HAMR -
High-Assurance Modeling and Rapid Engineering for Embedded Systems Using AADL

HCSS 2021 – May 6, 2021

Kansas State University

John Hatcliff
Robby
Jason Belt

Adventium Labs

Todd Carpenter
John Shackleton

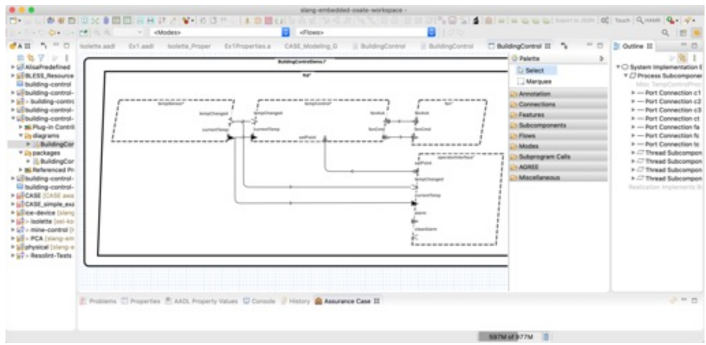
Collins Aerospace
Data61

This material is based on research sponsored in part by DARPA, US Army, AFRL, and the Software Engineering Institute

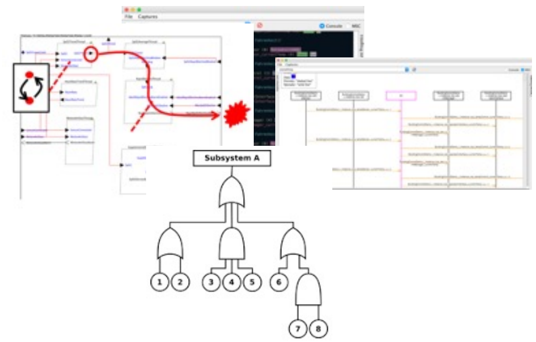
What is HAMR?

HAMR is a model-driven development tool-chain for high-assurance embedded systems

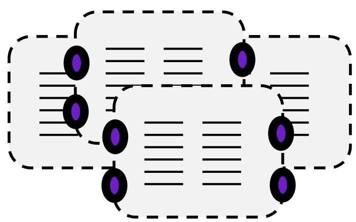
Modeling, analysis, and verification in the AADL modeling language



Leveraging analysis from AADL community



Component development and verification in multiple languages



- C
- Slang (safety-critical subset of Scala with a contract verification framework)
- CakeML (ML-variant with verified compiler)

Deployments aligned with AADL run-time on multiple platforms

JVM Deployment

JVM (Java Virtual Machine)

Linux Deployment

Linux

seL4 Deployment

seL4 Security. Performance. Proof.

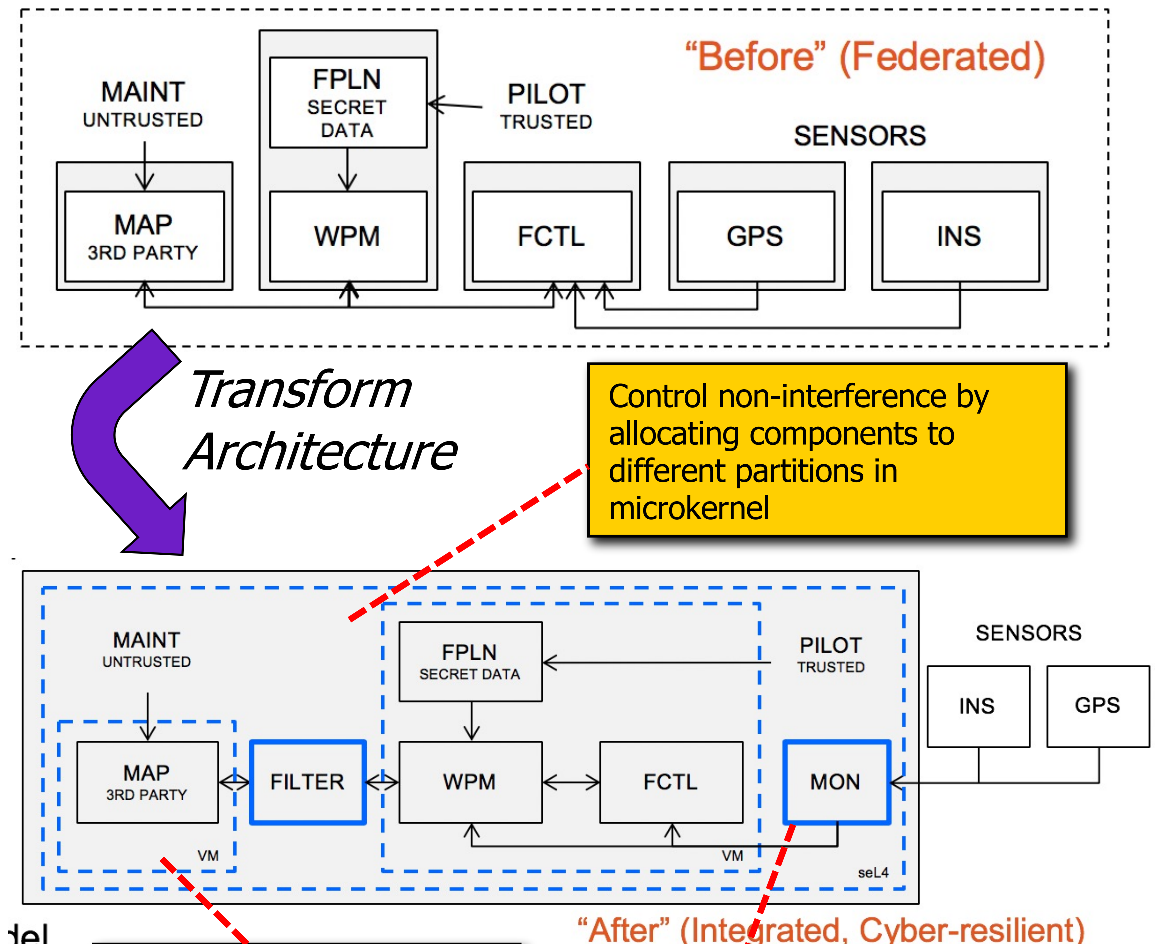
DARPA CASE Approach

HAMR is being developed by Kansas State and Adventium Labs on a team led by Collins Aerospace (Darren Cofer) that includes Data61 and University of Kansas

- **Capture requirements** for cyber-resiliency
- **Analyze** design
- **Transform** design
- **Verify new design** against requirements
- **Build / Deploy**

seL4 verified micro-kernel technology is a core technology

HAMR Focus



Control non-interference by allocating components to different partitions in microkernel

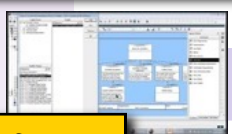
Wrap legacy untrusted component in a VM in micro-kernel partition

Insert attestation managers to ensure data is coming from a trusted source.

HAMR for seL4 Development

What does HAMR provide seL4 application developers?
A full systems engineering environment based on an standardized modeling language (AADL) with many accompanying analysis and verification tools + integration with industrial workflows

AADL-standardized "bridges" to industry workflows



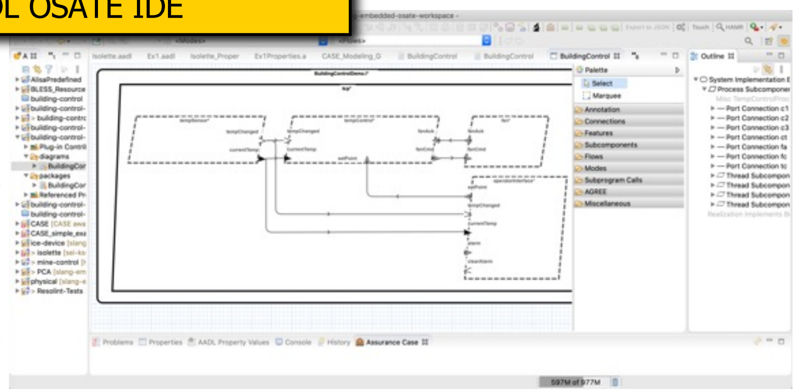
SysML

Future Airborne Capability Environment

FACE

Systems Engineering

AADL OSATE IDE

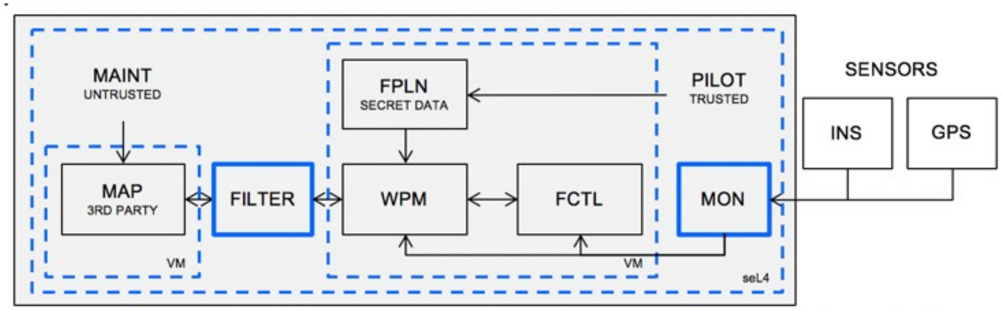


OSATE + HAMR Capabilities

Software/Hardware/Middleware Modeling
Hazard Analysis
Information Flow Analysis
Timing / Scheduleability
Component Contracts + Verification
Unit Testing
Simulation and Execution Visualization

seL4 Deployment

CAmKES Specification



del

"After" (Integrated, Cyber-resilient)

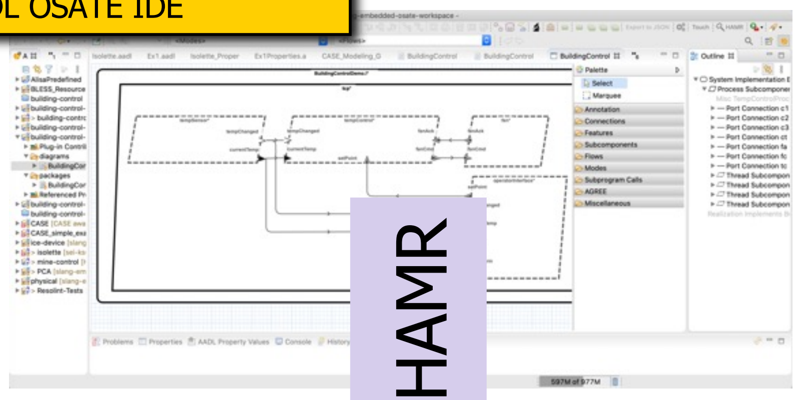
Data61 tool for component-oriented specification of seL4 partitioning and inter-partition communication

HAMR for seL4 Development

What does HAMR provide seL4 developers?
A full systems engineering environment based on an standardized modeling language (AADL) with many accompanying analysis and verification tools + integration with industrial workflows

Systems Engineering

AADL OSATE IDE

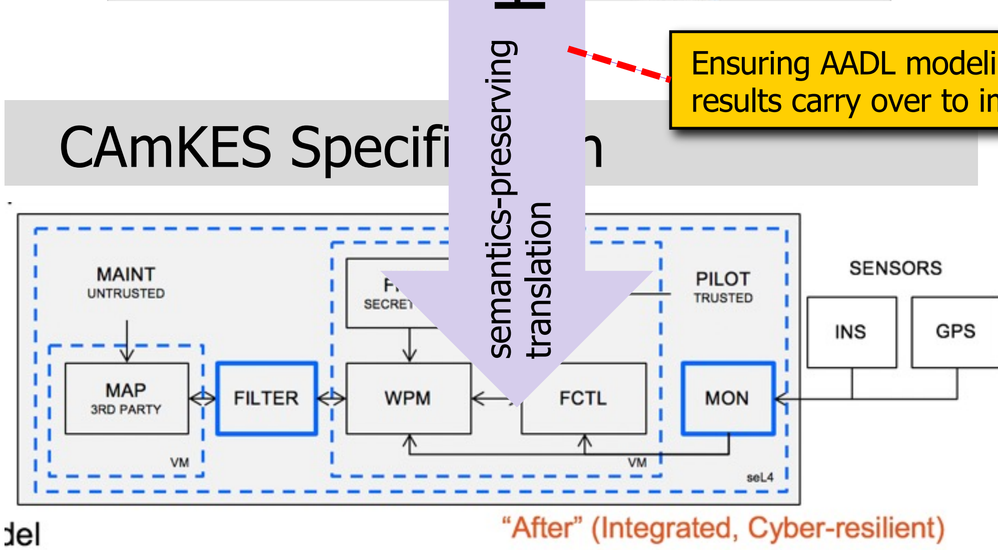


OSATE + HAMR Capabilities

Software/Hardware/Middleware Modeling
Hazard Analysis
Information Flow Analysis
Timing / Schedulability
Component Contracts + Verification
Unit Testing
Simulation and Execution Visualization

