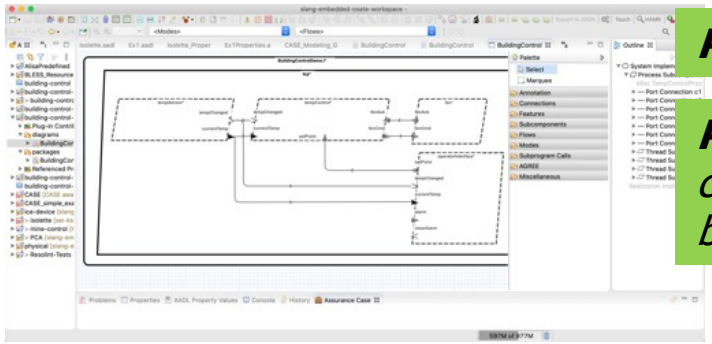
Jason Belt, John Hatcliff, Robby, John Shackleton, Jim Carciofini, Todd Carpenter, Eric Mercer, Isaac Amundson, Junaid Babar, Darren Cofer, David Hardin, Karl Hoech, Konrad Slind, Ihor Kuz, Kent Mcleod. "**Model-Driven Development for the seL4 Microkernel Using the HAMR Framework**". Journal of Systems Architecture. Volume 134, January 2023

http://people.cs.ksu.edu/~hatcliff/Papers/Belt-etal-JSA-2022-HAMR-sel4.pdf

HAMR on DARPA PROVERS

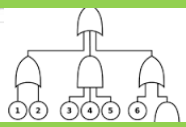
HAMR – tool chain for [H]igh [A]ssurance [M]odeling and [R]apid engineering for embedded systems (developed by Kansas State University and Galois)

Modeling, analysis, and verification in the **AADL** modeling language

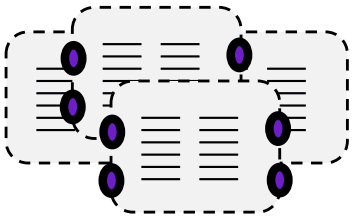


***PROVERS**: Add SysMLv2 prototype*

***PROVERS**: Enhanced support for contracts, verification, property-based testing*



Component development and verification in multiple languages



- C
- Slang
  -
  - contract verification framework
  - translates to C

***PROVERS**: Add code- and contract-generation, and property-based testing for Rust*

Deployments aligned with AADL run-time on multiple platforms

JVM Deployment

JVM (Java Virtual Machine)

Linux Deployment

Linux

seL4 Deployment

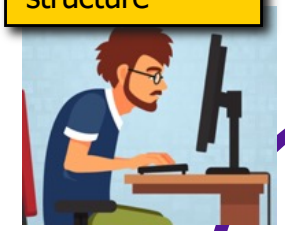
seL4 Security. Performance. Proof.

verified micro-kernel

***PROVERS**: Retarget to seL4 micro-kit (Core Platform)*

AADL Modeling Concepts

Developer configures computational structure



AADL Thread Property Options

Periodic
Sporadic
Hybrid
…

+ timing, scheduling constraints, etc.

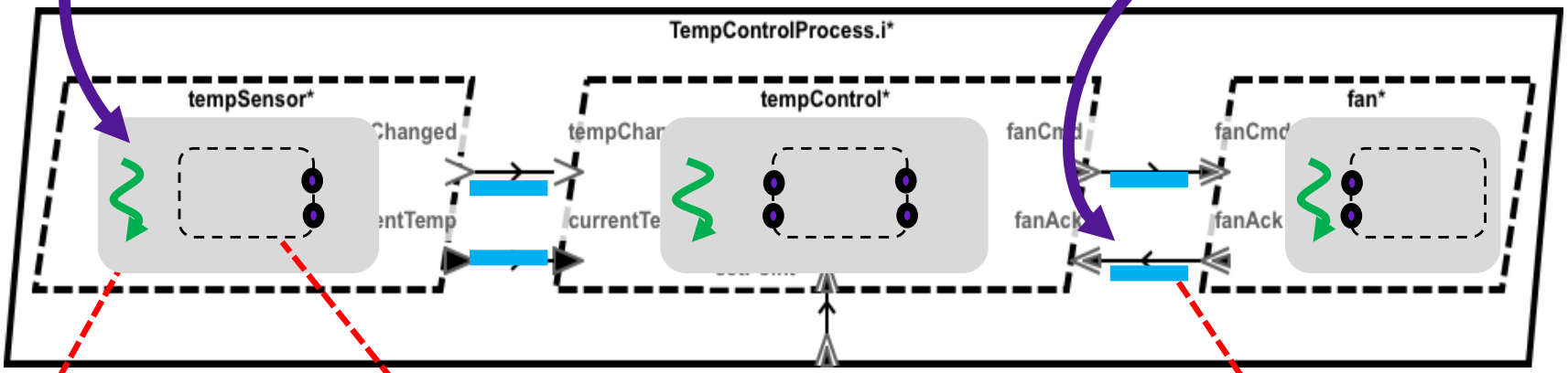
AADL Port & Connection Property Options

Event Data

Event Data
…

buffered notifications
shared data cells
(or data distribution service)
buffered messages
(message passing middleware)

+ QoS, buffer sizes, latencies, etc



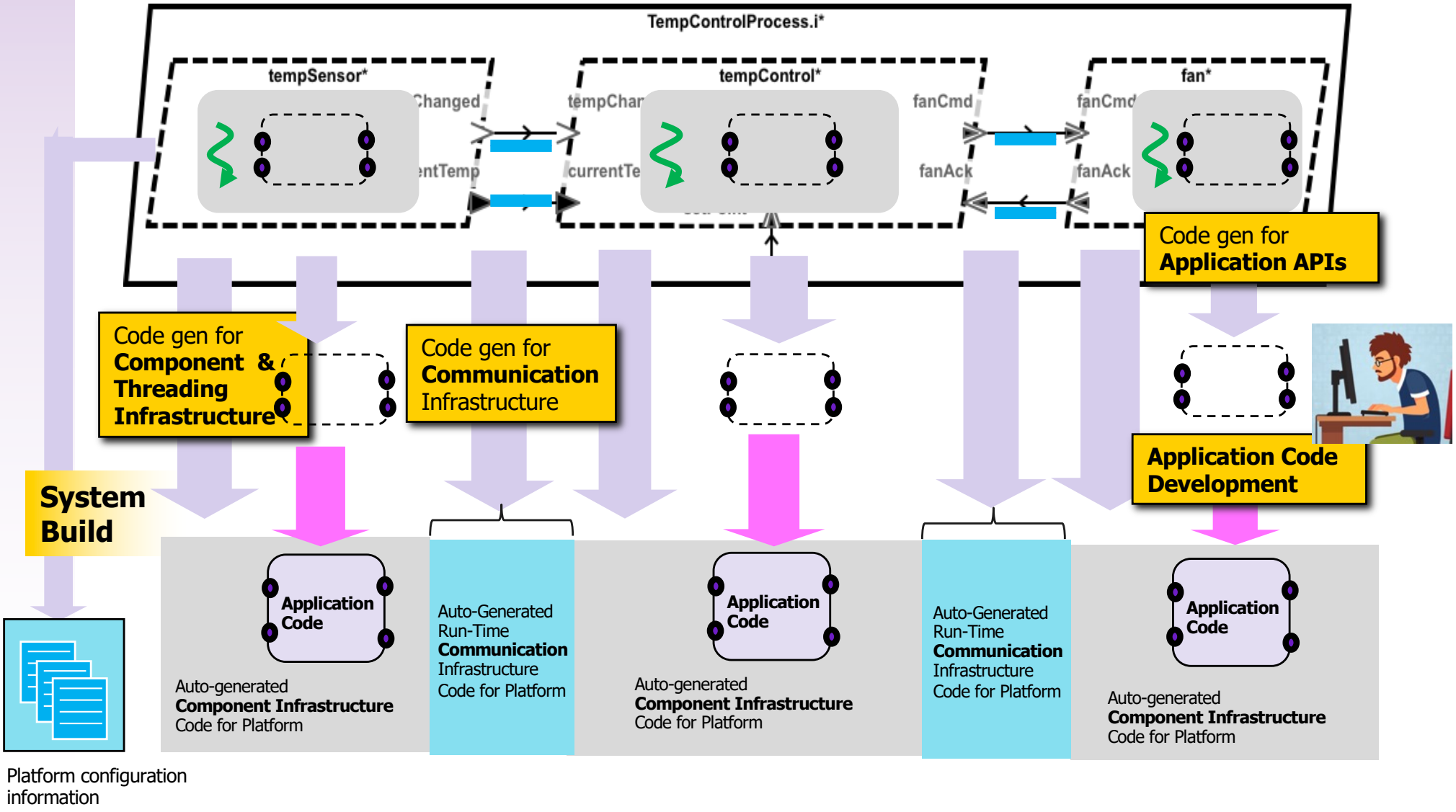
Code skeleton for selected thread pattern

Implied API Pattern for application code to access port communication, etc.

Implementation of selected communication pattern

…an AADL contract language and testing infrastructure should be aligned with these patterns.

HAMR Code Generation



Strengths / Weaknesses of AADL

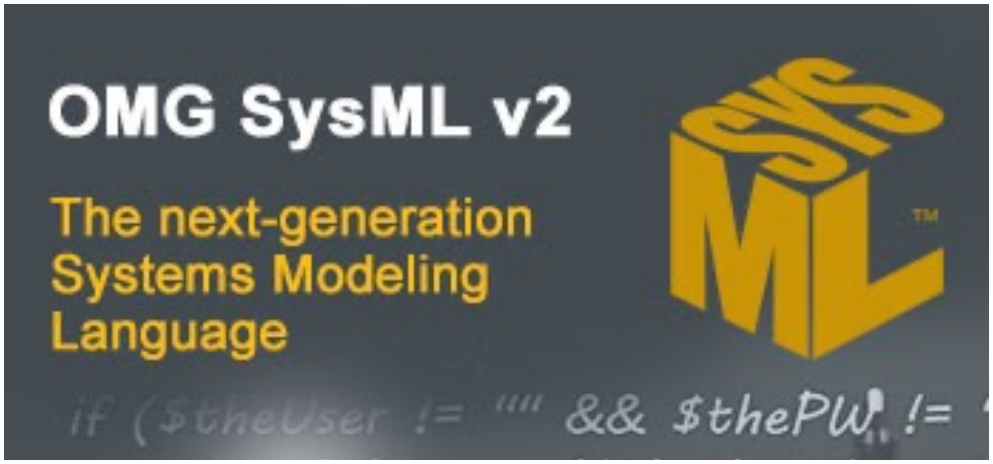
Strengths

- SAE Standard
- Strong pedigree of semantically solid real-time embedded systems (RTES) modeling concepts (originally derived from Honeywell – MetaH)
- Open source AADL editor (OSATE from SEI)
- AADL Annex concept supported extensions to core language to support a variety of analysis annotations, formal specifications, and tool plug-ins
- Used in a number of formal-methods oriented DoD/European research projects

Weaknesses inhibiting goal of formal methods integrated model-based development

- Weak commercial tool support
- Graphical editor support in OSATE was never strong enough for industrial use
- Workflow integration suffers because AADL does not include "lighter weight features" for things like stakeholder roles, use cases, sequence charts
  - Necessitated two modeling tools (develop models in SysML, export to AADL for analysis, etc.)

SysMLv2



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

- Will have wide-ranging commercial tool support as well as open source implementations
- Re-engineered from the ground up
  - No backwards compatibility with SysMLv1 except through translation
  - Not built as a profile of UML
- 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

AADL / SysMLv2 Integration
OMG Standards Work

About the SMC

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

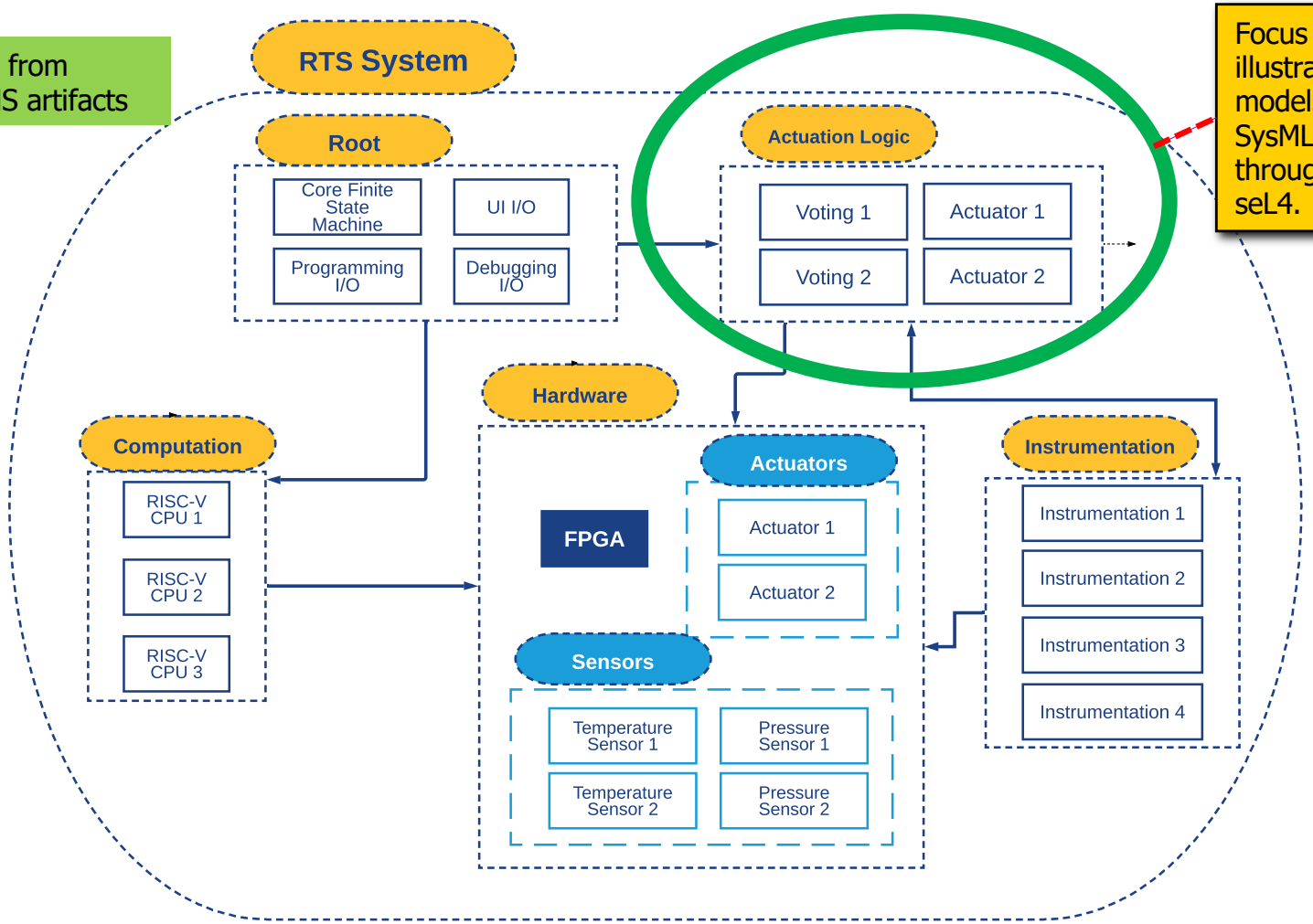
Charter: "*Develop domain libraries w/ KerML & SysMLv2 to support the precise modeling of Real-Time Embedded Safety-Critical Systems. Integrate capabilities from domain-specific models like SAE AADL, OpenGroup FACE, OMG MARTE, & AutoSAR*"

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

HARDENS Nuclear RTS Example

Open source demonstrator project of a reactor trip system built by Galois for the Nuclear Regulator Commission to demonstrate aspects of rigorous digital engineering

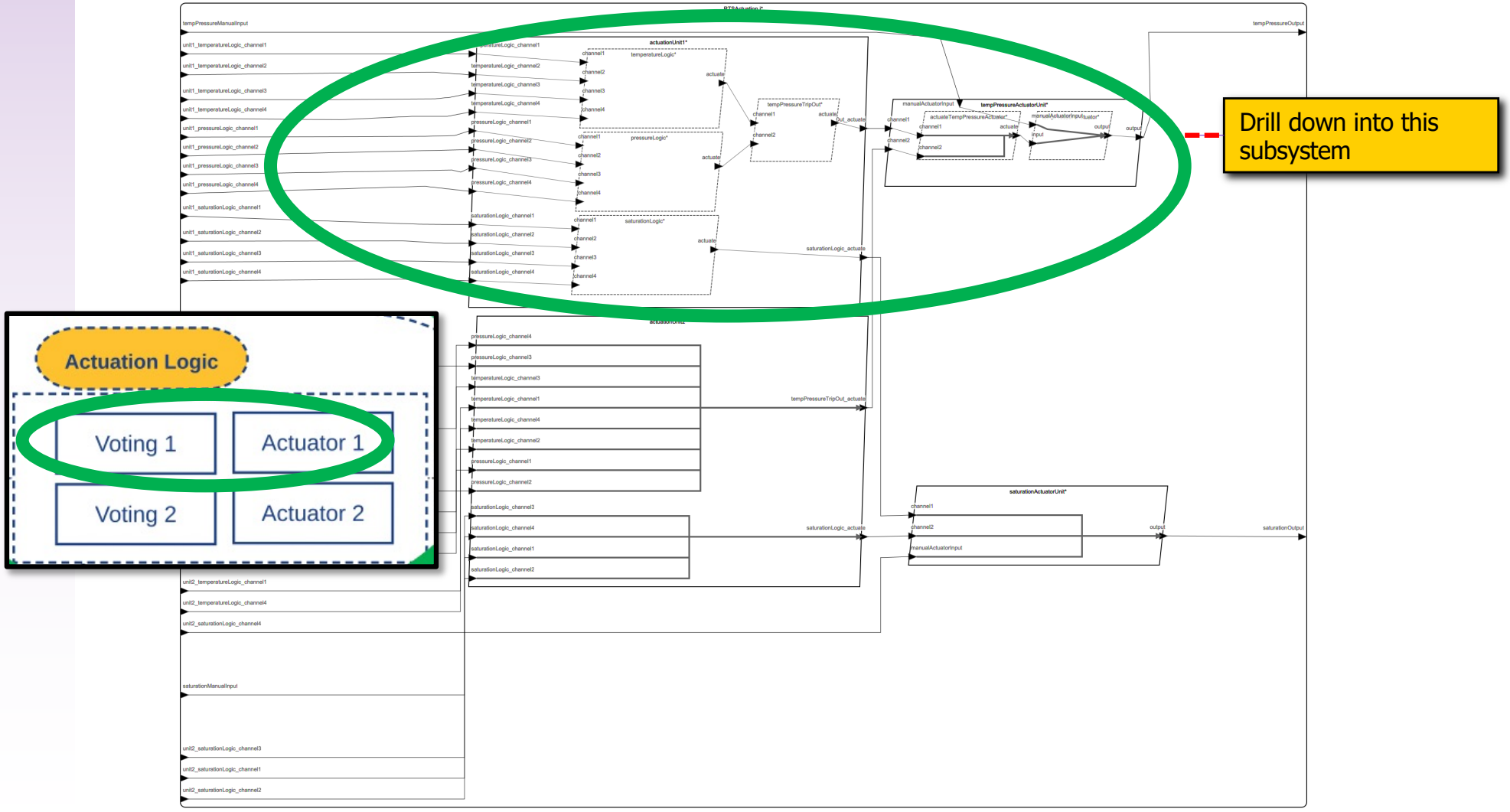
Diagram from HARDENS artifacts



Focus of HAMR illustration effort: models in AADL and SysMLv2, running through HAMR to seL4.

HARDENS Actuation Logic Subsystem in AADL

Actuation Logic Subsystem in AADL (as rendered by OSATE, with substantial tweaks for more readable layout)



Representing AADL in SysMLv2

RTESC workgroup represents AADL concepts as SysMLv2 types, attributes, etc.

SAE INTERNATIONAL AEROSPACE STANDARD

*Mark as AADL **thread***

SysMLv2

Set AADL port categories and types

```
part def CoincidenceLogic :> Thread  {

 in port channel1: DataPort {:> valueType : Base_Types::Boolean; }
 in port channel2: DataPort {:> valueType : Base_Types::Boolean; }
 in port channel3: DataPort {:> valueType : Base_Types::Boolean; }
 in port channel4: DataPort {:> valueType : Base_Types::Boolean; }
 out port actuate: DataPort {:> valueType : Base_Types::Boolean; }
@Properties {
    Dispatch_Protocol = Periodic;
    Period = 1000;
  }
}
```

Set AADL pre-defined property values for this thread

AADL Domain Library for SysMLv2

Developer uses domain library to annotate base SysMLv2 elements with AADL concepts

AADL / SysMLv2 Component Types Side-by-Side

***PROVERS**: Prototype - support a subset of SysMLv2 corresponding to HAMR-supported AADL*

...illustration using Galois HARDENS nuclear reactor trip system (excerpts)

AADL

```
thread CoincidenceLogic
  features
  channel1: in data port Base_Types::Boolean;
  channel2: in data port Base_Types::Boolean;
  channel3: in data port Base_Types::Boolean;
  channel4: in data port Base_Types::Boolean;
  actuate: out data port Base_Types::Boolean;
  properties
   Dispatch_Protocol => Periodic;
   Period => 1000ms;
end CoincidenceLogic;

thread implementation CoincidenceLogic.i
end CoincidenceLogic.i;
```

SysMLv2

```
part def CoincidenceLogic :> Thread {

 in port channel1 : DataPort {:> valueType : Base_Types::Boolean; }
 in port channel2 : DataPort {:> valueType : Base_Types::Boolean; }
 in port channel3 : DataPort {:> valueType : Base_Types::Boolean; }
 in port channel4 : DataPort {:> valueType : Base_Types::Boolean; }
 out port actuate : DataPort {:> valueType : Base_Types::Boolean; }
 @Properties {
   Dispatch_Protocol = Periodic;
   Period = 1000;
  }
}
```

https://github.com/santoslab/rts-showcase

Challenges

Challenges in migrating AADL Formal Methods to SysMLv2

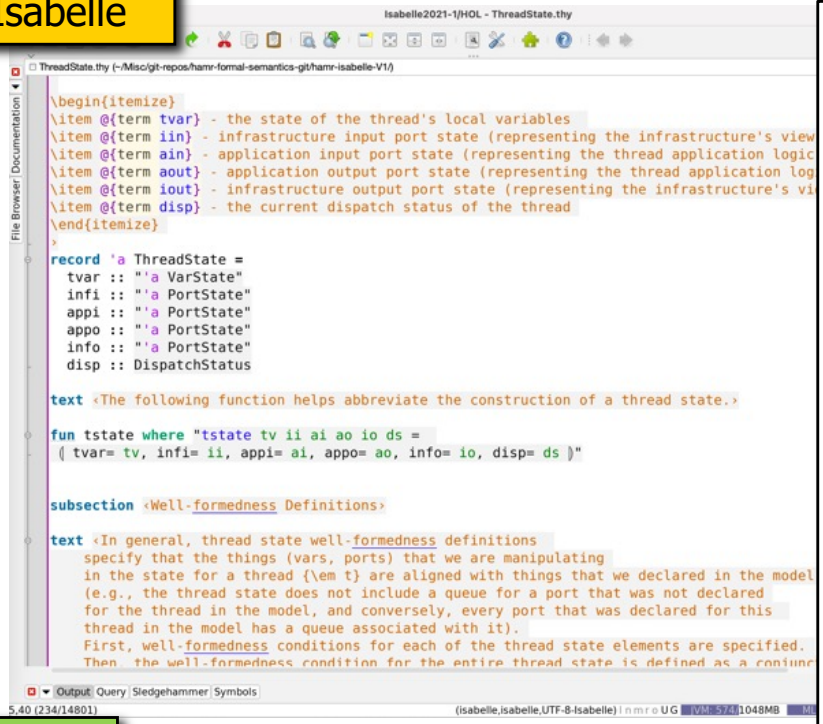
- SysMLv2 has no "annex mechanism"; need to figure out how to represent AADL Annexes
  - behavior contracts, architectural constraints language, hazard analysis
- Representation of AADL Properties
  - model configuration parameters
- Developing a suitable open source SysMLv2 implementation to support research
  - KSU is building a prototype SysMLv2 that will be used on DARPA PROVERS
- 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

AADL / HAMR Formal Semantics

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

Joint work with Stefan Hallerstede (U. Aarhus)

Isabelle



Latex/PDF generated from Isabelle

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 for a thread *t* 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* ⇒ *CompId* ⇒ (*'a VarState*) ⇒ *bool* **where**
wf-ThreadState-tvar m c vs ≡ *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* ⇒ *CompId* ⇒ (*'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* ⇒ *CompId* ⇒ (*'a PortState*) ⇒ *bool* **where**
wf-ThreadState-appi m c ps ≡ *wf-PortState ps* {*p* . *isInCIDPID m c p*}

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

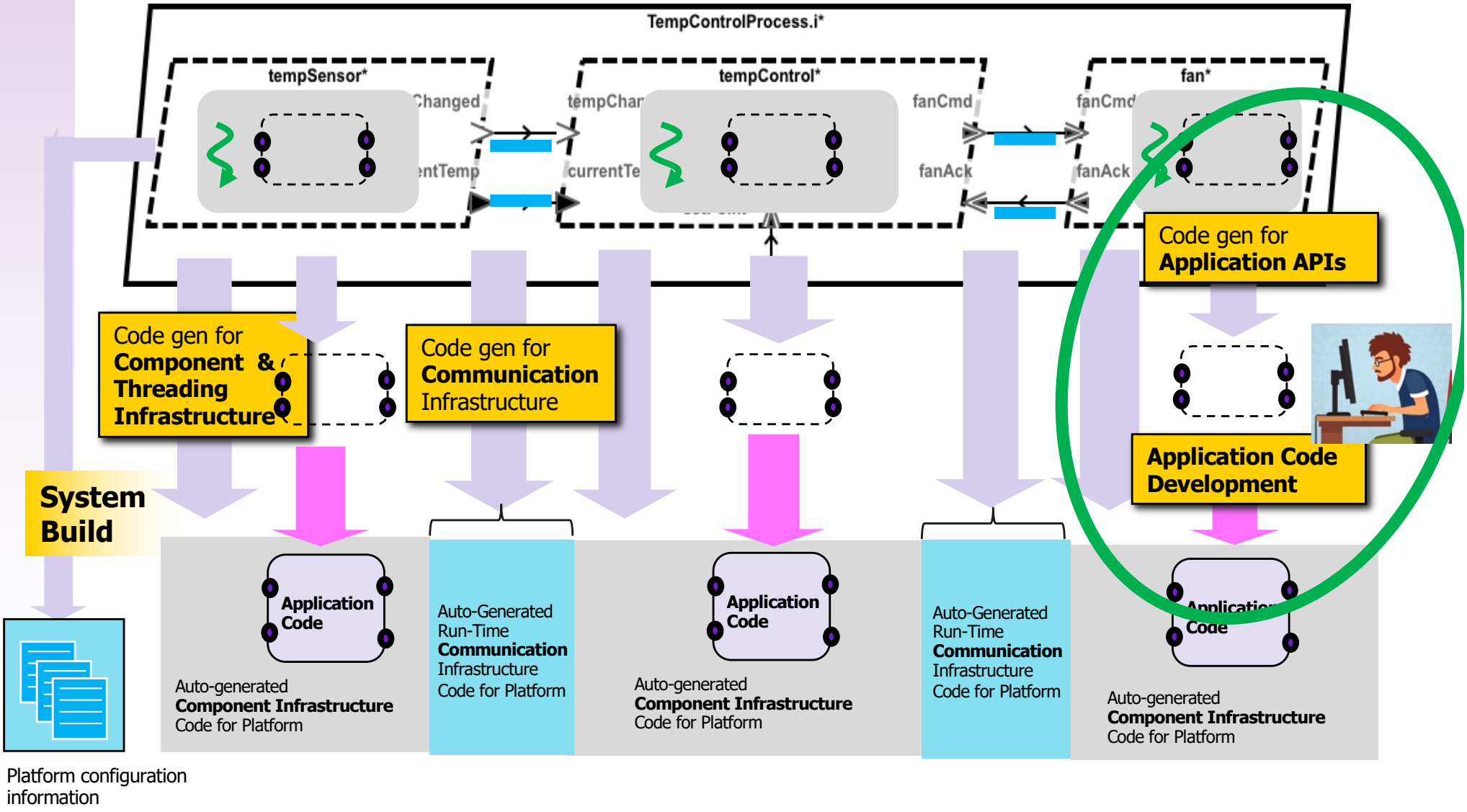
definition *wf-ThreadState-info*:: *Model* ⇒ *CompId* ⇒ (*'a PortState*) ⇒ *bool* **where**
wf-ThreadState-info m c ps ≡ *wf-PortState ps* {*p* . *isOutCIDPID m c p*}