Ensuring AADL modeling, analysis, and verification results carry over to implementation

seL4 Deployment



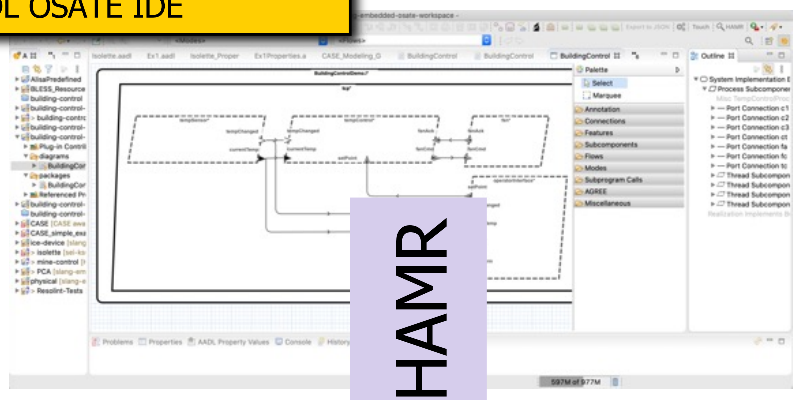
***A HAMR Goal**: Framework for building **verified applications** on top of seL4 verified infrastructure*
(DARPA, US Army, AFRL SBIRs w/ Adventium)

HAMR for seL4 Development

What does HAMR provide seL4 developers?
A full systems engineering environment based on an standardized modeling language (AADL) with many accompanying analysis and verification tools + integration with industrial workflows

Systems Engineering

AADL OSATE IDE

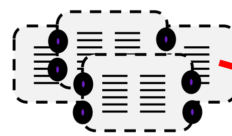


OSATE + HAMR Capabilities

- Software/Hardware/Middleware Modeling
- Hazard Analysis
- Information Flow Analysis
- Timing / Scheduleability
- Component Contracts + Verification
- Unit Testing
- Simulation and Execution Visualization

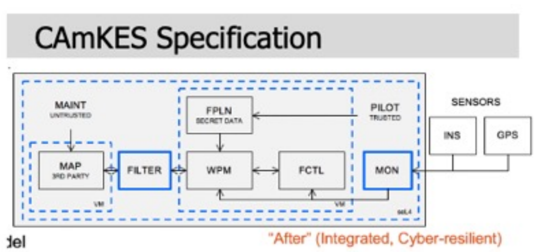
HAMR

semantics-preserving translation

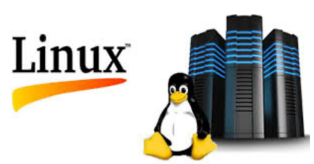


Component application logic programmed in **C** or **Slang** (Scala subset and compilable to C)

seL4 Deployment



Linux Deployment



JVM Deployment

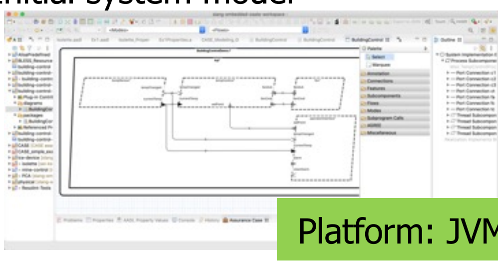
JVM (Java Virtual Machine)

Example HAMR Multiple-Platform Workflow on DARPA CASE

Rapid prototyping / agile development progression to seL4 deployment

Workflow

Initial system model

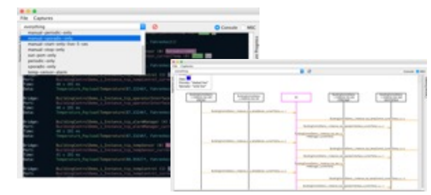


Platform: JVM

Slang mock-up of component functions

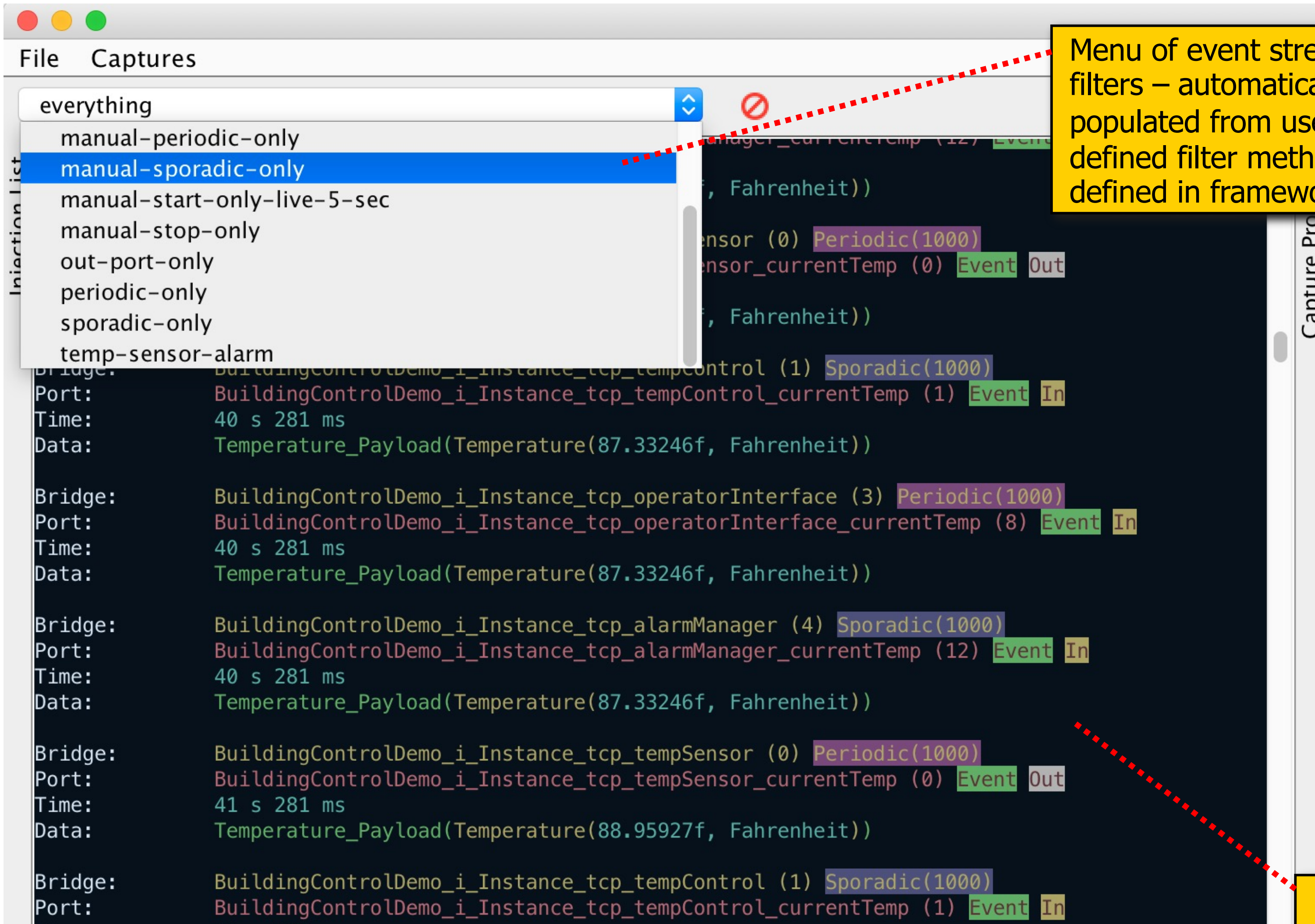
- Initial system model
- System message types designed
- Integration planned
- Component unit testing

HAMR JVM Simulation and Visualization



HAMR JVM Event Stream Filtering

HAMR JVM platform enables very flexible filtering and visualization of inter-component communication, with the ability to filter on different catagories of messages, ports, components, etc.

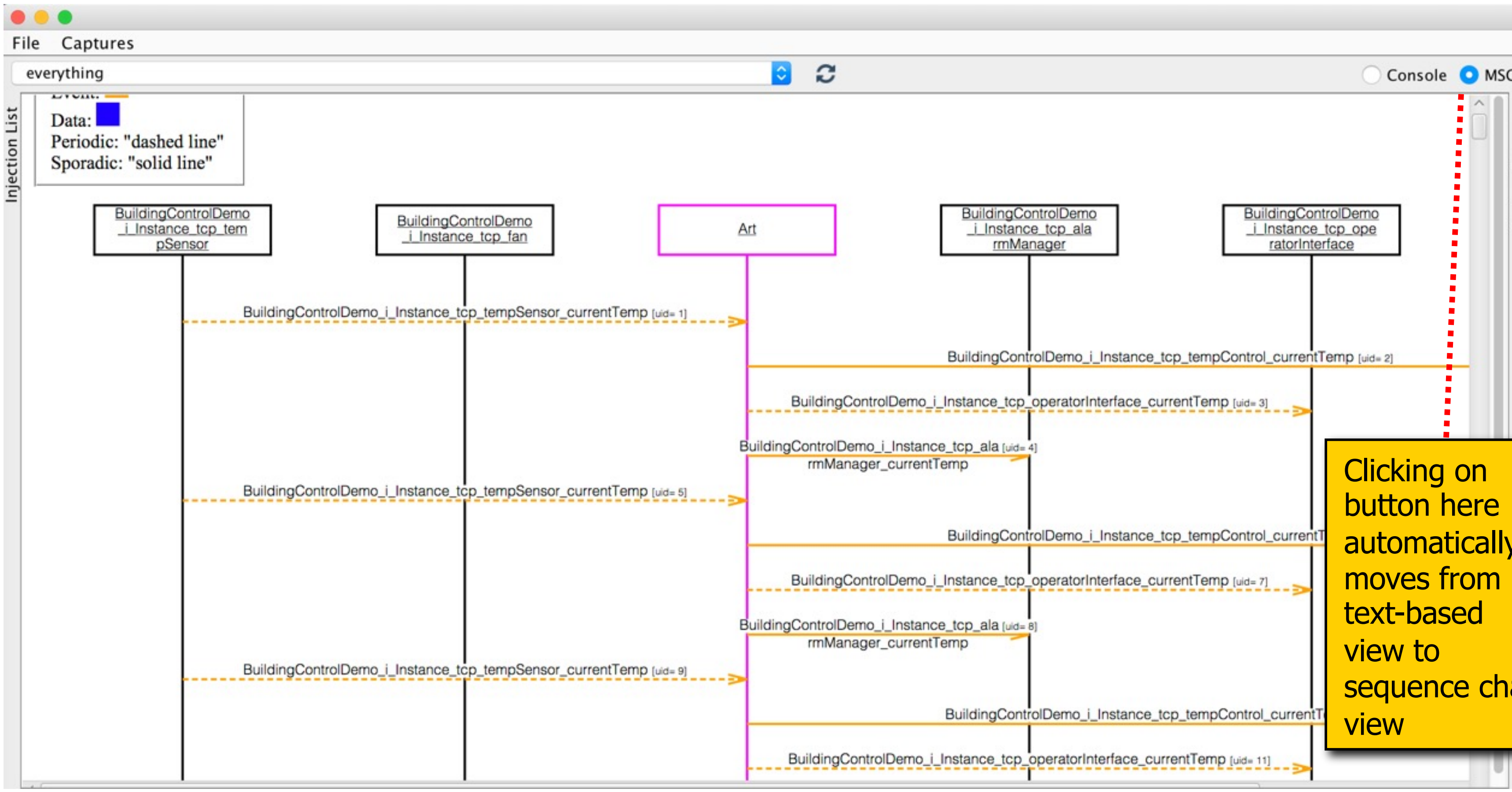


Menu of event stream filters – automatically populated from user-defined filter methods defined in framework

Event stream

HAMR JVM Execution Visualizations

HAMR JVM provides ability to visualize system execution – below, dynamically generated/updated message sequence charts (filterable) of inter-component communication



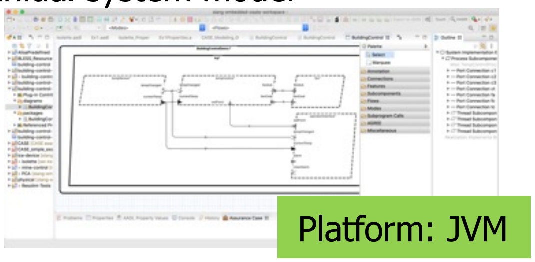
Clicking on button here automatically moves from text-based view to sequence chart view

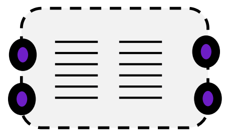
Example HAMR Multiple-Platform Workflow on DARPA CASE

Rapid prototyping / agile development progression to seL4 deployment

Workflow

Initial system model

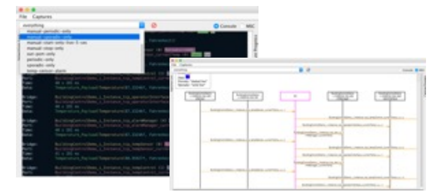


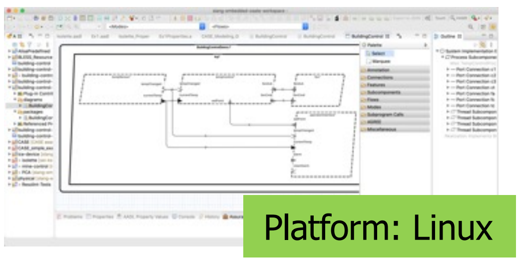


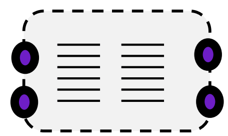
Slang mock-up of component functions

- Initial system model
- System message types designed
- Integration planned
- Component unit testing