PROVERS

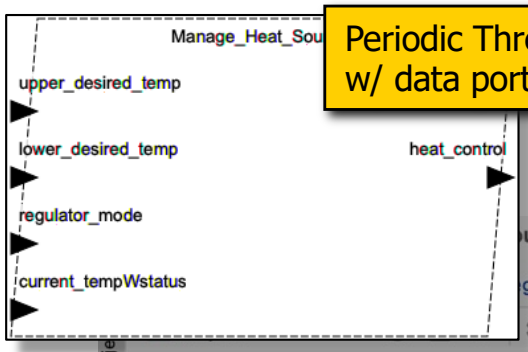
- Enhanced and scope expanded
- Prove soundness of contract framework
- Extend formalization downwards towards seL4 proof-base

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

HAMR Code Generation



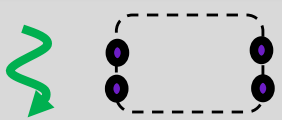
Component Application Code Interfaces Generated from AADL Model



Periodic Thread w/ data ports

...Interfaces/APIs/Skeletons for application code are auto-generated from AADL model

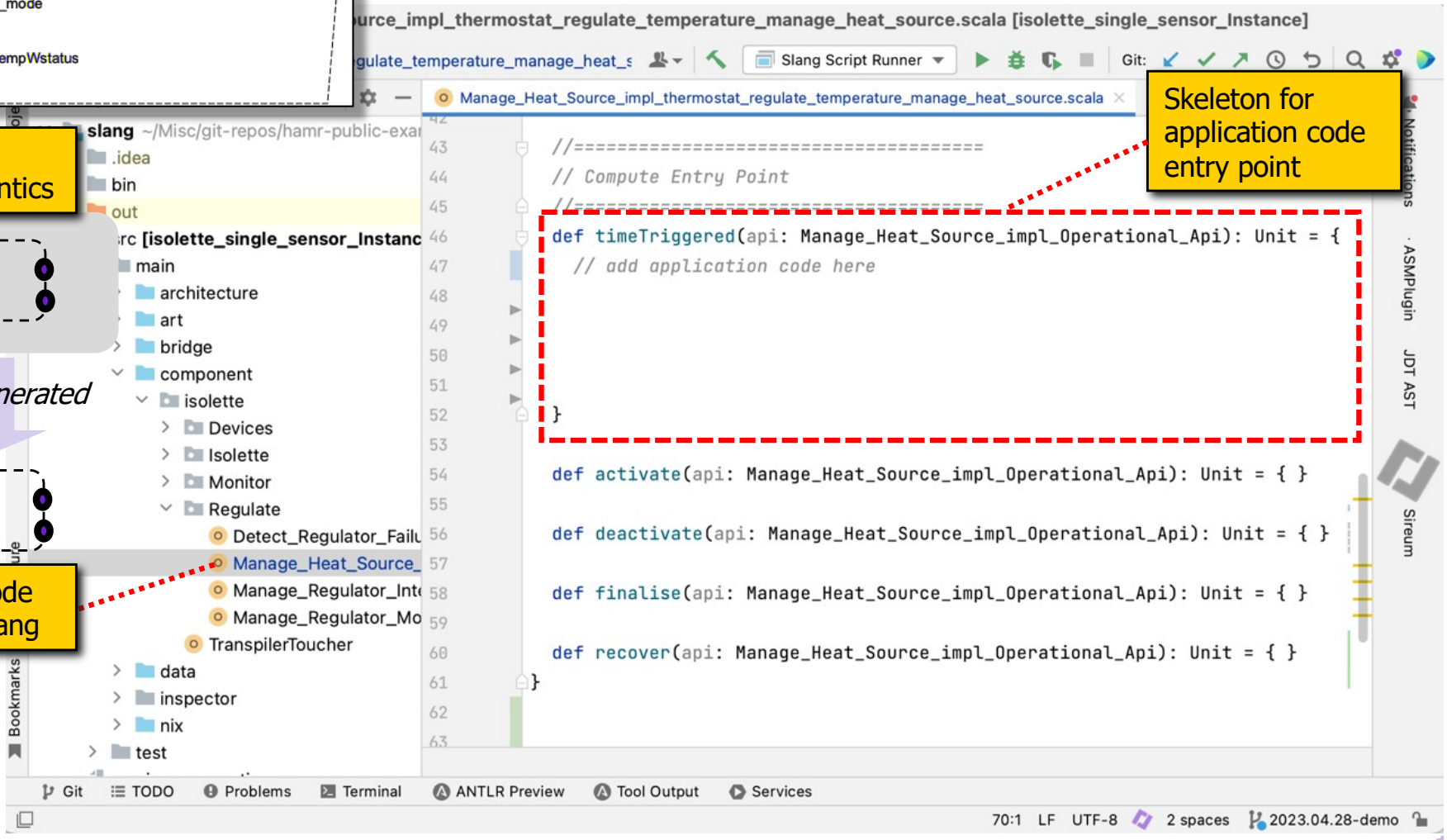
AADL Model Implied Semantics



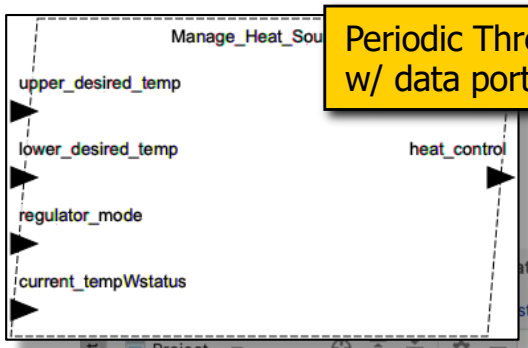
auto-generated

Application Code Skeleton in Slang

Skeleton for application code entry point



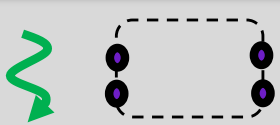
Component Application Code Interfaces Generated from AADL Model



Periodic Thread w/ data ports

...Interfaces/APIs/Skeletons for application code are auto-generated from AADL model