HAMR JVM Simulation and Visualization

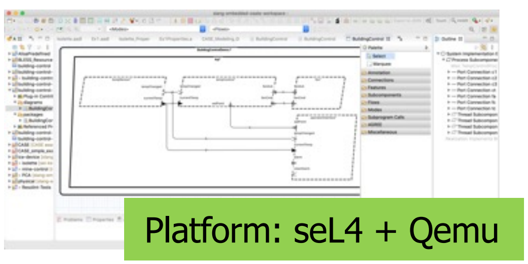


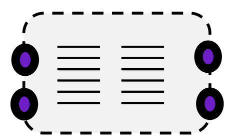




C coding of component functions (hand-written or translated from Slang)

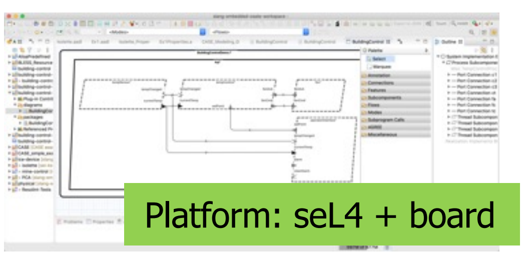
- Testing of C components
- Mock up of VM functions
- Initial System Testing

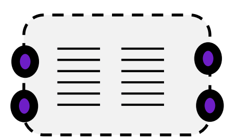




C components

- Design of domain schedule
- Integration of VMs
- Enhanced system testing





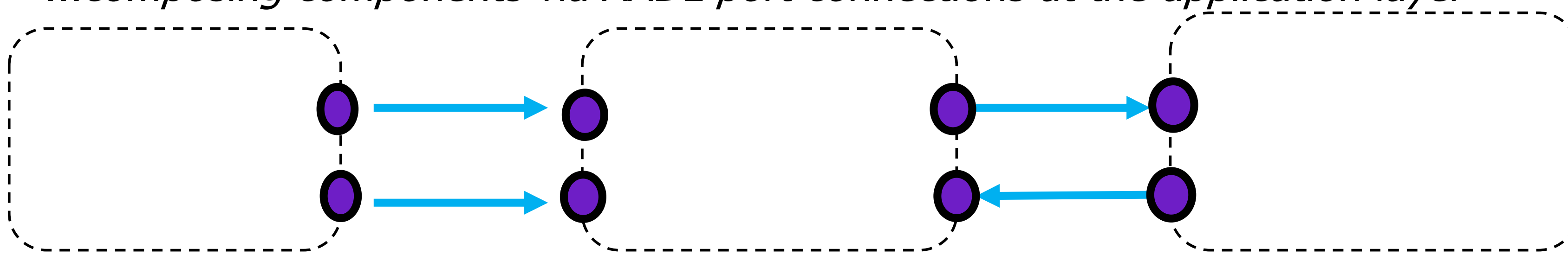
C components + drivers, communication stack

- Integration of drivers
- Final system testing
- Penetration testing

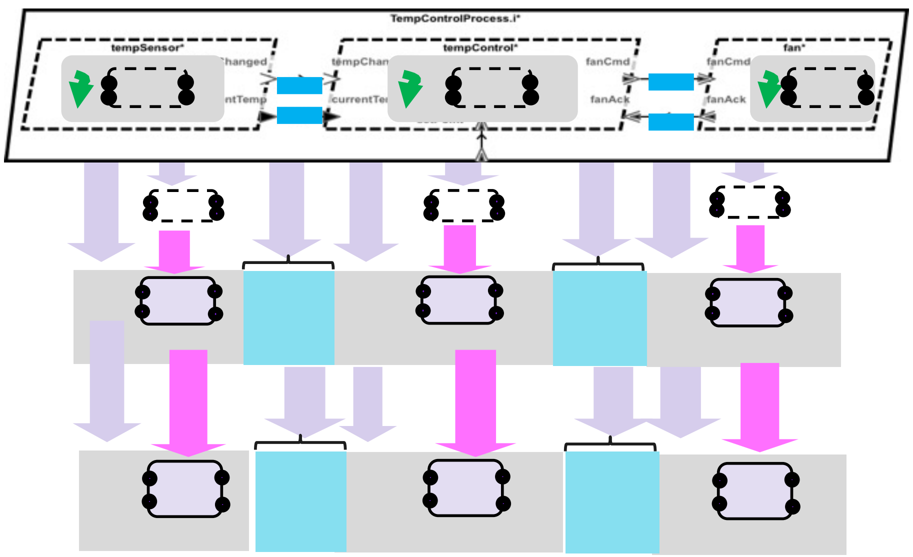
Notions of Composition

Horizontal Composition

...composing components via AADL port connections at the application layer



Vertical Composition

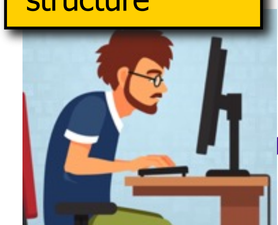


...composing multiple layers of abstractions for threading and communication (refinements)

...ensuring consistency in threading and communication structure to ensure correctness and enable horizontal composition at lower levels

AADL Computational Model

Developer configures computational structure



AADL Thread Property Options

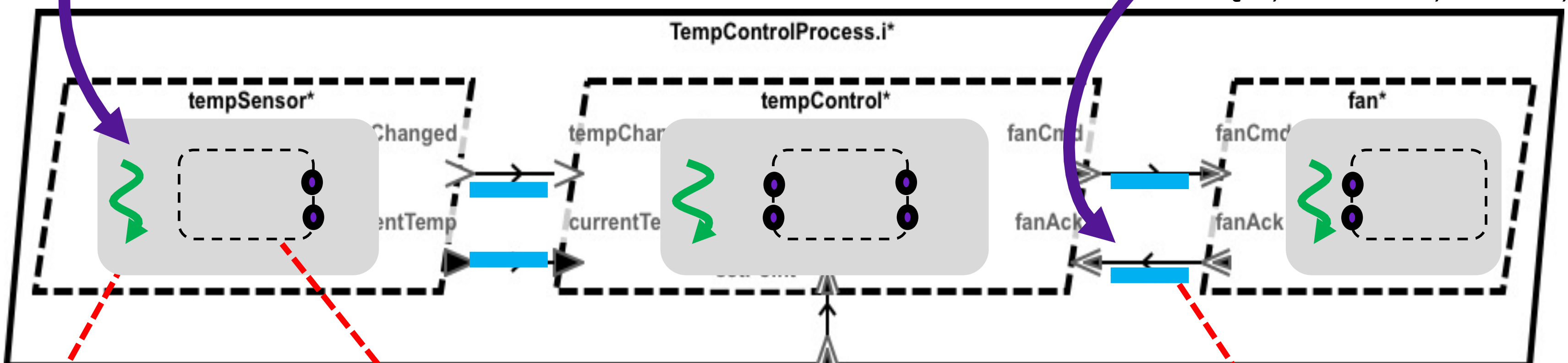
Periodic
Sporadic
Hybrid
…

+ timing, scheduling constraints, etc.

AADL Port & Connection Property Options

Event	*buffered notifications*
Data	*shared data cells (or data distribution service)*
Event Data	*buffered messages (message passing middleware)*
…	

+ QoS, buffer sizes, latencies, etc

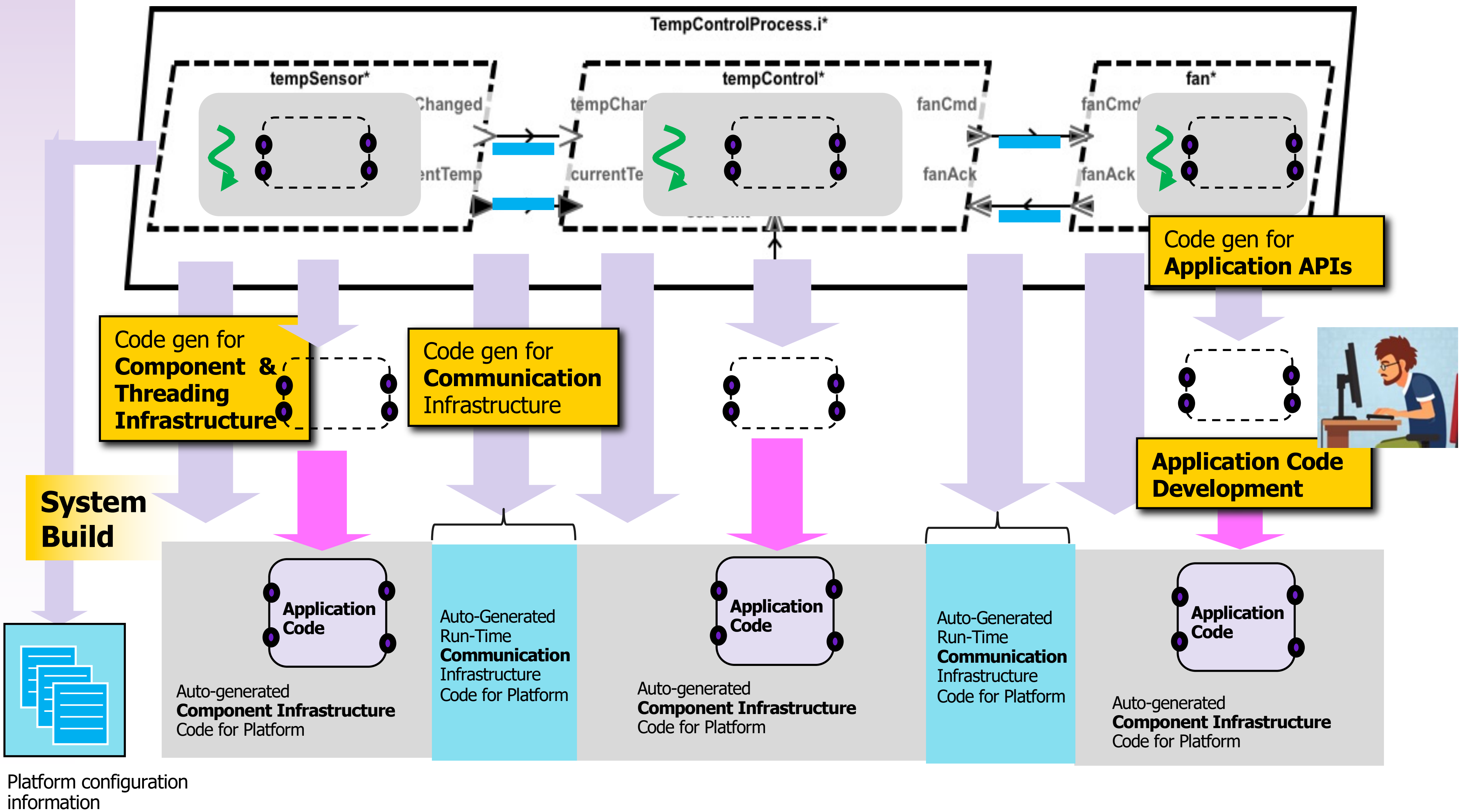


Selected thread pattern

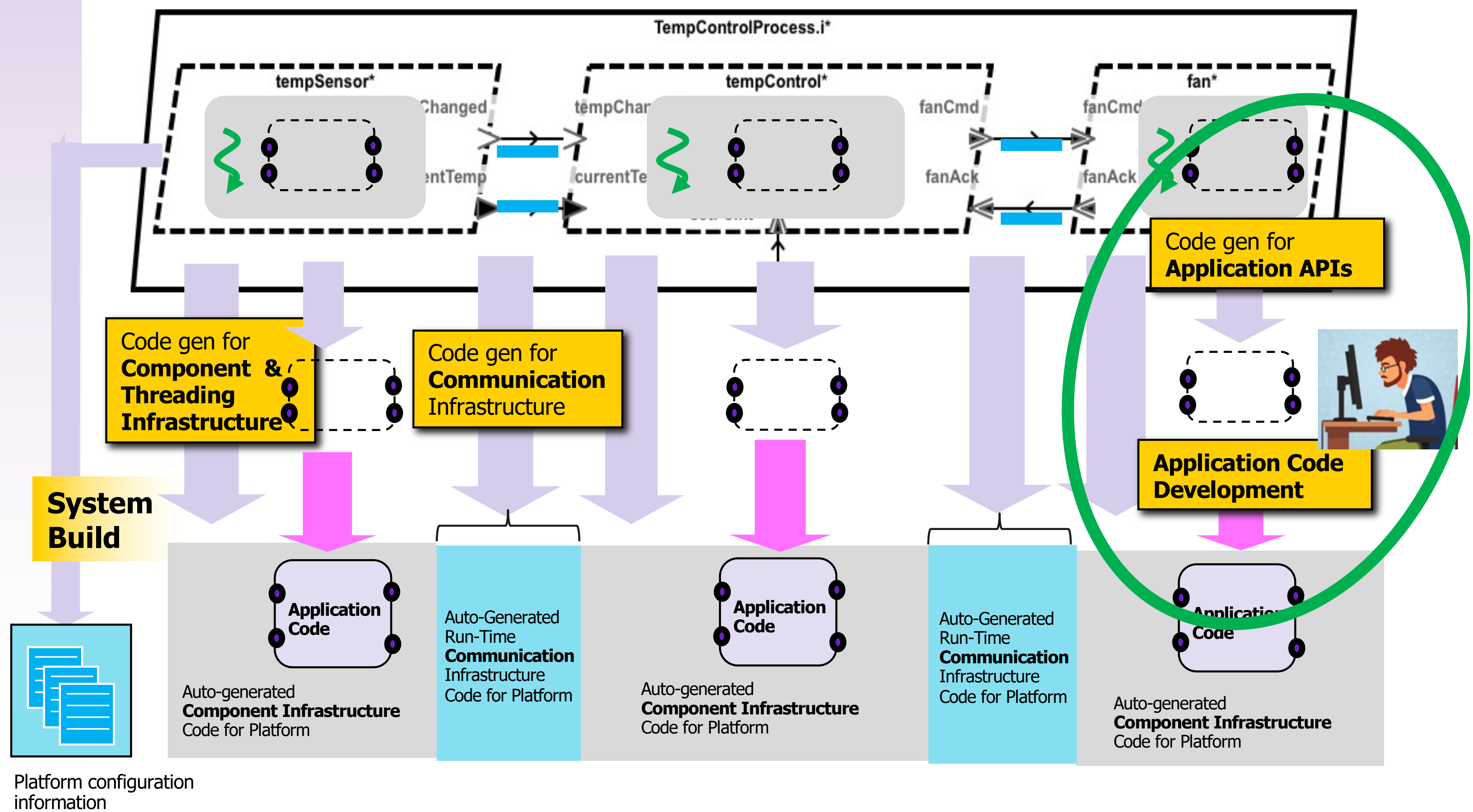
Implied API Pattern for application code to access port communication, etc.

Selected communication pattern

HAMR Code Generation

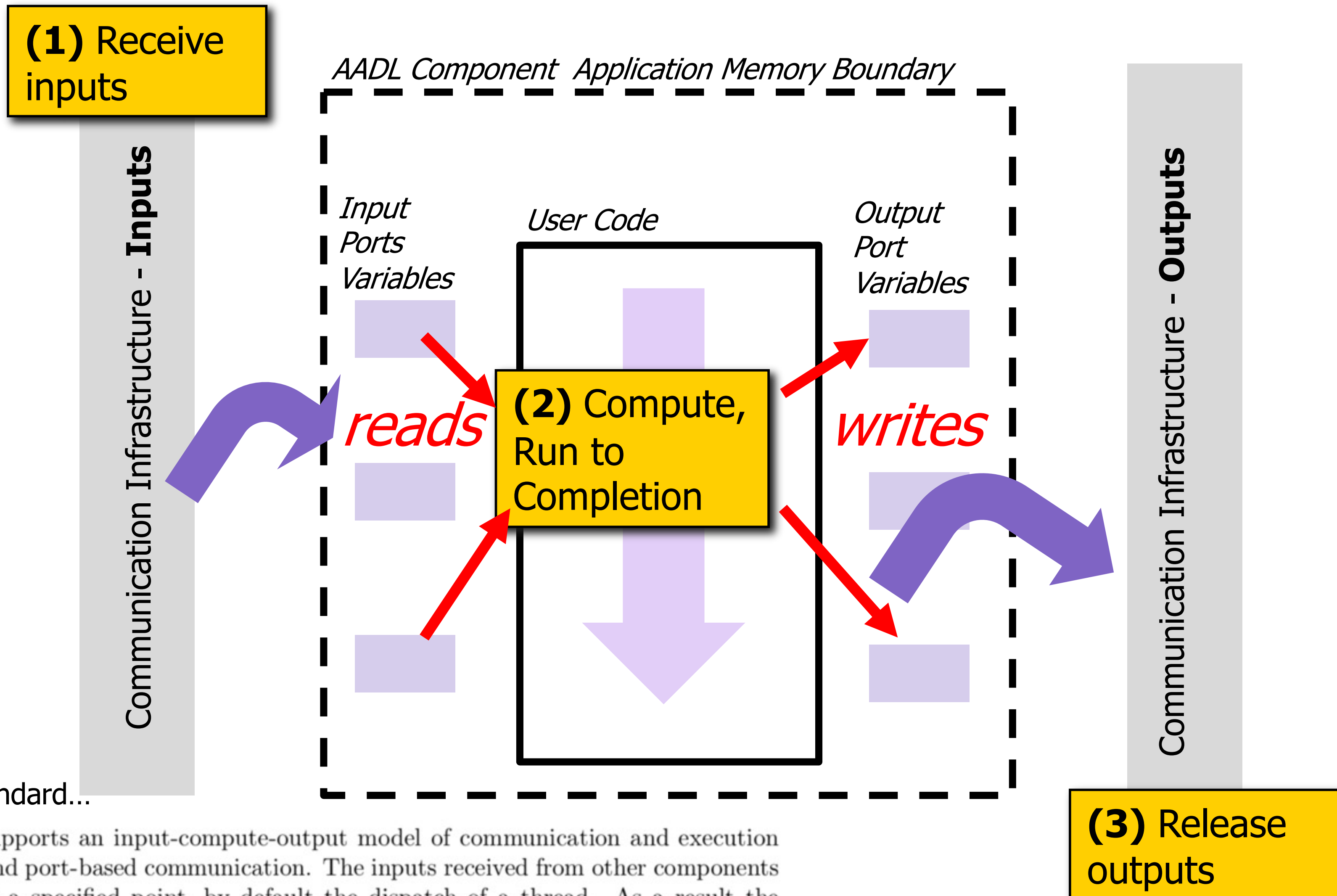


HAMR Code Generation



Port Semantics and Thread Execution

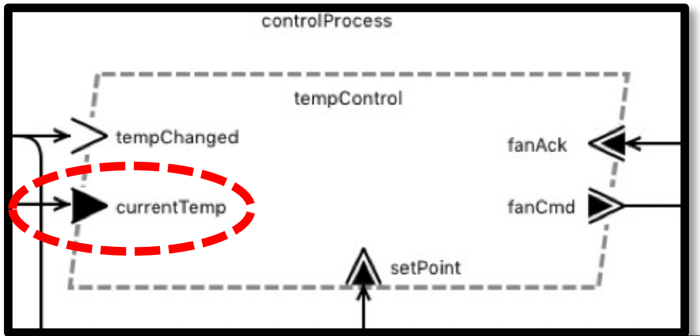
On each dispatch, AADL threads follow a well-known **input-compute-output** pattern for real-time tasks that aid analysis and verification…



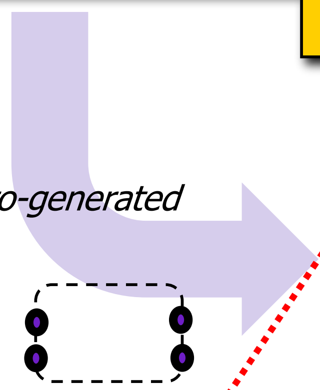
From AADL standard…

(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

Platform-Independent Port APIs Generated from AADL Model



...For each port, HAMR generates an API for communicating over that port. These link to auto-generated implementations of port communication for the chosen platform



auto-generated

APIs for Port Communication

Reference parameter for receiving port value.

```
bool api_get_currentTemp(Temp value);

bool api_get_fanAck(FanAck_Type *value);

bool api_get_setPoint(SetPoint value);

void api_send_fanCmd(FanCmd_Type value);

bool api_get_tempChanged();
```

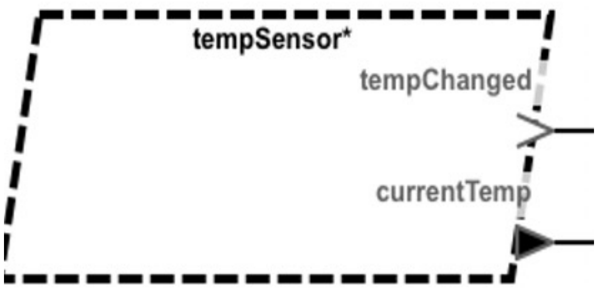
Status value indicating if operation succeeded

Implementations for Port Communication APIs for target platform

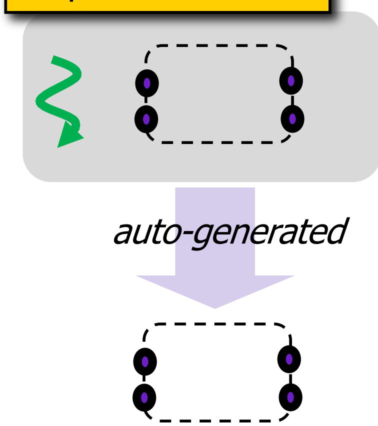
Linux

seL4 *Uses CAmkES + HAMR-generated "glue code"*

Platform-Independent Thread Structure Generated from AADL Model - *Periodic*



AADL Model Implied Semantics

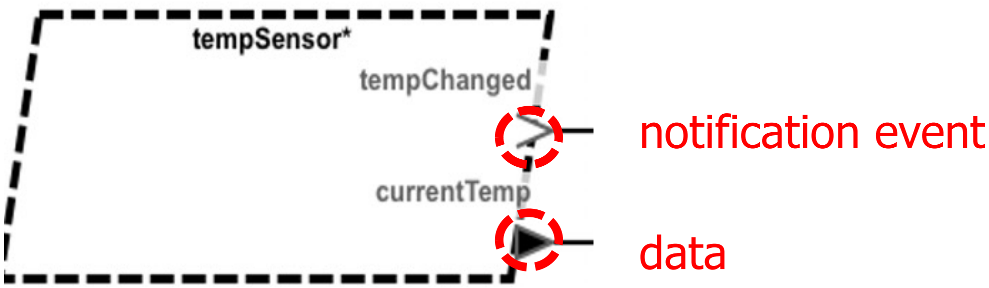


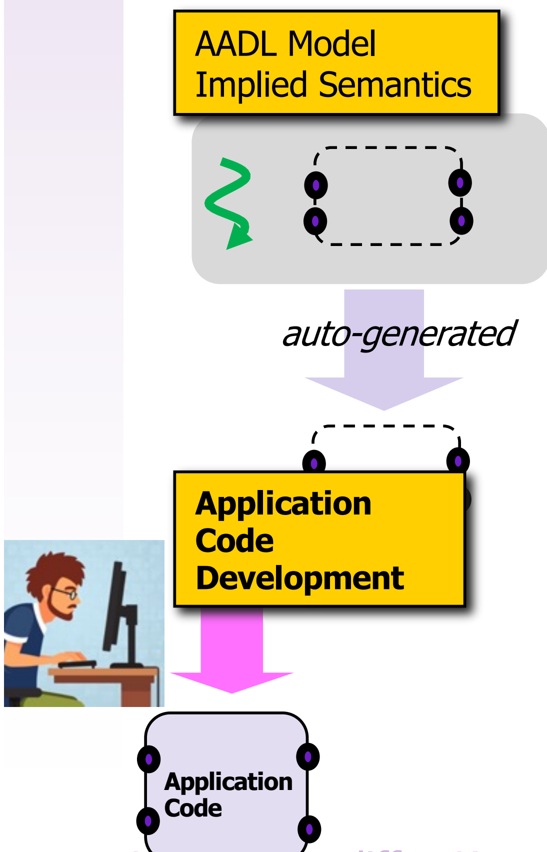
Application Code Skeleton in C

***Periodic** thread – skeleton for application logic*

```
Unit timeTriggered() {
   …
}
```

Platform-Independent Thread Structure Generated from AADL Model - *Periodic*





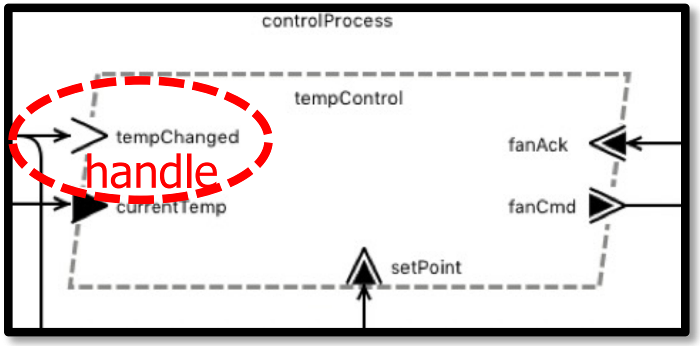
***Periodic** thread – skeleton filled with application logic*

```
Unit timeTriggered() {
  DeclNew_Temperature(currTemp);

  // read current temp from hardware sensor
  senseTemperature(&currTemp);

  // take action if temperature has changed
  if(lastTemperature.degrees != currTemp.degrees) {
    lastTemperature = currTemp;
    api_send_currentTemp(&lastTemperature);
    api_send_tempChanged();
  }
}
```