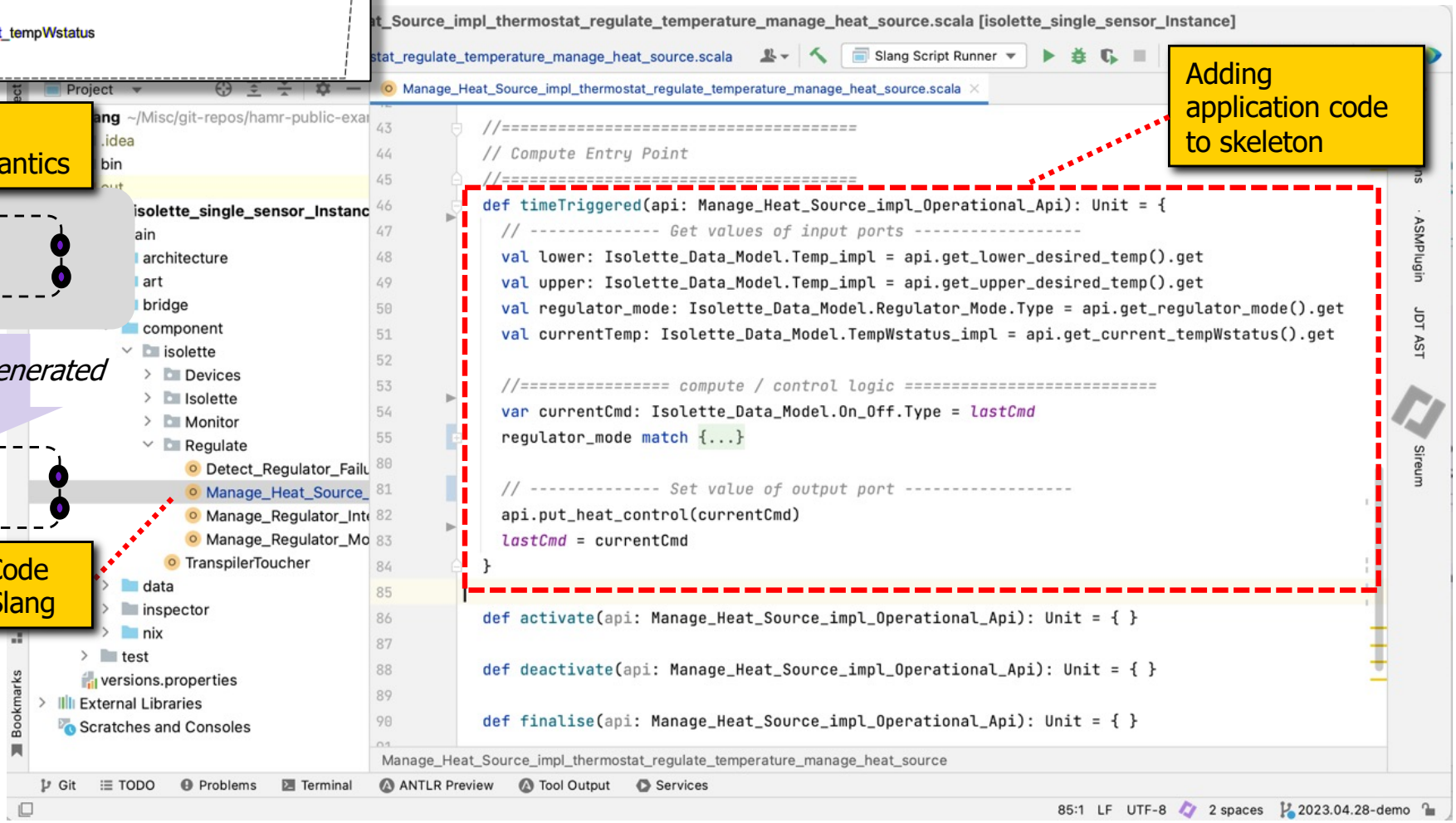
AADL Model Implied Semantics



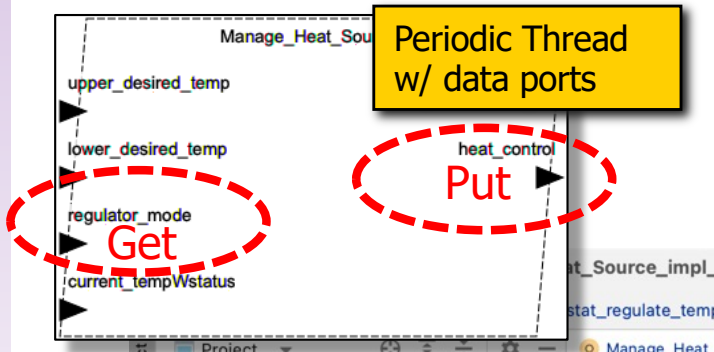
auto-generated

Application Code Skeleton in Slang

Adding application code to skeleton



Component Application Code Interfaces Generated from AADL Model



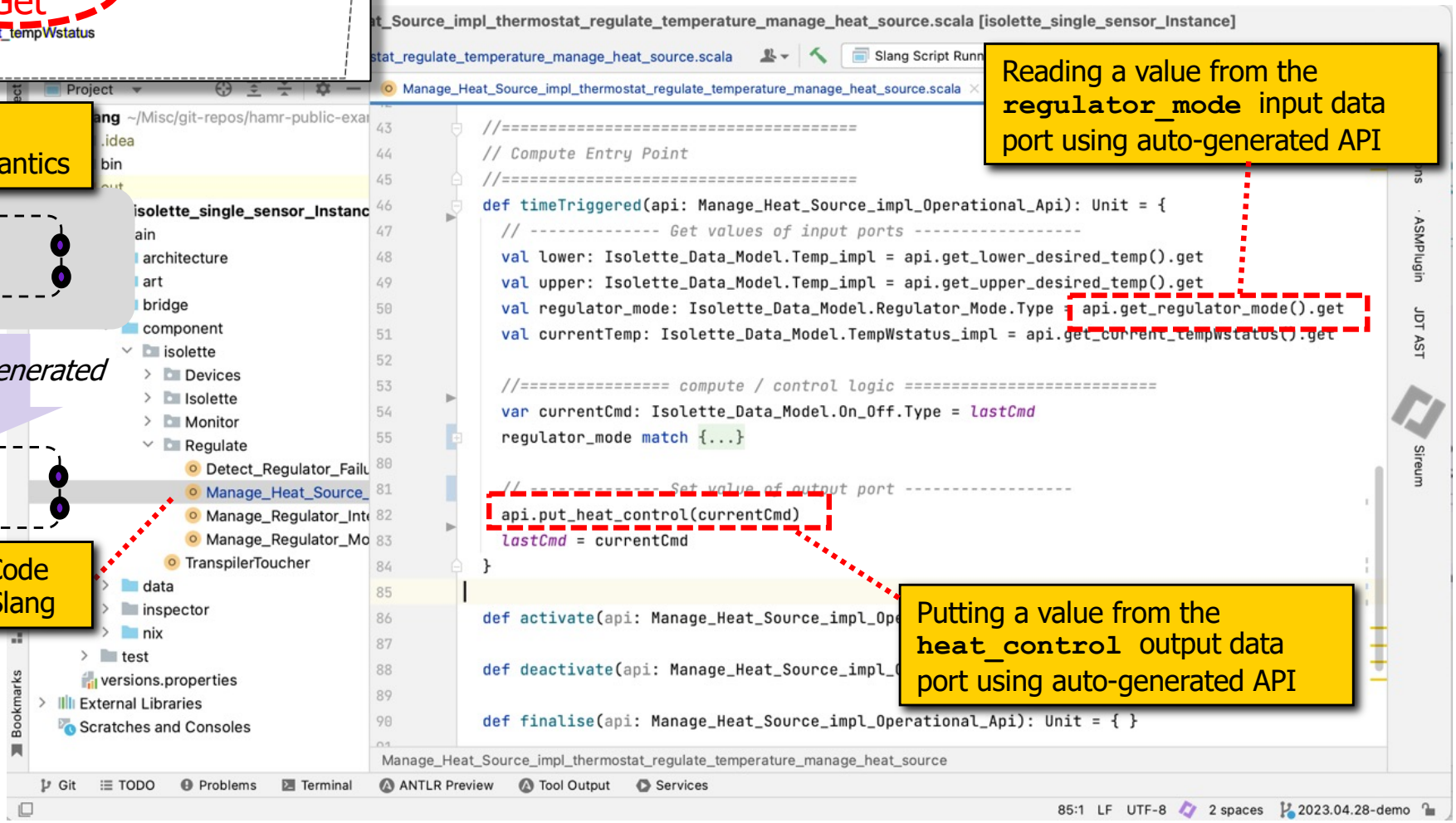
Periodic Thread w/ data ports

...Interfaces/APIs/Skeletons for application code are auto-generated from AADL model

AADL Model Implied Semantics

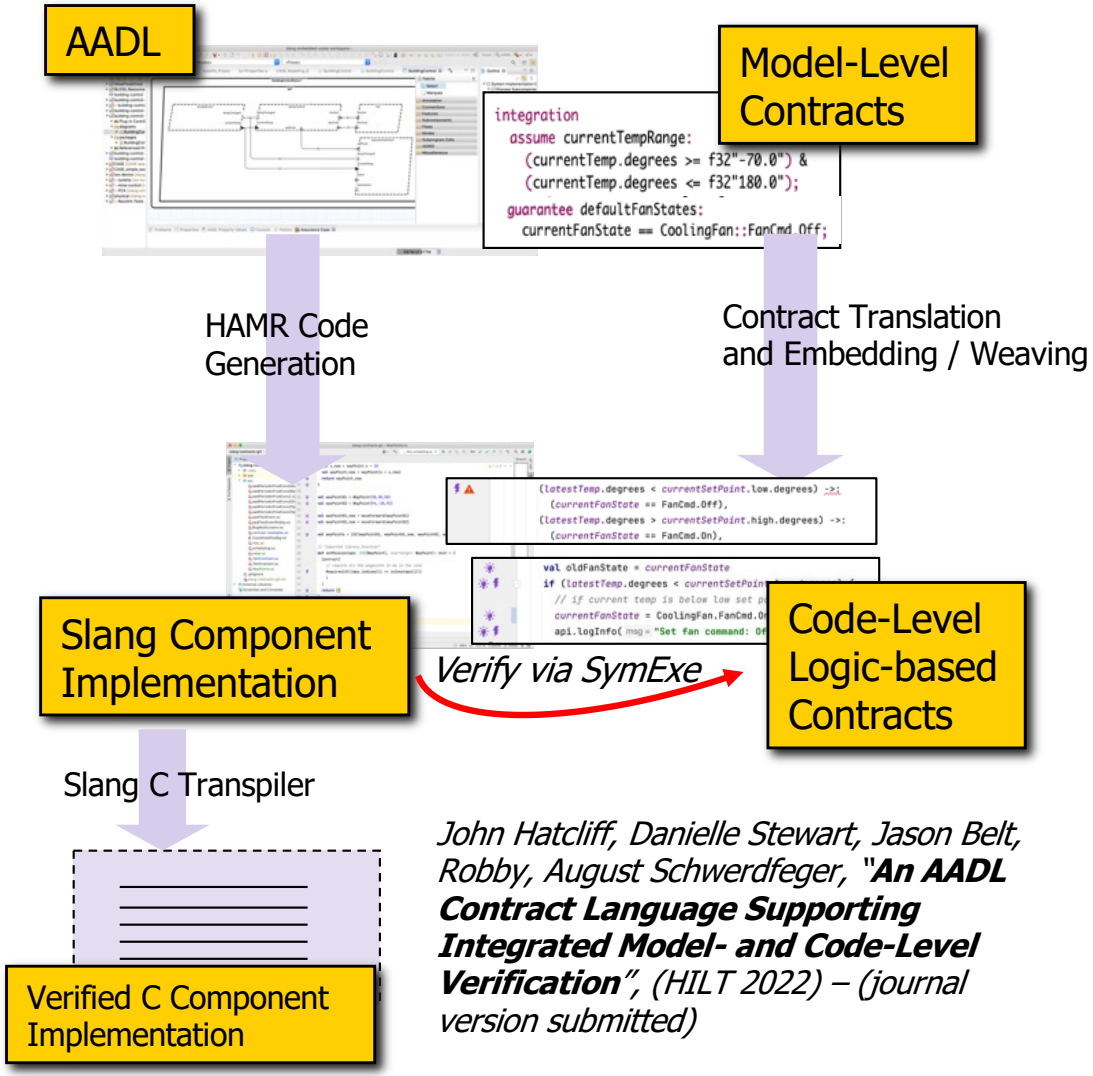
auto-generated

Application Code Skeleton in Slang



Application:
Integrated Model/Code Contract Language

KSU / Galois - US Army research project (SBIR Phase II)...



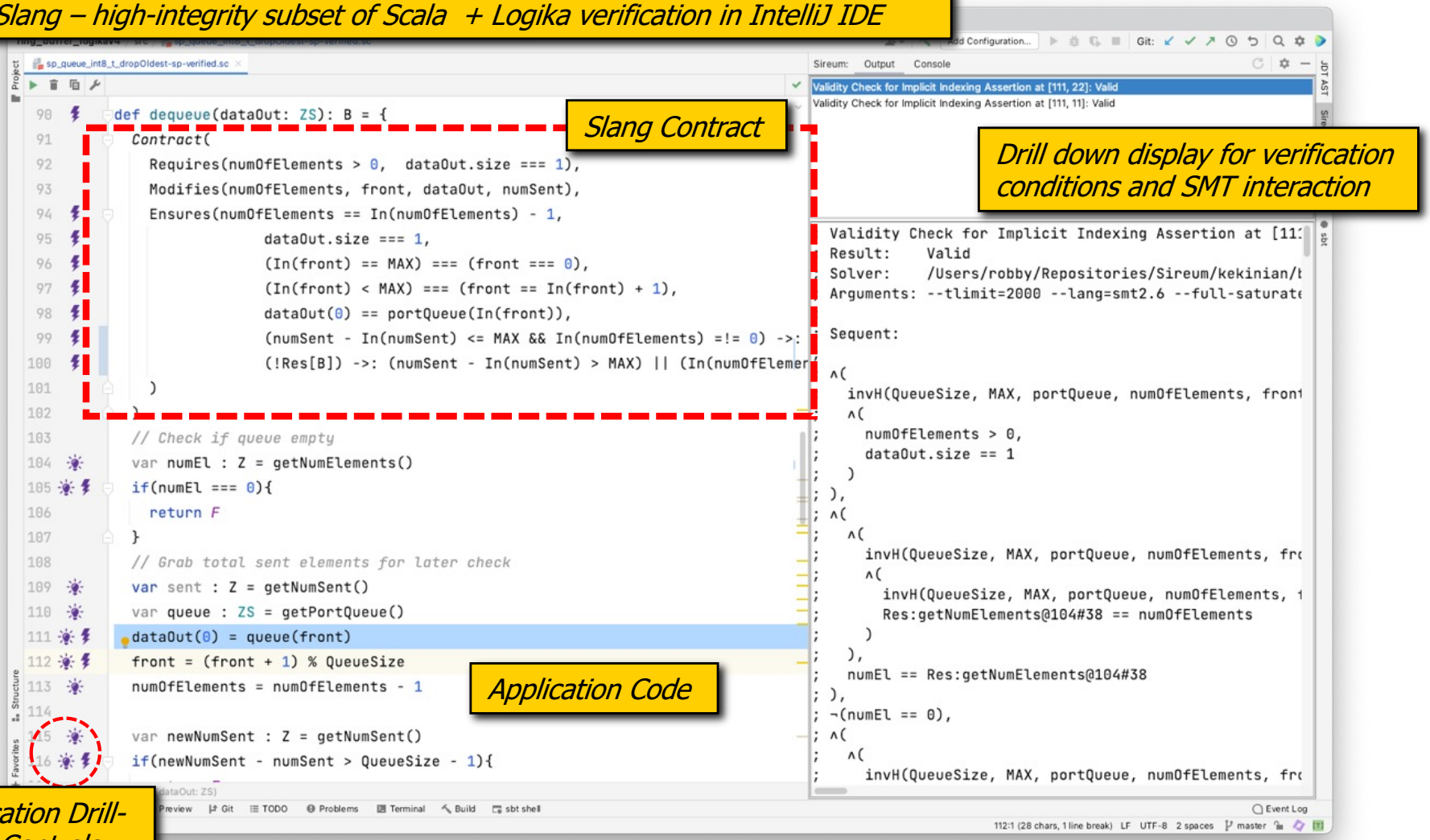
*John Hatcliff, Danielle Stewart, Jason Belt, Robby, August Schwerdfeger, "**An AADL Contract Language Supporting Integrated Model- and Code-Level Verification**", (HILT 2022) – (journal version submitted)*

DARPA SBIR w/ Galois

Slang Contracts

and Automated Verification via Symbolic Execution (Logika)

Slang – high-integrity subset of Scala + Logika verification in IntelliJ IDE



Slang applications can be integrated with Scala and Java and executed on JVM or transpiled to JS or C. The generated C has bounded memory usage and no garbage collection & compatible with verified CompCert compiler.

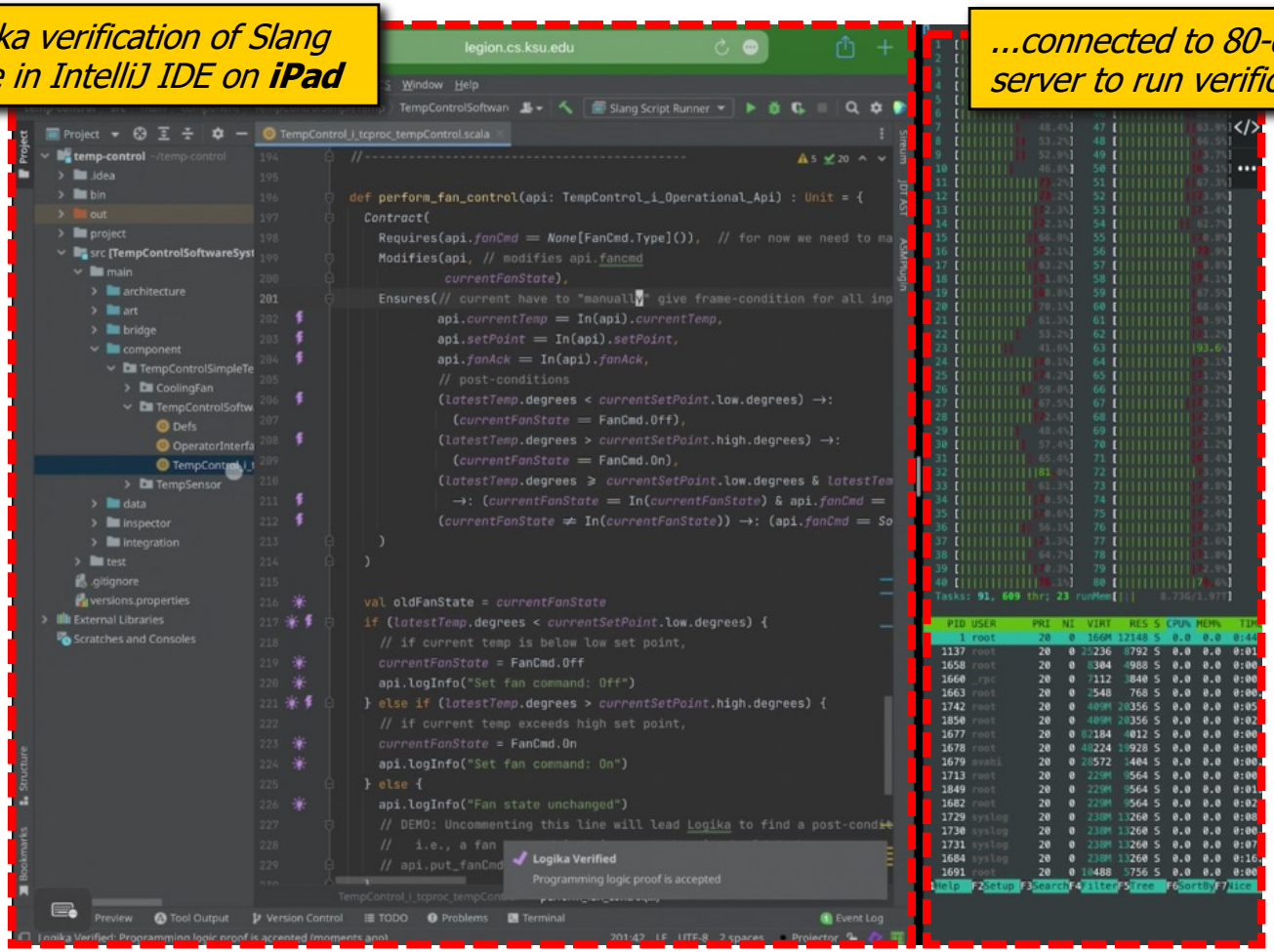
Logika Verification

Featureful, Integrated Capabilities

Logika has a **server-based architecture** with a suite of SMT solvers (Z3, CVCx, Alt-Ergo), uses massive parallelization, with "always on" smart incremental checking