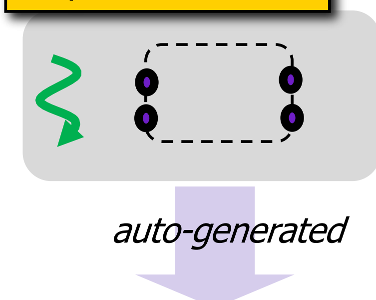
Put new temp value on output data port

Notify consumers that temperature changed

Platform-Independent Thread Structure Generated from AADL Model - *Sporadic*



AADL Model
Implied Semantics

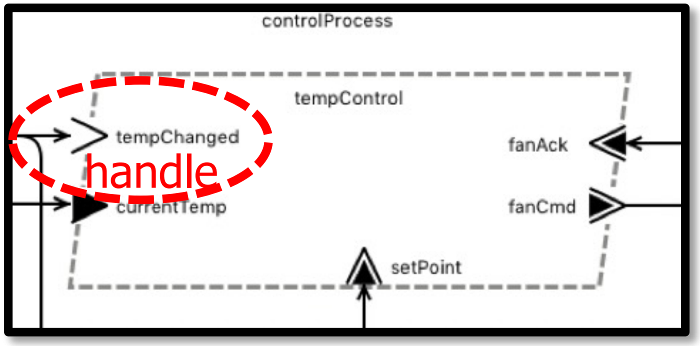


Application Code
Skeleton in C

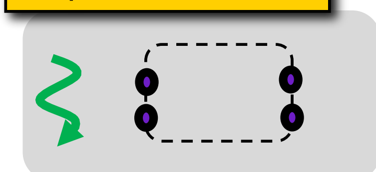
Sporadic** thread – skeleton for application logic – **event handlers

```
void TempControl_tempChanged(void) {
  …

}
```

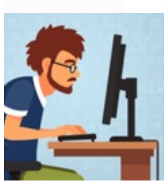
Platform-Independent Thread Structure Generated from AADL Model - *Sporadic*



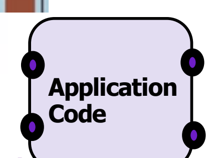
AADL Model Implied Semantics



auto-generated



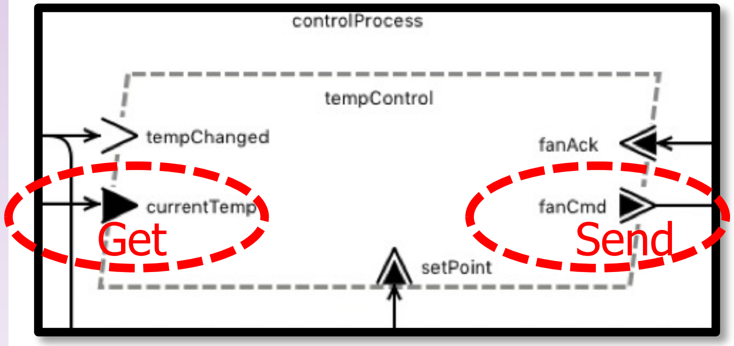
Application Code Development



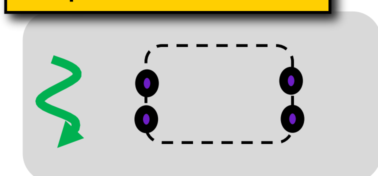
Sporadic** thread – skeleton for application logic – **event handlers

```
void tempChanged(void) {
 DeclTemp(currTemp); // macro to allocate Temp in stack

 if (api_get_currentTemp(&currTemp)){
  struct Temp currTempInF = convertToF(&currTemp);

  if (currTempInF.degrees > setPoint.high.degrees) {
   api_send_fanCmd(FanCmd_On);
  } else
  if (currTempInF.degrees < setPoint.low.degrees) {
   api_send_fanCmd(FanCmd_Off);
  }
 }
}
```

Platform-Independent Thread Structure Generated from AADL Model - *Sporadic*

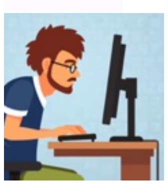


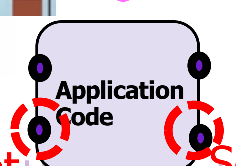
AADL Model
Implied Semantics



auto-generated

Application Code Development





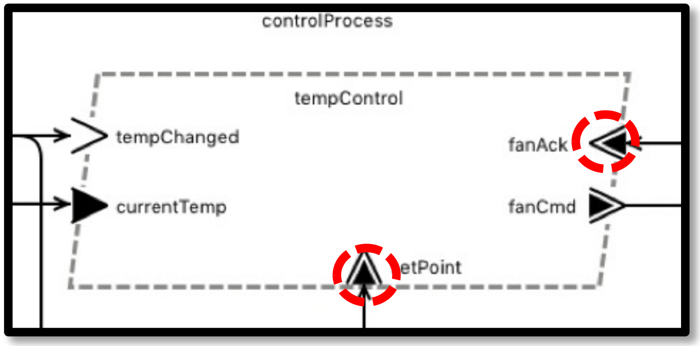
Sporadic** thread – skeleton for application logic – **event handlers

```
void tempChanged(void) {
 DeclTemp(currTemp); // macro to allocate T                    k

 if (api_get_currentTemp(&currTemp)) {
  struct Temp currTempInF = convertToF(&currTemp);

  if (currTempInF.degrees > setPoint.high.degrees) {
   api_send_fanCmd(FanCmd_On);
  } else
  if (currTempInF.degrees < setPoint.low.degrees) {
   api_send_fanCmd(FanCmd_Off);
  }
 }
}
```

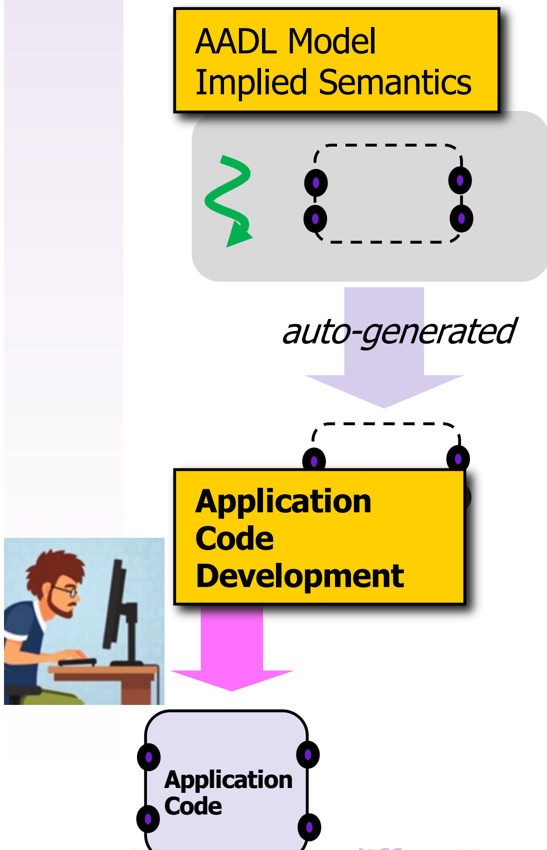
Get current temperature

Send command to fan

Platform-Independent Thread Structure Generated from AADL Model - *Sporadic*



…HAMR also generates handlers for the remaining input event data ports



Sporadic *thread – setPoint–* ***event handler***

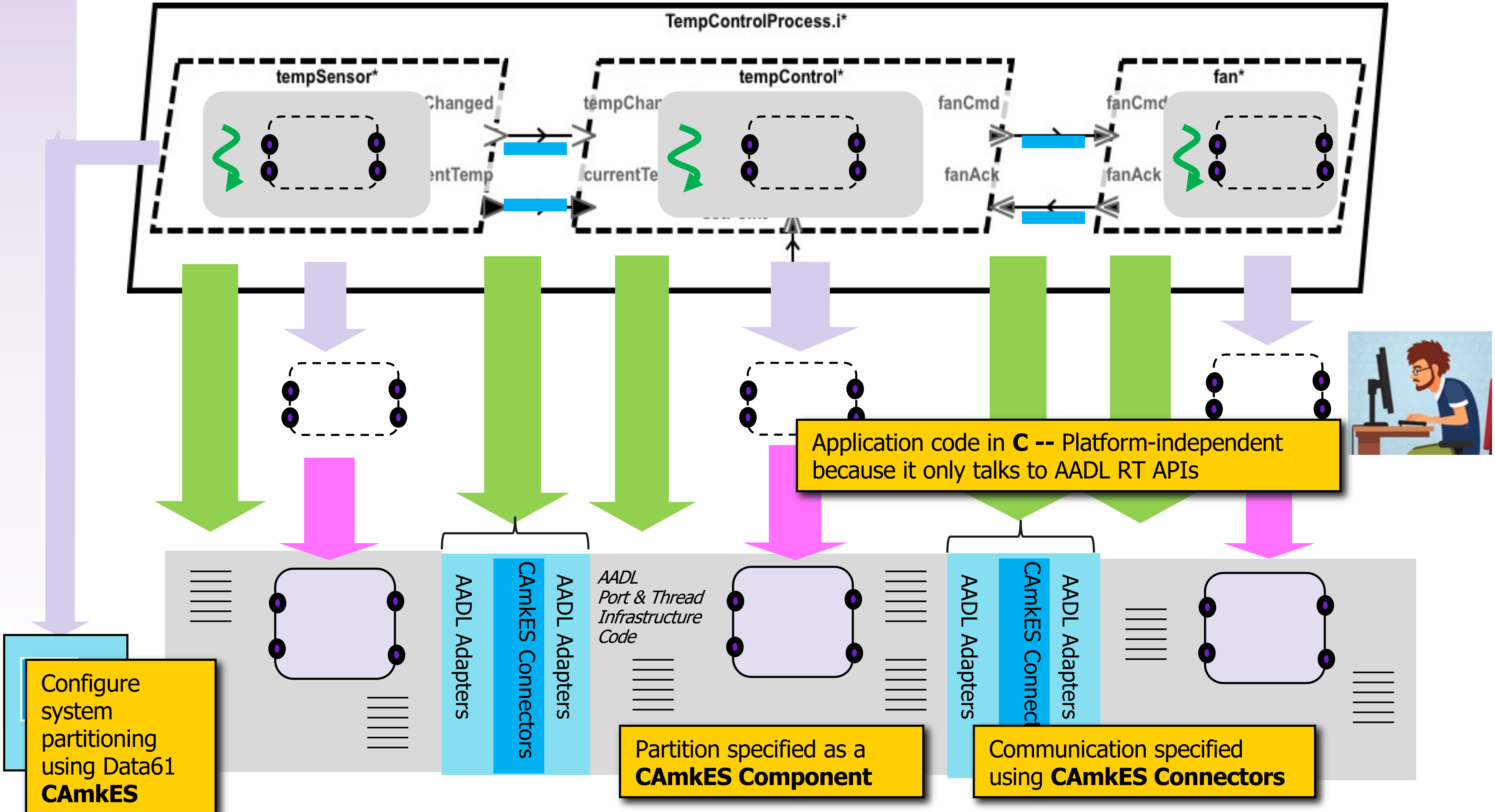
```
void handlesetPoint(SetPoint value) {
   ...
}
```

Sporadic *thread – fanAck–* ***event handler***

```
void handlefanAck(FanAck_Type value) {
   ...
}
```

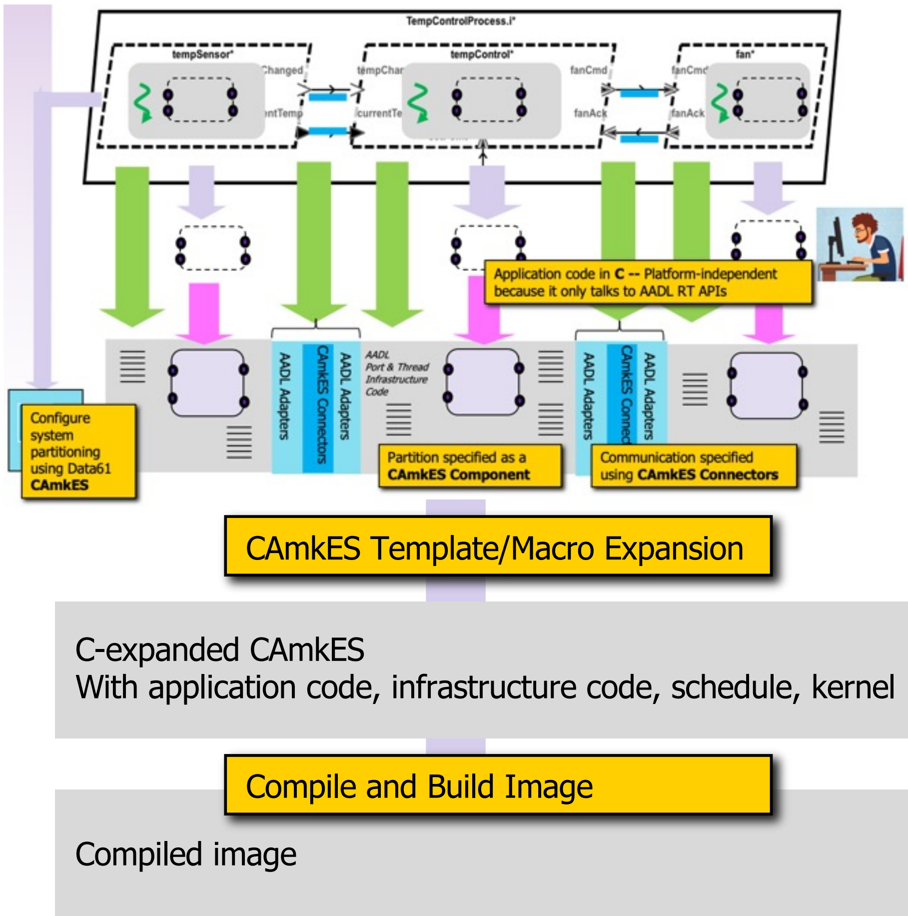
HAMR Code Generation seL4 Platform

HAMR instantiation for C-based development on **SeL4 microkernel** (e.g., DARPA CASE)