*Logika verification of Slang code in IntelliJ IDE on **iPad***

...connected to 80-core server to run verification

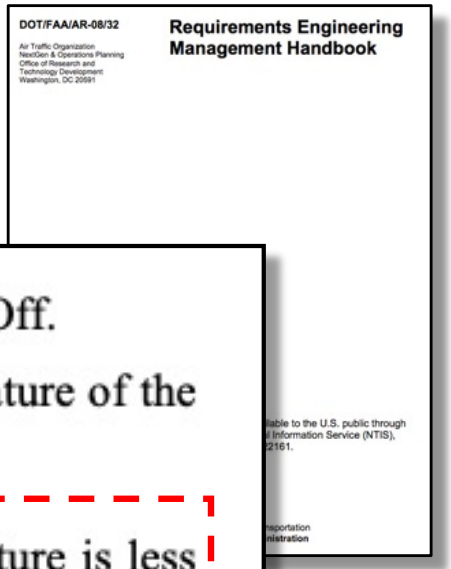


From a TCCOE conference demo video of Logika in January 2022

See https://drive.google.com/uc?export=download&id=1vkBNWM8pocSz8jUG-E16zdVleELZr2Sk for Slang / Logika overview talk given at the Trusted Computing Center of Excellence Symposium

Requirements to Contracts

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


DOT/FAA/AR-08/32

Air Traffic Organization
NextGen & Operations Planning
Office of Research and
Technology Development
Washington, DC 20591

Requirements Engineering Management Handbook

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.

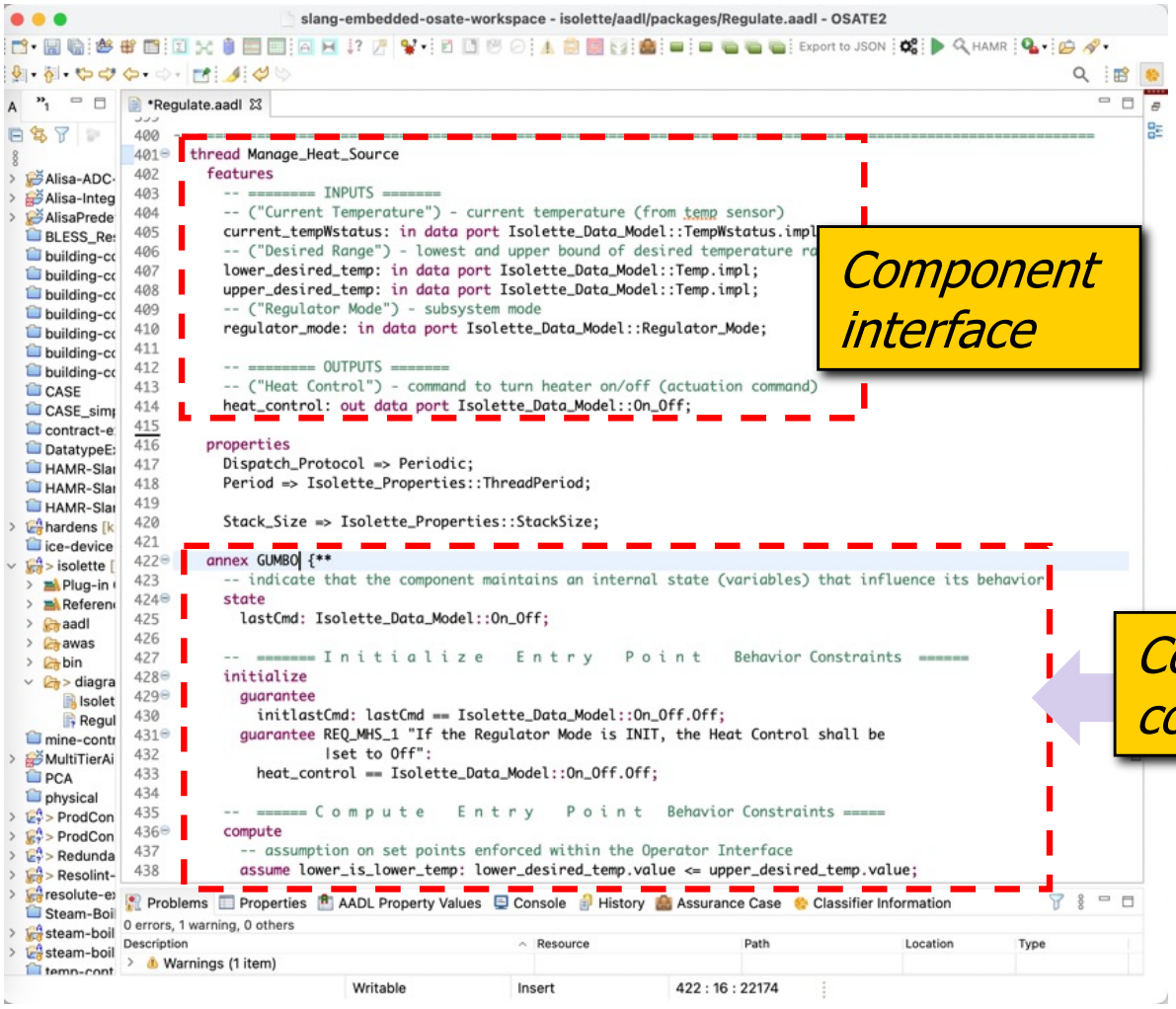
REQ-MHS-3: If the Regulator Mode is NORMAL and the Current Temperature is greater than the Upper Desired Temperature, the Heat Control shall be set to Off.

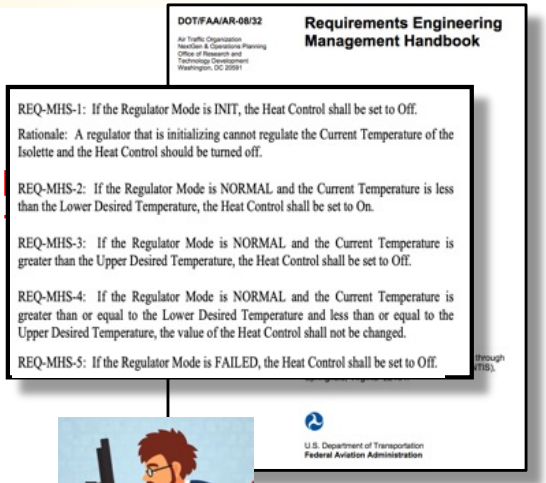
REQ-MHS-4: If the Regulator Mode is NORMAL and the Current Temperature is greater than or equal to the Lower Desired Temperature and less than or equal to the Upper Desired Temperature, the value of the Heat Control shall not be changed.

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

Requirements to Contracts

GUMBO contracts are written together with the thread interface in the AADL OSATE IDE (using AADL Annex clause)



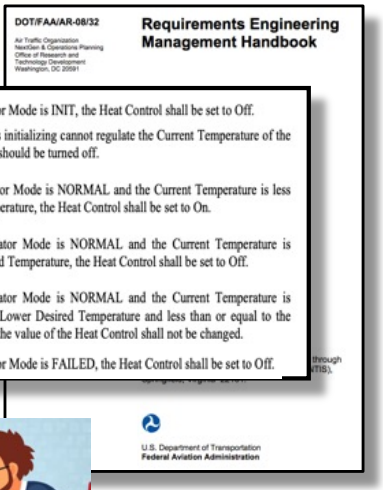


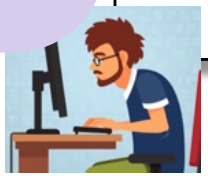
Developer formalizes requirements

Manage Heat Source Contracts

AADL GUMBO Contracts for **Manage Heat Source** Thread, with traceability to REMH requirements.

```
case REQ_MHS_1 "If the Regulator Mode is INIT, the Heat Control shall be
        |set to Off.":
  assume regulator_mode == Isolette_Data_Model::Regulator_Mode.Init_Regulator_Mode;
  guarantee heat_control == Isolette_Data_Model::On_Off.Off;

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.Onn;

case 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.":
  assume (regulator_mode == Isolette_Data_Model::Regulator_Mode.Normal_Regulator_Mode)
    & (current_tempWstatus.value > upper_desired_temp.value);
  guarantee heat_control == Isolette_Data_Model::On_Off.Off;

case 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.":
  assume (regulator_mode == Isolette_Data_Model::Regulator_Mode.Normal_Regulator_Mode)
    & (current_tempWstatus.value >= lower_desired_temp.value
    & current_tempWstatus.value <= upper_desired_temp.value);
  guarantee heat_control == In(lastCmd);

case REQ_MHS_5 "If the Regulator Mode is FAILED, the Heat Control shall be
        |set to Off.":
  assume regulator_mode == Isolette_Data_Model::Regulator_Mode.Failed_Regulator_Mode;
  guarantee heat_control == Isolette_Data_Model::On_Off.Off;
```





Developer formalizes requirements

OSATE AADL Editor

Manage Heat Source Contracts

AADL GUMBO Contracts for **Manage Heat Source** Thread, with traceability to REMH requirements.

Mode condition

Compare current temperature to desired range

...

```
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.Onn;
```

...

Set the desired state of the heater

OSATE AADL Editor

Automatically Embedded Slang Logical Contracts

Verification against contracts using Logika tool (Symbolic Execution)

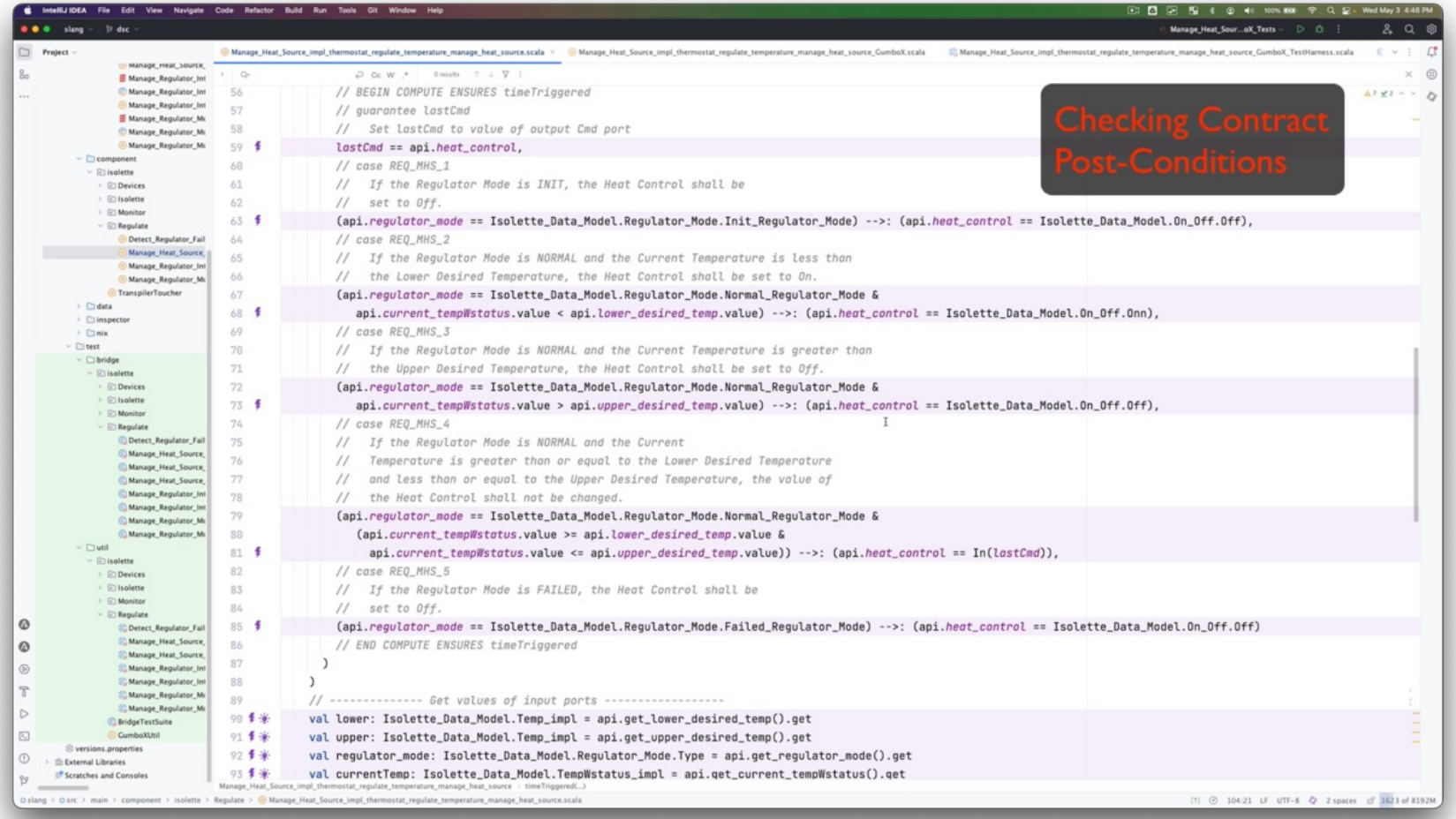
Manage_Heat_Source_impl_thermostat_regulate_temperature_manage_heat_source.scala

```
47  def timeTriggered(api: Manage_Heat_Source_impl_Operational_Api): Unit = {
48    Contract(
49      Requires(
50        // BEGIN COMPUTE REQUIRES timeTriggered
51        // assume lower_is_lower_temp
52        api.lower_desired_temp.value <= api.upper_desired_temp.value
53        // END COMPUTE REQUIRES timeTriggered
54      ),
55      Modifies(api.lastCmd),
56      Ensures(
57        // BEGIN COMPUTE ENSURES timeTriggered
58        // guarantee lastCmd
59        //   Set lastCmd to value of output Cmd port
60        lastCmd == api.heat_control,
61        // case REQ_MHS_1
62        //   If the Regulator Mode is INIT, the Heat Control shall be
63        //   set to Off.
64        //   http://pub.santoslab.org/high-assurance/module-requirements/reading/FAA-DoT-Requirements-AR-08-32.pdf#page=110
65        (api.regulator_mode == Isolette_Data_Model.Regulator_Mode.Init_Regulator_Mode) -->: (api.heat_control == Isolette_Data_Model.On_Off.Off),
66        // case REQ_MHS_2
67        //   If the Regulator Mode is NORMAL and the Current Temperature is less than
68        //   the Lower Desired Temperature, the Heat Control shall be set to On.
69        //   http://pub.santoslab.org/high-assurance/module-requirements/reading/FAA-DoT-Requirements-AR-08-32.pdf#page=110
70        (api.regulator_mode == Isolette_Data_Model.Regulator_Mode.Normal_Regulator_Mode &
71          api.current_tempWstatus.value < api.lower_desired_temp.value) -->: (api.heat_control == Isolette_Data_Model.On_Off.Onn),
72        // case REQ_MHS_3
73        //   If the Regulator Mode is NORMAL and the Current Temperature is greater than
74        //   the Upper Desired Temperature, the Heat Control shall be set to Off.
75        //   http://pub.santoslab.org/high-assurance/module-requirements/reading/FAA-DoT-Requirements-AR-08-32.pdf#page=110
76        (api.regulator_mode == Isolette_Data_Model.Regulator_Mode.Normal_Regulator_Mode &
77          api.current_tempWstatus.value > api.upper_desired_temp.value) -->: (api.heat_control == Isolette_Data_Model.On_Off.Off),
```