HAMR Code Generation seL4 Platform

HAMR utilizes the Data61 CAmkES framework to create the final build for deployment

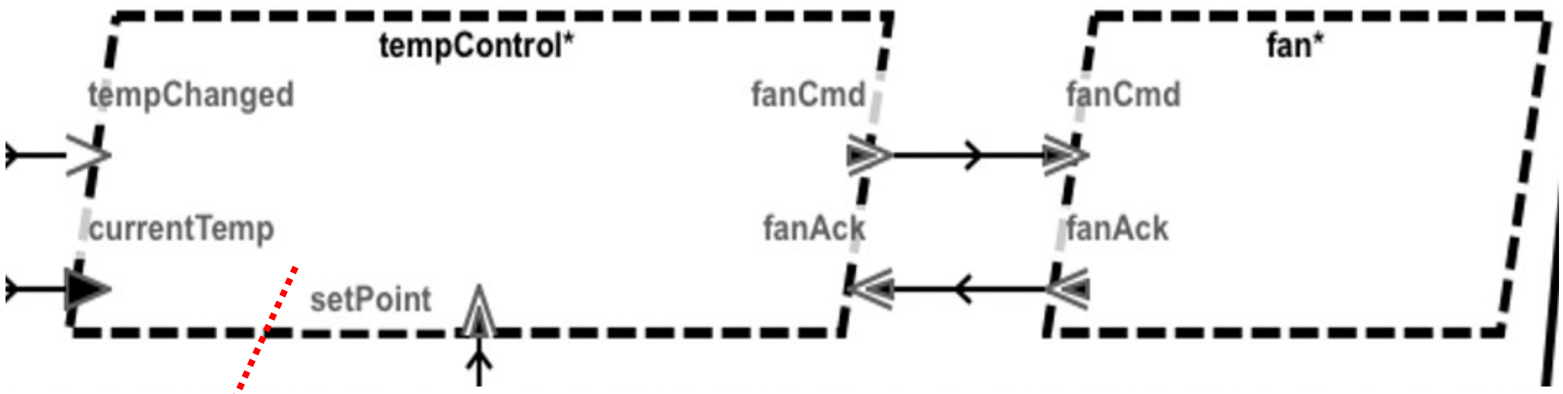


Qemu Emulator

Hardware (e.g., Odroid)

CAmkES seL4 Configuration

AADL and Platform Independent Application Code



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

```
component TempControl {
```

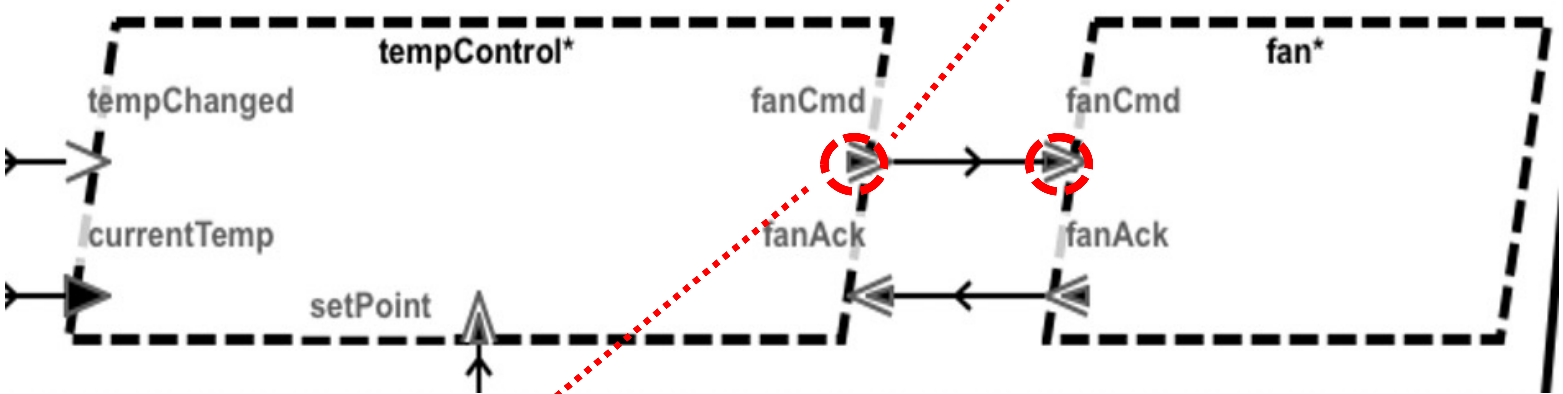
```
component Fan {

}
```

For each AADL thread component create an seL4 partition

CAmkES seL4 Config n

AADL Event Data port (message with payload)

AADL and Platform Independent Application Code



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

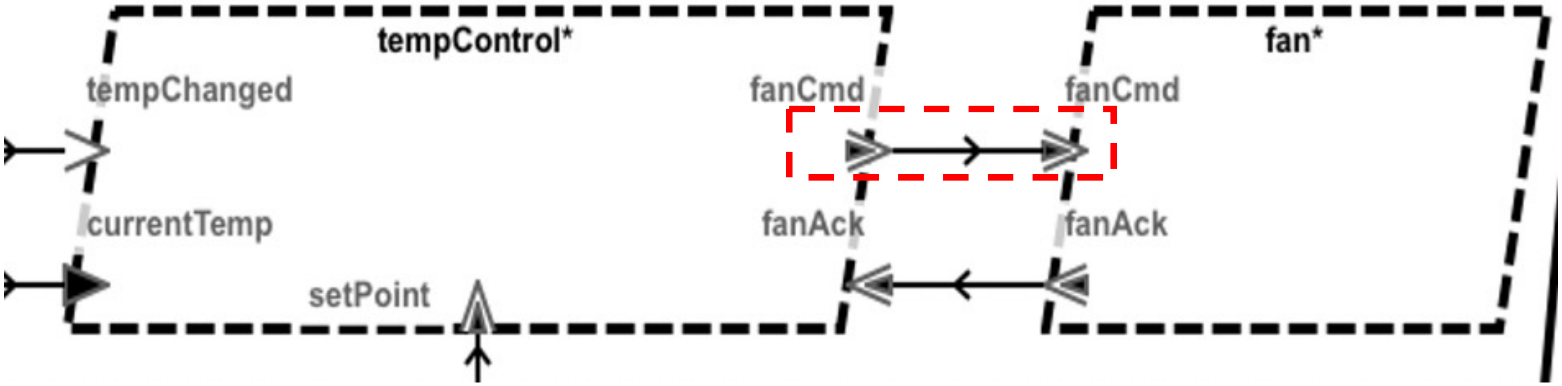
```
component TempControl {
  emits ReceiveEvent
        fanCmd_notification;
  dataport DataContent
           fanCmd_queue;
  ...
}
```

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

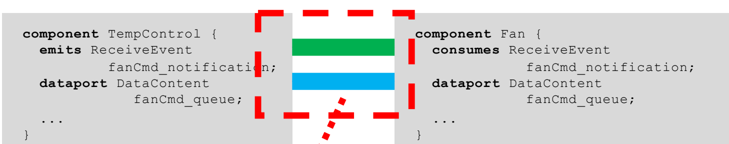
AADL Event Data port is represented using a CAmkES **notification** + **dataport** – introducing finer granularity as we move to platform

CAmkES seL4 Configuration

AADL and Platform Independent Application Code



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

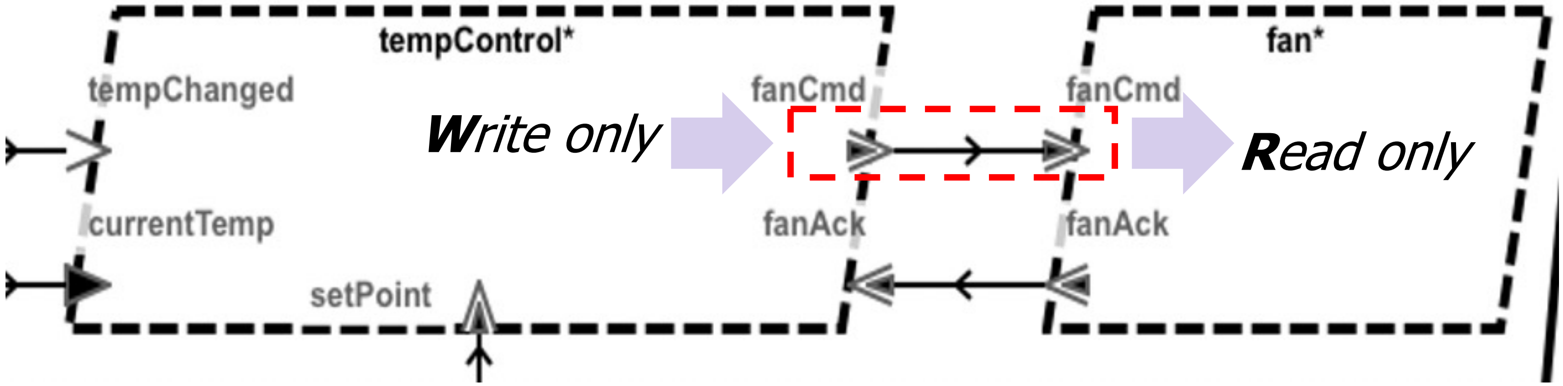


CAmkES **assembly** specifies system topology, including allowed communication between seL4 partitions

```
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)

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

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

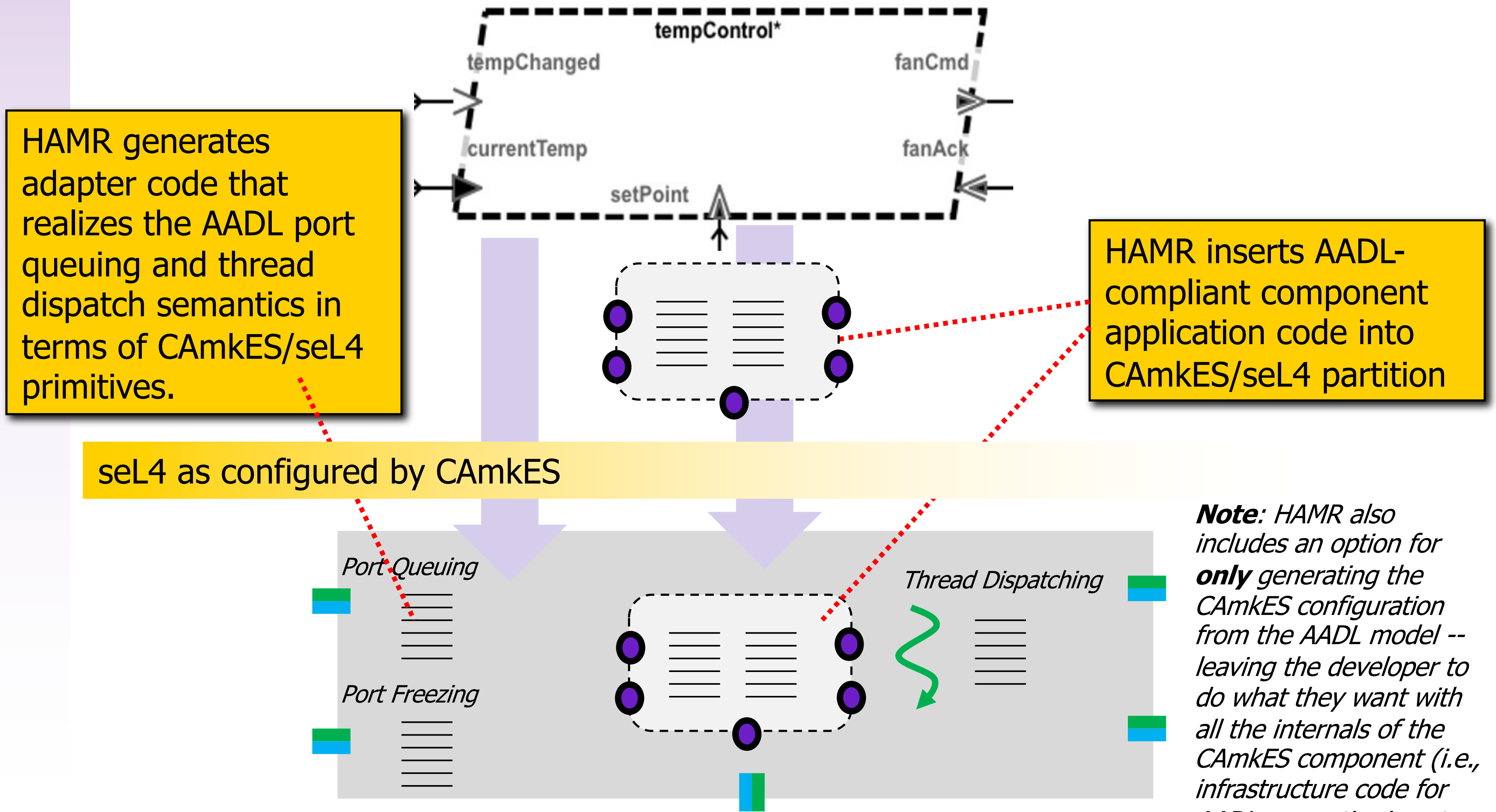
CAmkES **configuration** specifies seL4 capabilities for partition interaction.