HAMR automatically translates AADL contracts into code-level Slang contracts.

Demo

Verification against contracts using Logika tool (Symbolic Execution)



Rust Verification via Verus

Rust code with integrated contracts and verification with Verus (CMU)

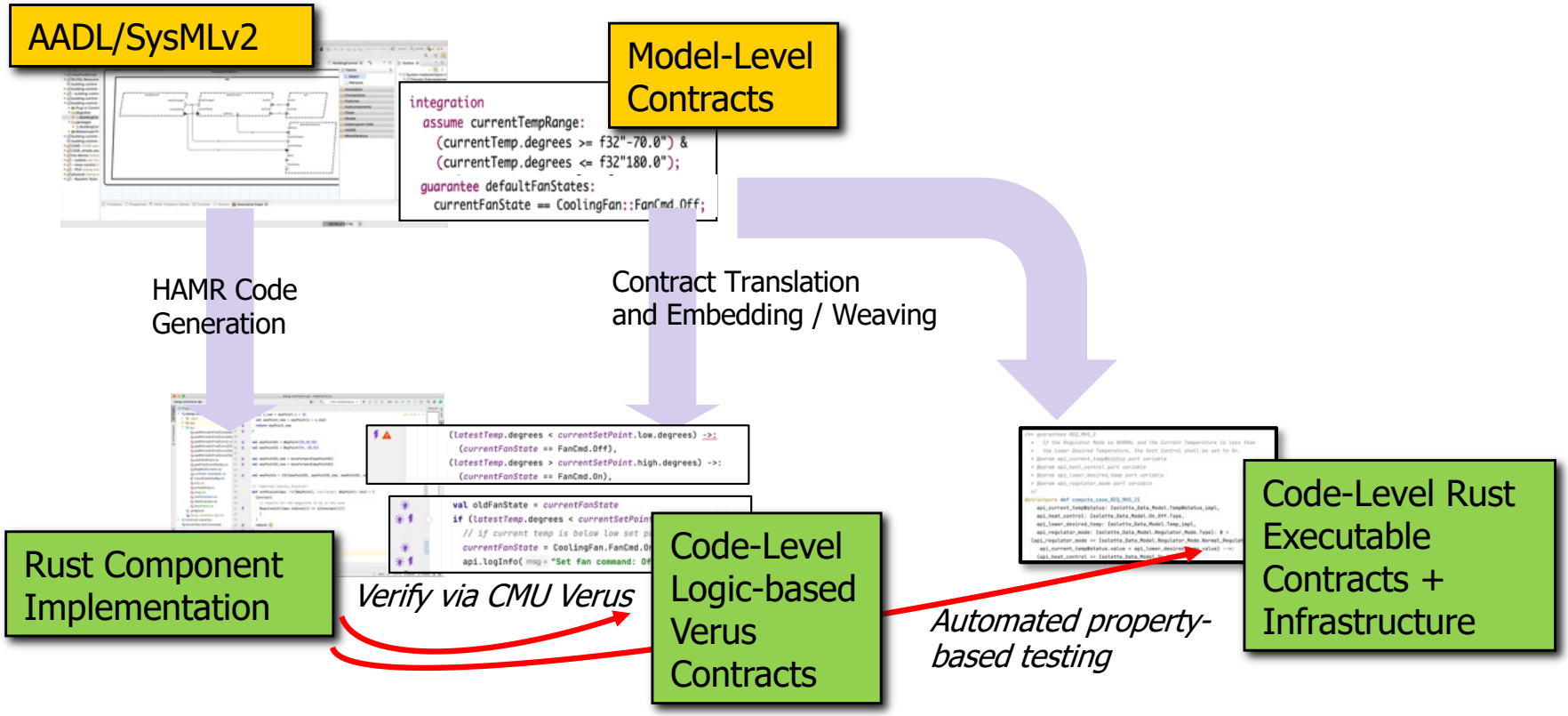
```
26 #[exec] fn fibo_impl(n: u64) -> u64 {
27   requires(fibo_fits_u64(n)); Ⓖ spec
28   ensures(|result: u64| result == fibo(n)); Ⓗ spec
29
30   if n == 0 { return 0; }
31   let mut prev: u64 = 0;
32   let mut cur: u64 = 1;
33   let mut i: u64 = 1;
34   while i < n {
35     invariant([ Ⓘ spec
36       0 < i && i <= n,
37       fibo_fits_u64(n as nat),
38       fibo_fits_u64(i as nat),
39       cur == fibo(i),
40       prev == fibo(i as nat - 1),
41     ]);
42     let new_cur = cur + prev;
43     prev = cur;
44     cur = new_cur;
45     assert(prev == fibo(i as nat)); Ⓙ proof
46     i = i + 1;
47     lemma_fibo_is_monotonic(i, n); Ⓚ proof
48   }
49   cur
50 }
```

Verus contracts (pre/post-conditions)

Inline invariants and assertions

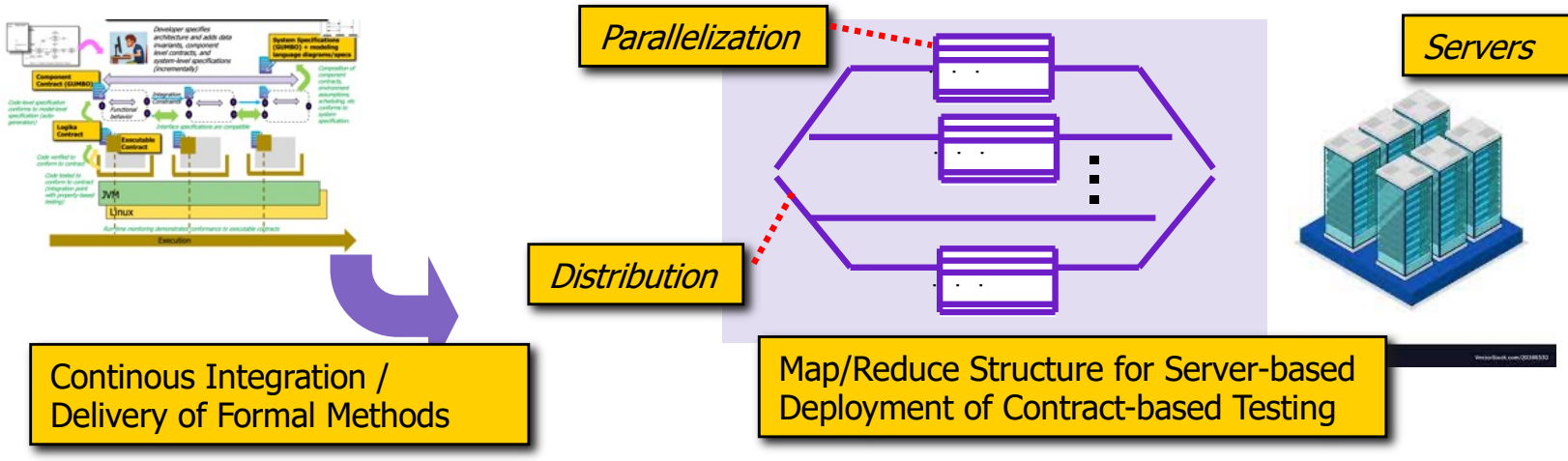
Integrated Model/Code Contracts

Extend existing Slang-based framework to support Rust..



Scaling Up - Property-based Testing Server-Based Deployment

For Slang property-based testing, HAMR generates a server-based deployment to run the framework in a distributed/parallel fashion...



- Random generators and contract-based tests are farmed out to a configurable family of servers
- Test vectors and results are serialized for flexible deployment, reporting, and replay of the tests
- Currently hosted using our Jenkins setup, but easy for HAMR to automatically generate deployment scripts, e.g., for AWS, in the future

Extensive Assurance Artifacts

HAMR provides extensive auto-generation and reporting of assurance artifacts

PROVERS:
Integrate with Collins *assurance dashboard*

Architecture

Component Model Info
Interfaces

GUMBO Contracts

Component Code
Application

Executable Contracts

Static Metrics

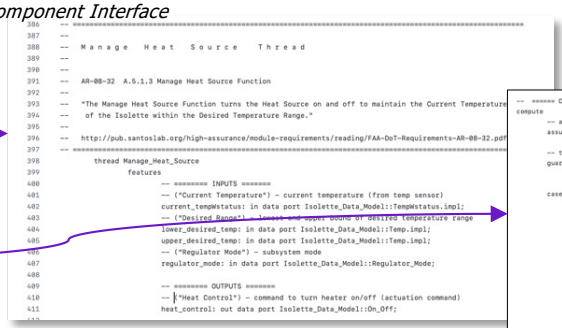
Configurations
Config Name$_1$

Config Name$_2$

⋮

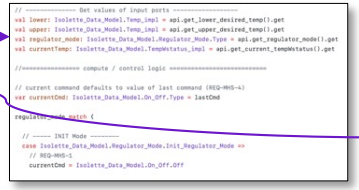
Property Satisfaction & Coverage

Component Interface

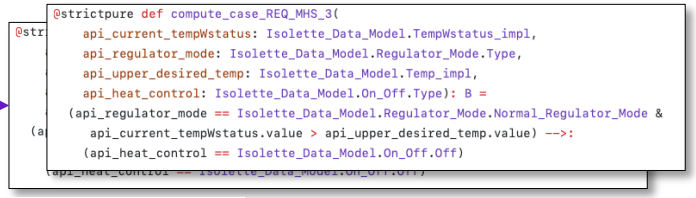


GUMBO Contracts

Component Code



Component Executable Contracts

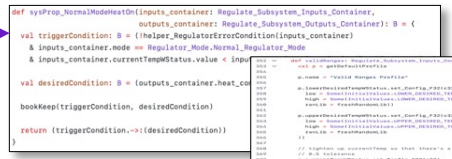


```
@strictpure def compute_case_REQ_MHS_3(
    api_current_tempWstatus: Isolette_Data_Model.TempWstatus_impl,
    api_regulator_mode: Isolette_Data_Model.Regulator_Mode.Type,
    api_upper_desired_temp: Isolette_Data_Model.Temp_impl,
    api_heat_control: Isolette_Data_Model.On_Off.Type): B =
  (api_regulator_mode == Isolette_Data_Model.Regulator_Mode.Normal_Regulator_Mode &
    api_current_tempWstatus.value > api_upper_desired_temp.value) -->:
    (api_heat_control == Isolette_Data_Model.On_Off.Off)
```

Configuration N_2

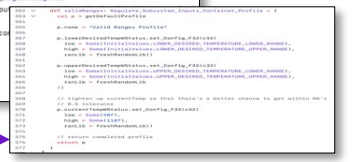
HC__Normal____Heat_On	
Description:	Heat Control control laws; NORMAL mode => Heat ON result state
Script Schema:	Regulator_1HP_script_schema
Property:	sysProp_NormalModeHeatOn
Randomization Profile:	Valid Ranges Profile
Random Vector Filter:	compute_spec_lower_is_not_higher_than_upper_assume
Relevant Coverage:	RegMRI, RegMRM, RegMHS, DRF