```
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



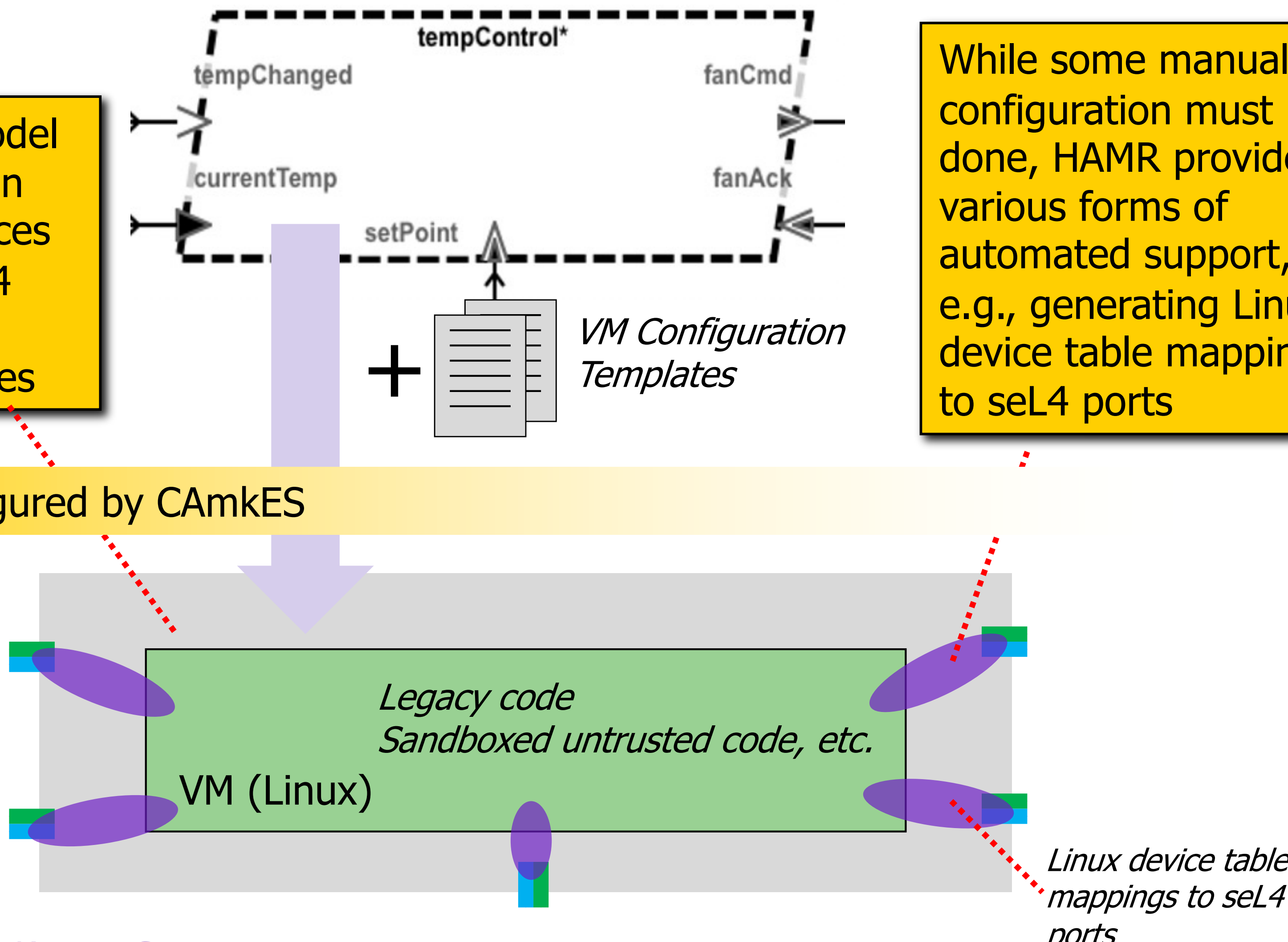
Note: *HAMR also includes an option for **only** generating the CAmkES configuration from the AADL model -- leaving the developer to do what they want with all the internals of the CAmkES component (i.e., infrastructure code for AADL semantics is not included)*

VM Insertion

HAMR provides AADL modeling and code generation to automatically insert Linux virtual machines in to CAmkES/seL4 partitions (e.g., to host legacy or non-AADL-aligned code)

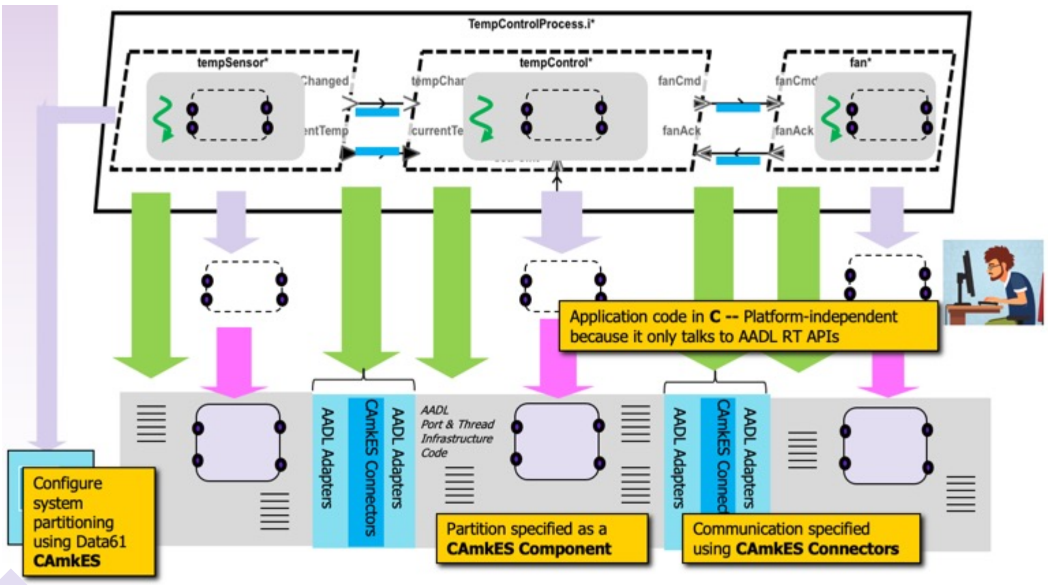
Guided by AADL model directives, HAMR can generate VM instances for an CAmkES/seL4 partition using pre-defined VM templates

While some manual configuration must be done, HAMR provides various forms of automated support, e.g., generating Linux device table mappings to seL4 ports



Static Schedule via Domain Schedule

As a step towards the eventual use of seL4 MCS extensions, DARPA CASE is using the seL4 *domain schedule* to enforce temporal partitioning (see Todd Carpenter's talk in Assured Systems II session)



Domain Schedule

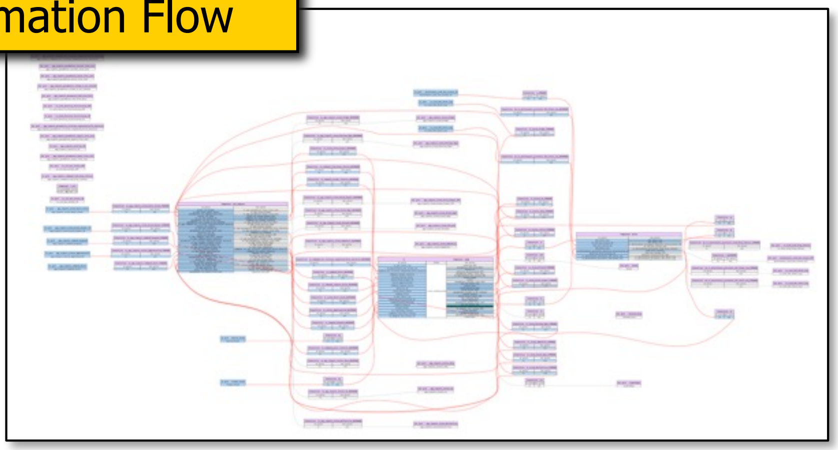
Part of Platform Configuration information

```
const dschedule_t ksDomSchedule[] = {
  { .domain = 0, .length = 100 }, // all other threads
  { .domain = 2, .length = 5 },   // tempSensor
  { .domain = 3, .length = 5 },   // tempControl
  { .domain = 4, .length = 5 },   // fan
  { .domain = 0, .length = 380 }, // pad rest of period
};
const word_t ksDomScheduleLength =
  sizeof(ksDomSchedule) / sizeof(dschedule_t);
```

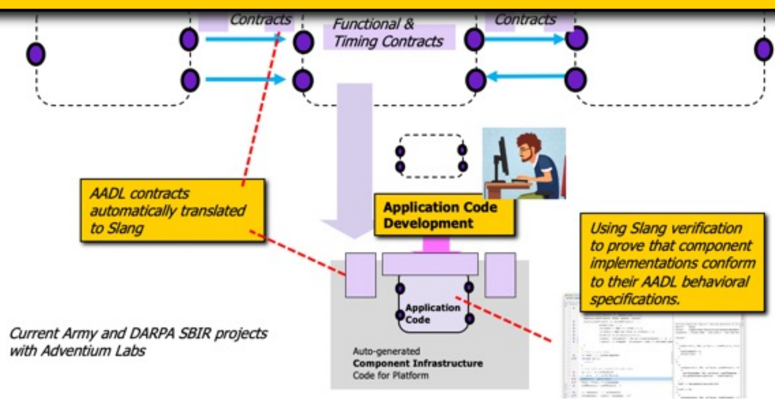
Up Next

Analyses and Verification enabled by planning for compositionality...

Information Flow

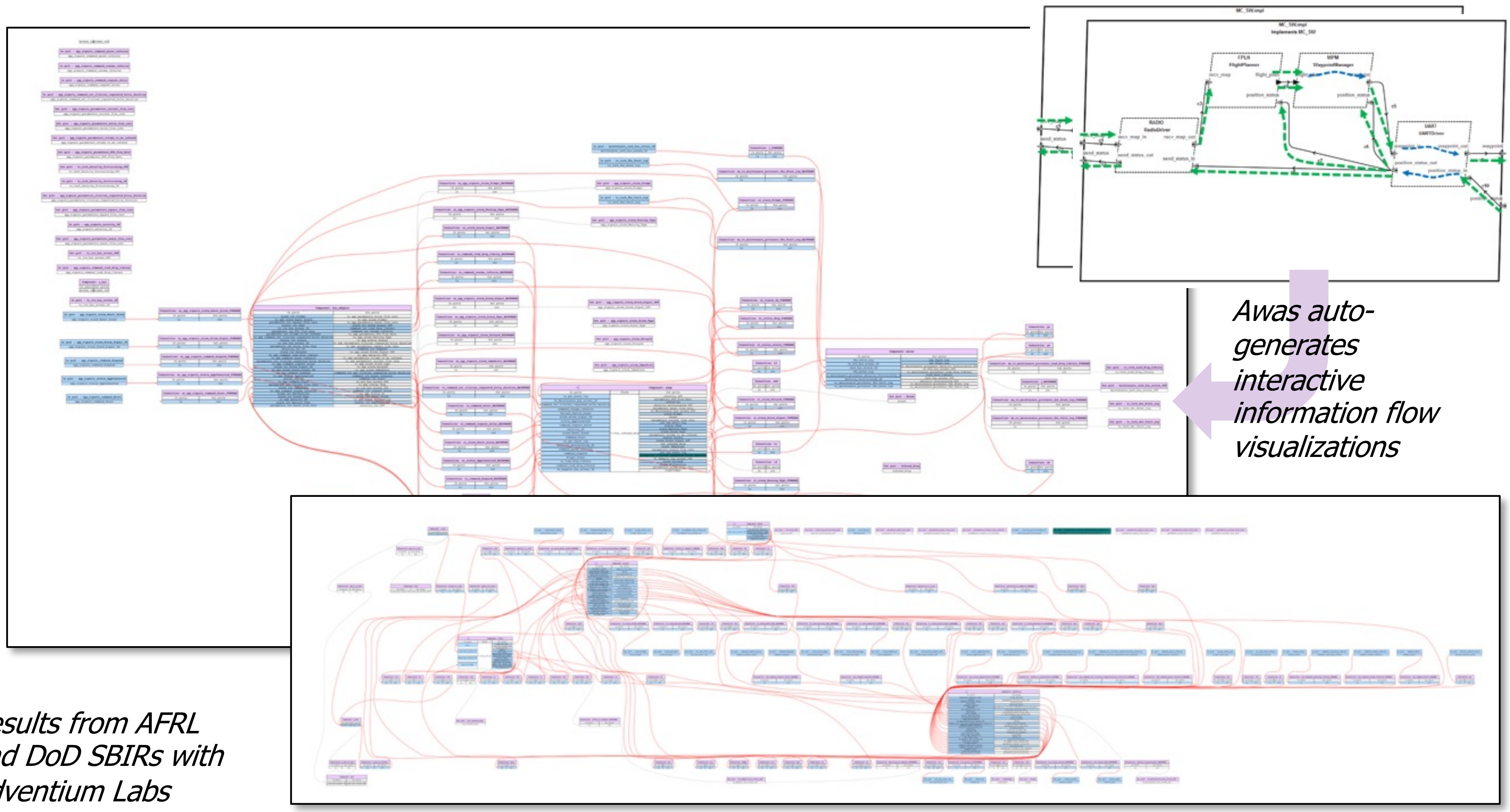


Contracts and Assume Guarantee Reasoning



AADL Analysis Example: Information Flow

The KSU Awas tool (https://awas.sireum.org) generates scalable interactive visualizations of AADL information flows and model-based hazard analysis results