Property Specification

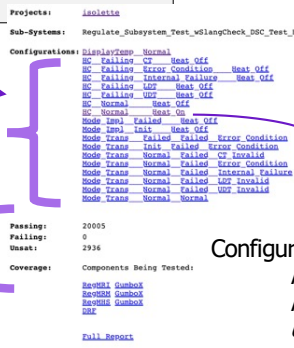


```
def sysProp_NormalModeHeatOn(inputs_container: Regulate_Subsystem_Inputs_Container,
                             outputs_container: Regulate_Subsystem_Outputs_Container): B = {
  val triggerCondition: B = ((helper_RegulatorErrorCondition(inputs_container)
    & inputs_container.mode == Regulator_Mode.Normal_Regulator_Mode
    & inputs_container.currentTempWStatus.value < inpu

  val desiredCondition: B = (outputs_container.heat_co

  bookKeep(triggerCondition, desiredCondition)

  return (triggerCondition.->:(desiredCondition))
}
```

Random Number Generation Profile Specification



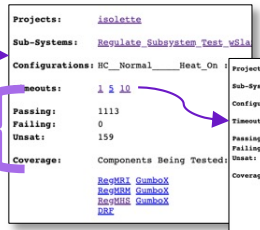
Configurations



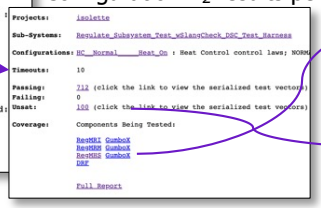
Total Tests
Passing
Failed
Unsat

Configuration Tests
Passing
Failed
Unsat

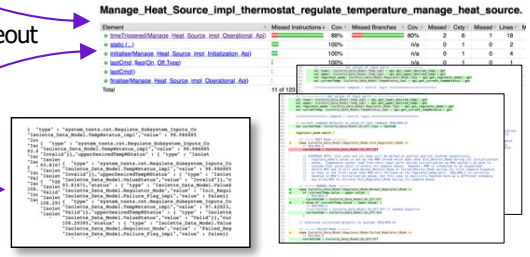
Configuration N_2 results per timeout



Configuration N_2 results per timeout



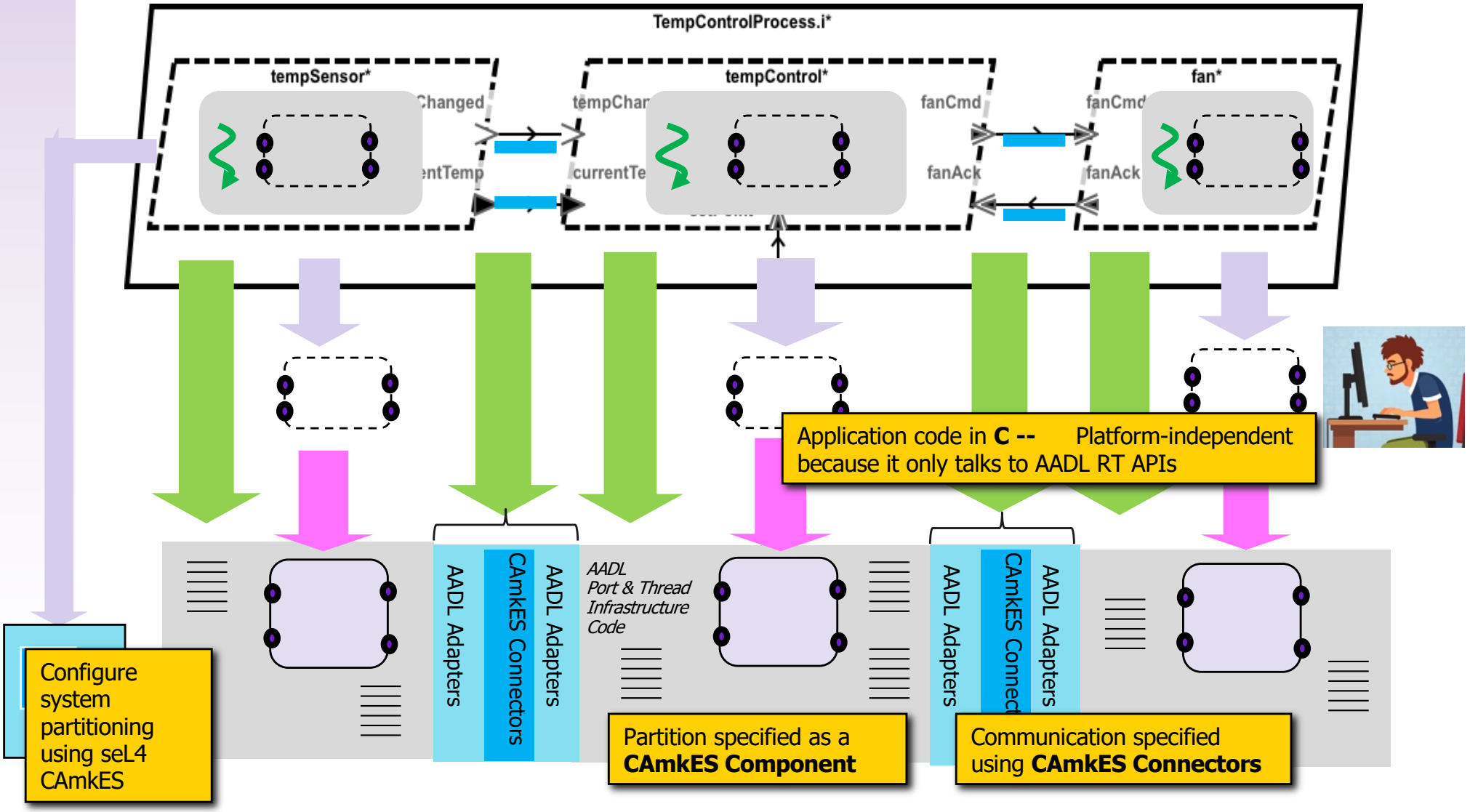
Coverage Reports



Test Vectors

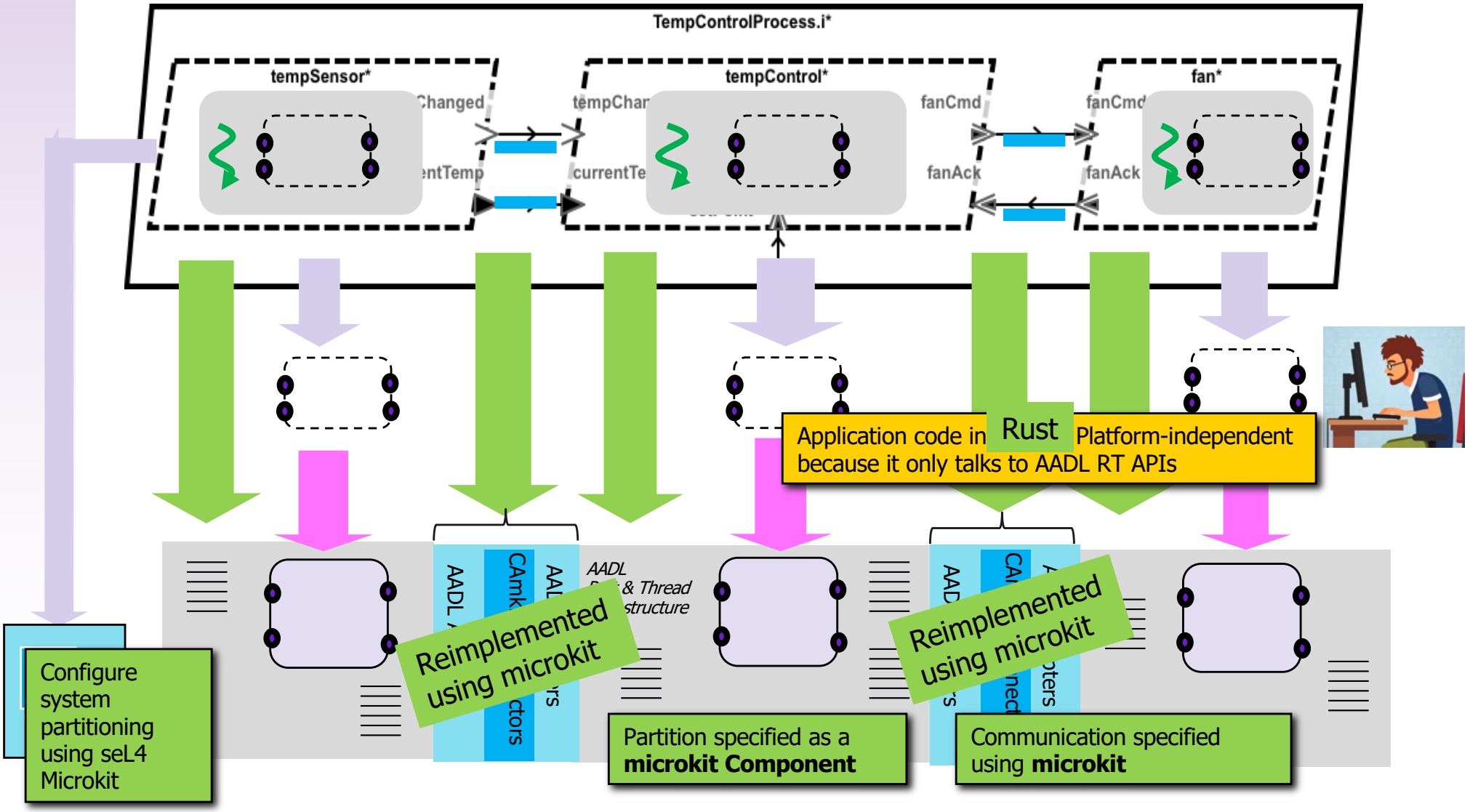
HAMR Code Generation seL4 Platform (baseline from CASE)

HAMR instantiation for C - based development on **SeL4 microkernel** using CAmkES



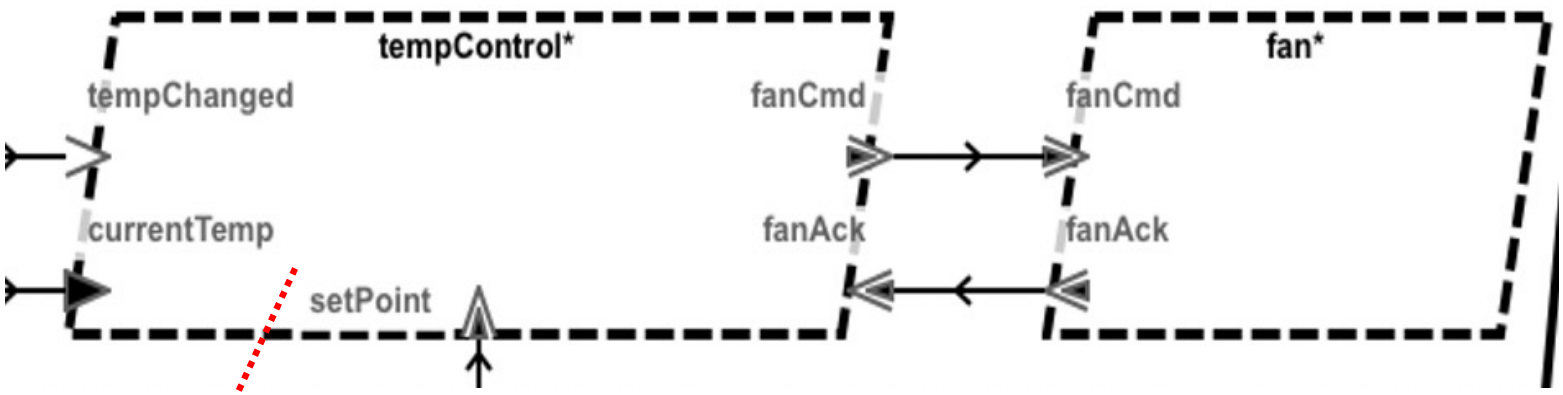
HAMR Code Generation seL4 Platform (PROVERS goals)

HAMR instantiation for Rust based development on **SeL4 microkernel** using seL4 microkit



CAmkES seL4 Configuration

AADL and Platform Independent Application Code



seL4 Platform Configuration Using CAmkES (auto-generated by HAMR)

```
component TempControl {
```

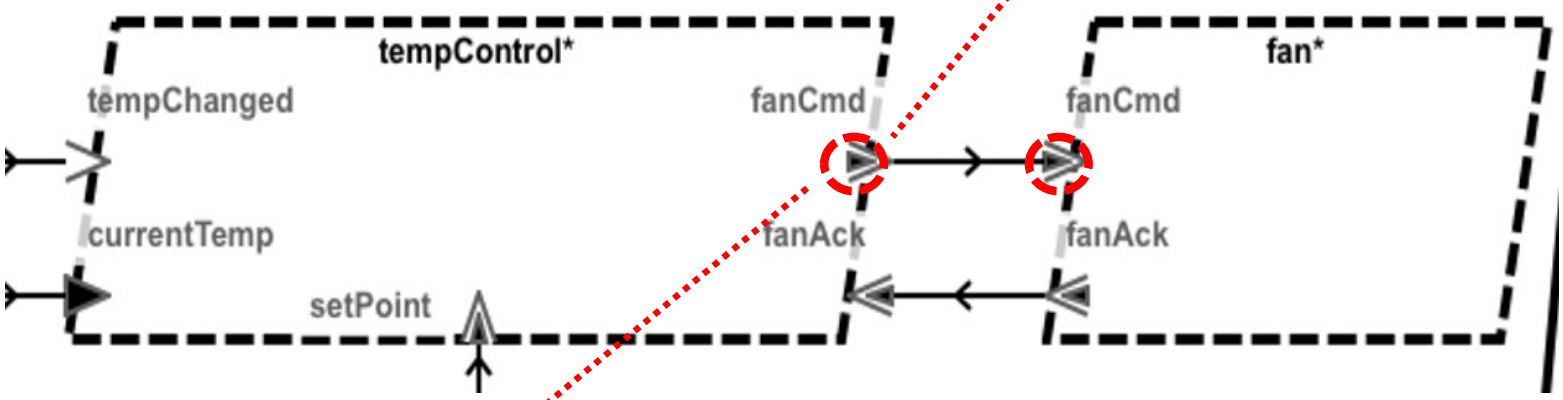
```
component Fan {

}
```

For each AADL thread component create an seL4 partition

CAmkES seL4 Configuration

AADL Event Data port (message with payload)

AADL and Platform Independent Application Code



seL4 Platform Configuration Using CAmkES (auto-generated by HAMR)

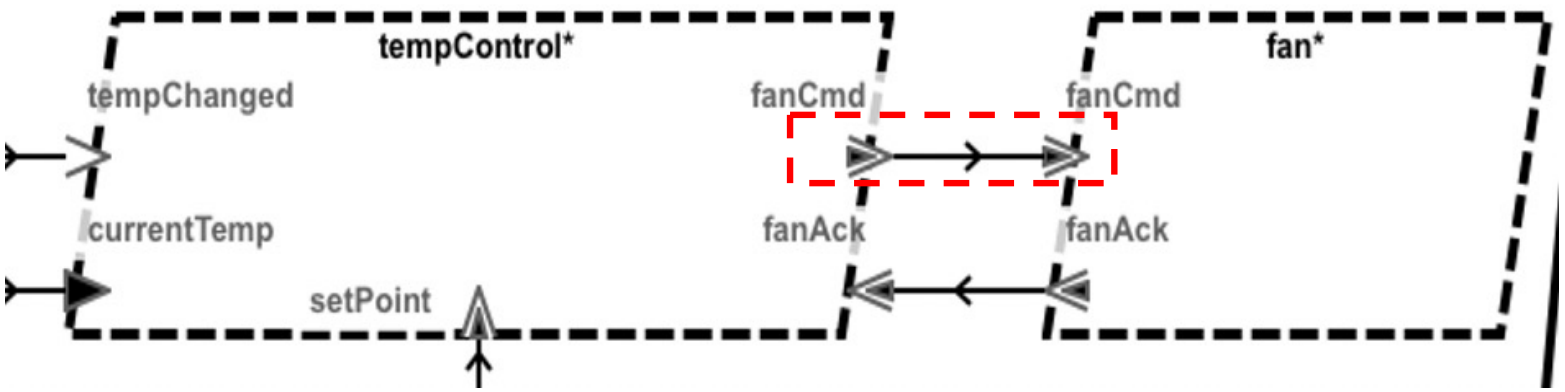
```
component TempControl {
  emits ReceiveEvent
          fanCmd_notification;
  dataport DataContent
             fanCmd_queue;
  ...
}
```

```
component Fan {
  consumes ReceiveEvent
             fanCmd_notification;
  dataport DataContent
             fanCmd_queue;
  ...
}
```

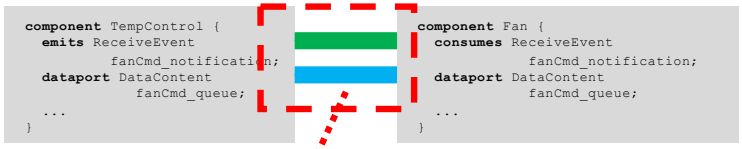
AADL Event Data port is represented using a CAmkES **notification** + **dataport** – introducing finer granularity as we move to platform

CAmkES seL4 Configuration

AADL and Platform Independent Application Code



seL4 Platform Configuration Using CAmkES (auto-generated by HAMR)

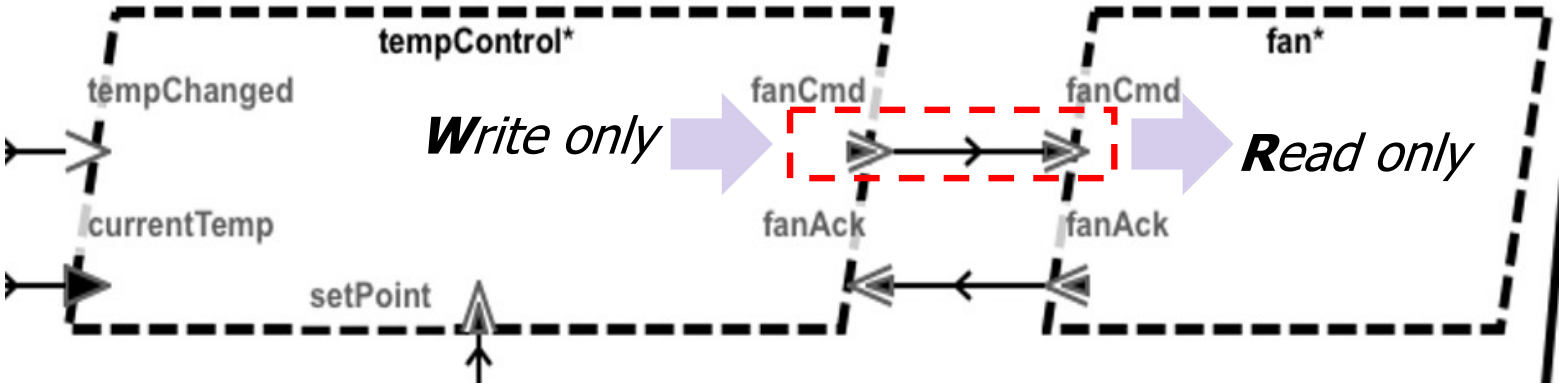


CAmkES **assembly** specifies system topology, including allowed communication between seL4 partitions

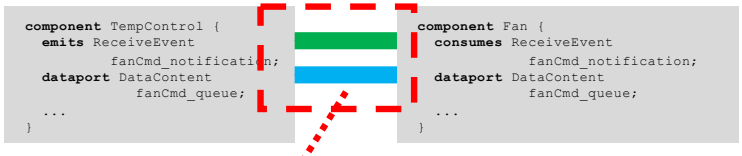
```
assembly {
  composition {
    component TempControl tempControl;
    component Fan fan;
    connection seL4Notification conn4(
      from tempControl.fanCmd_notification,
      to fan.fan_notification);
    connection seL4SharedData conn5(
      from tempControl.fanCmd_queue,
      to fan.fanCmd_queue);
    ... }
}
```

CAmkES seL4 Configuration

AADL and Platform Independent Application Code



seL4 Platform Configuration Using CAmkES (auto-generated by HAMR)



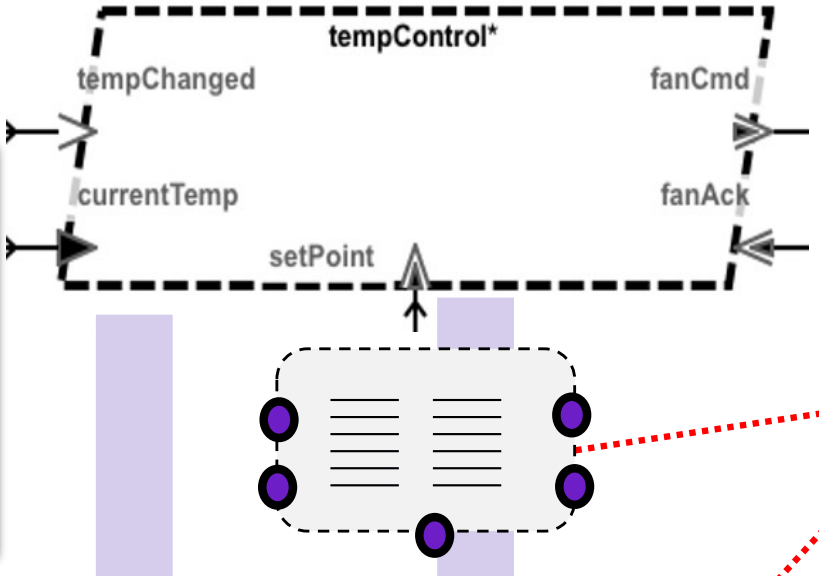
CAmkES **configuration** specifies seL4 capabilities for partition interaction.

```
assembly {
  ...

  configuration {
    tempControl.fanCmdqueue_access = "W";
    fan.fanCmd_queue_access = "R";
    ...
  }
}
```

Ensures that info flow implied by AADL model is achieved in deployed system using the formally verified seL4 info flow controls

Application Code Insertion

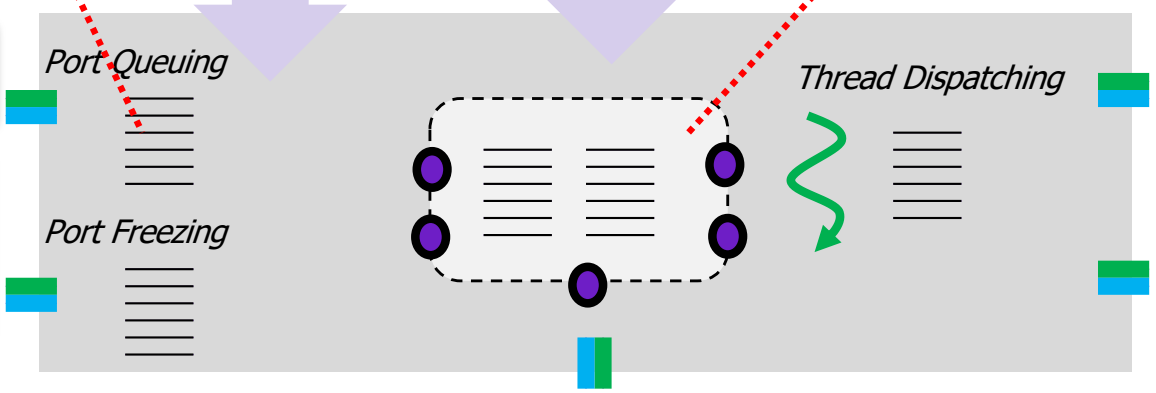
AADL with Component Application Code



HAMR generates adapter code that realizes the AADL port queuing and thread dispatch semantics in terms of CAmkES/seL4 primitives.

HAMR inserts AADL-compliant component application code into CAmkES/seL4 partition

seL4 as configured by CAmkES



Rework all of these concepts using Microkit

Potentially verify the correctness using emerging automated C verification from UNSW

***Note**: HAMR also includes an option for **only** generating the CAmkES configuration from the AADL model -- leaving the developer to do what they want with all the internals of the CAmkES component (i.e., infrastructure code for AADL semantics is not included)*

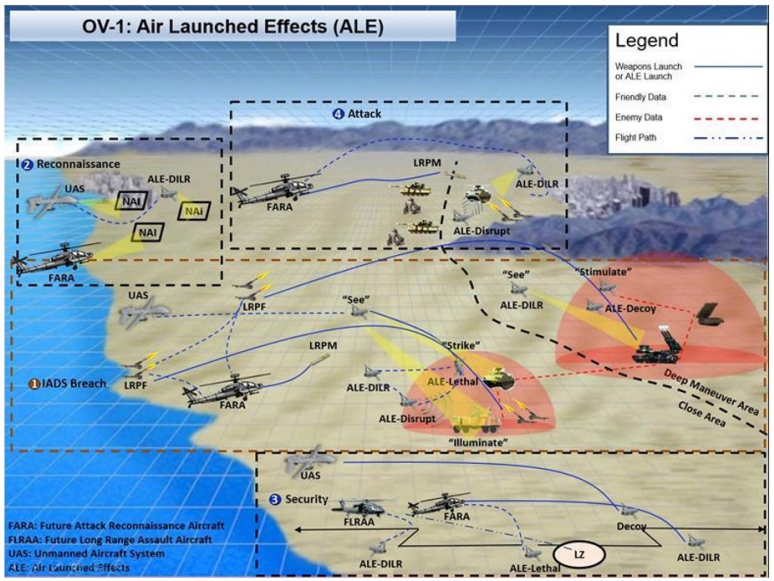
Collins DARPA PROVERS Demonstration Target

Mission computers for tube-launched UAVs…

TA2 : PLATFORM DEVELOPMENT

U.S. Army Air Launched effects (ALE)

- Family of Systems (FoS) consisting of a tube-launched air vehicle, payload(s), mission system applications, and associated support equipment to autonomously or semi-autonomously deliver effects as a single agent or as a member of a team
- ALE extends tactical and operational reach and lethality of manned assets, allowing them to remain outside of the range of enemy sensors and weapon systems while delivering kinetic and non-kinetic, lethal and non-lethal mission effects
- Relatively low-cost systems, attritable or optionally recoverable, allow rapid integration of new technologies.



https://www.flightglobal.com/military-uavs/us-army-outlines-recon-and-electronic-warfare-missions-for-air-launched-effects/139780.article

Collins Aerospace

Conclusion

- HAMR provides *model-based development for high-assurance applications* deployed on seL4 (and others)
- Provides workflow-integrated *systems engineering* by using industry standard modeling languages
- The tool chain is infused with developer-friendly formal methods at both model and code levels
- To achieve better adoption of these techniques, we are working with AADL SAE and SysMLv2 committees to migrate previous DoD-funded capabilities into SysMLv2

Resources

Sireum HAMR

High Assurance Model-based Rapid Engineering of Embedded Systems

Publicly available tool:
http://hamr.sireum.org

Resources on HAMR web site

- Distribution available for Windows, Linux, and Mac (also virtualized) hamr.sireum.org
- Documentation, examples, and tutorial material for HAMR
- Educational resources -- slides, recorded lectures, and guided exercises for HAMR Slang back end

- Online textbook for Slang/Logika available later this fall

AADL / SysMLv2 Integration
OMG Standards Work

Overarching goals

SMC

To develop modeling guidelines and a library for expressing AADLv2.3 using KerML/SysMLv2

Cover each chapter of SAE AADLv2.3 standard document

- Ensure coverage of the standard
- Listing incompatibilities if any

Goal: capture AADLv2 semantics

- SysMLv2 typing to enforce AADL rules, with caveat
- Strict adherence to AADL rules can be lifted to adhere to SysMLv2 idioms

Note: RTESC WG will interact with Semantics and Execution WG to a) check correctness of the approach, b) discuss issues with syntax or semantics