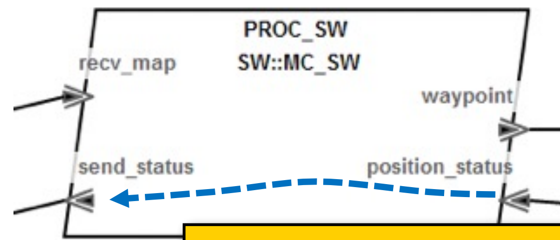
Awas auto-generates interactive information flow visualizations

Results from AFRL and DoD SBIRs with Adventium Labs

Information flow graphs can be dynamically browsed and queried with path logic.

Intra-Component Information Flow

```
process MC_SW
    features
        recv_map: in event data port Command.Impl;
        send_status: out event data port Coordinate.Impl;
        waypoint: out event data port MissionWindow.Impl;
        position_status: in event data port Coordinate.Impl;
    flows
        compute_waypoint_flight_plan: flow path recv_map -> waypoint;
        compute_waypoint_pos_status :flow path position_status -> waypoint;
        compute_status : flow path position_status -> send_status;
```

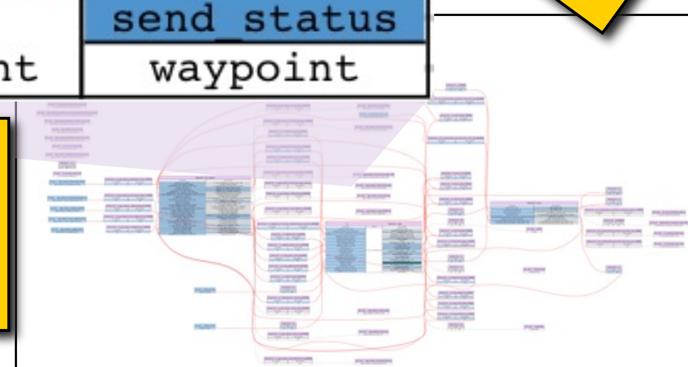


Specification of **flows** through **component application logic**

Example component node in Awas graph visualization

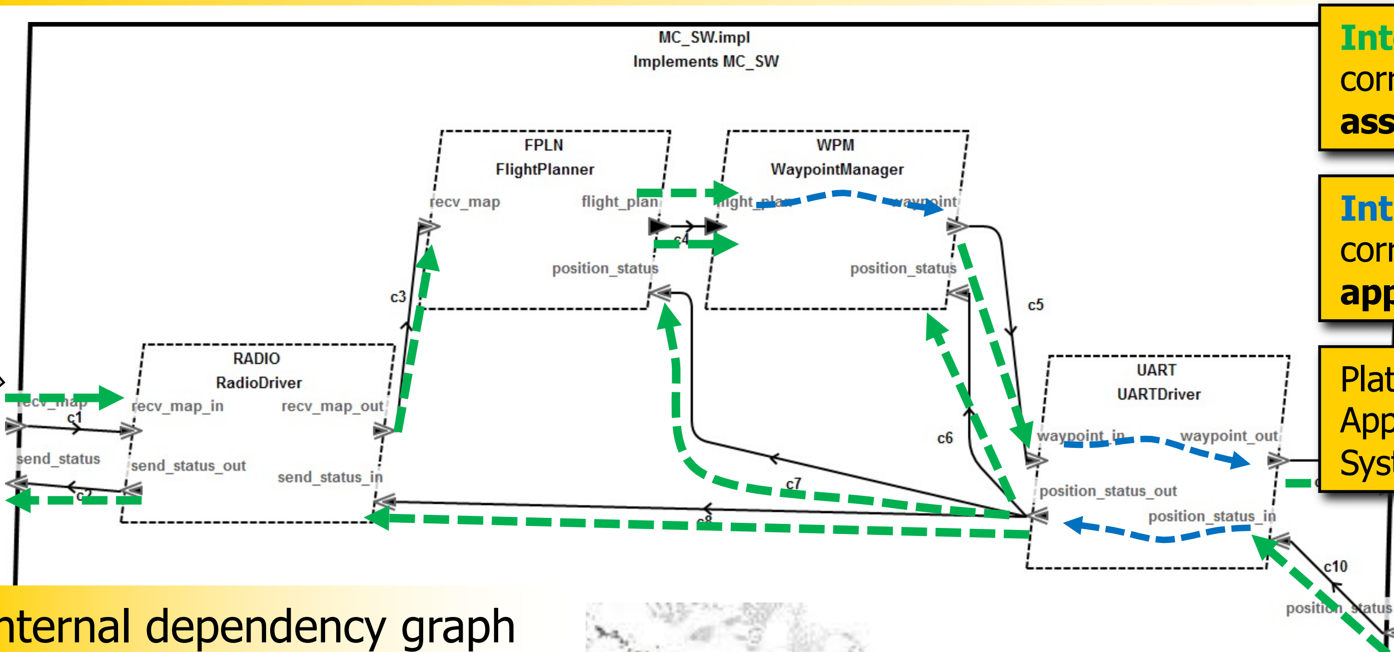
Component: PROC_SW		
In ports	*Flows*	*Out ports*
processor_IN	position_status->send_status	processor_OUT
position_status	recv_map->waypoint	send_status
recv_map	position_status->waypoint	waypoint

Awas component **visualizations focus developer attention on information flow aspects** (other aspects of AADL specs are elided)



Information Flow Analysis Foundation

Internal dependency graphs upon which analysis is performed are built from architecture connections and intra-component AADL flow annotations as well as AADL EM annotations

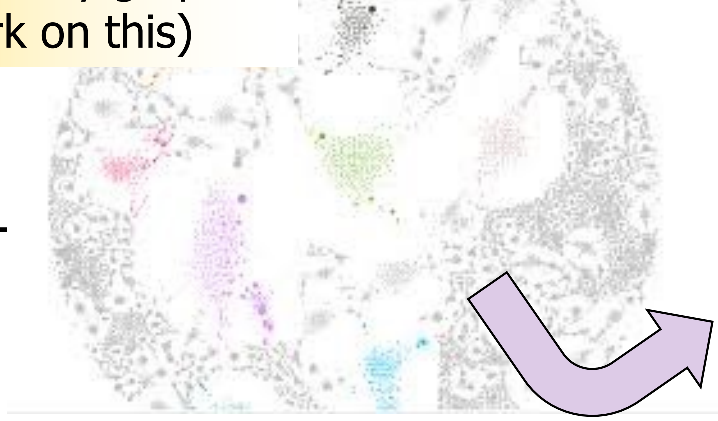


Inter-component info flow correctness is part of **platform assurance** (done once)

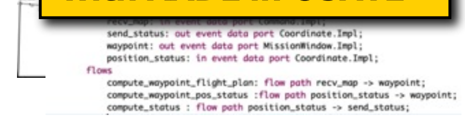
Intra-component info flow correctness is part of **application assurance**

Platform assurance **+** Application assurance **=>** System assurance

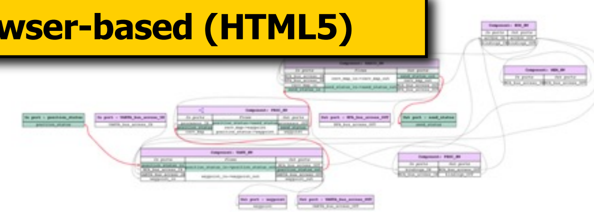
Internal dependency graph (algorithms work on this)



Markup/Interaction with AADL in OSATE



Browser-based (HTML5)

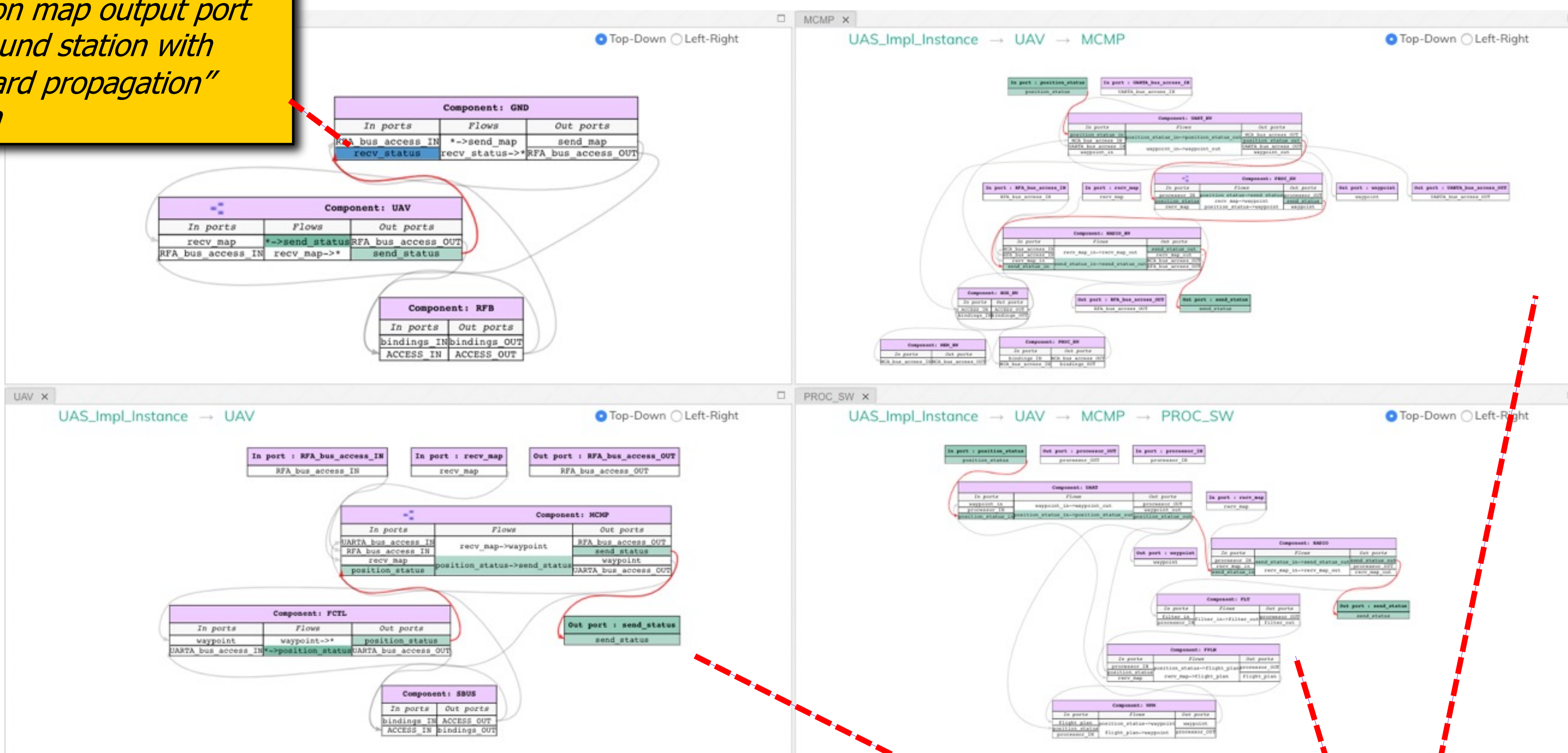


Interactions and rendered results

Interactive Browsing of Information Flows (AADL Level)

Example: In Ground Station / UAV example used on DARPA CASE, ask "how does map information propagation from ground station to UAV and through UAV's mission computer to produce a waypoint?"

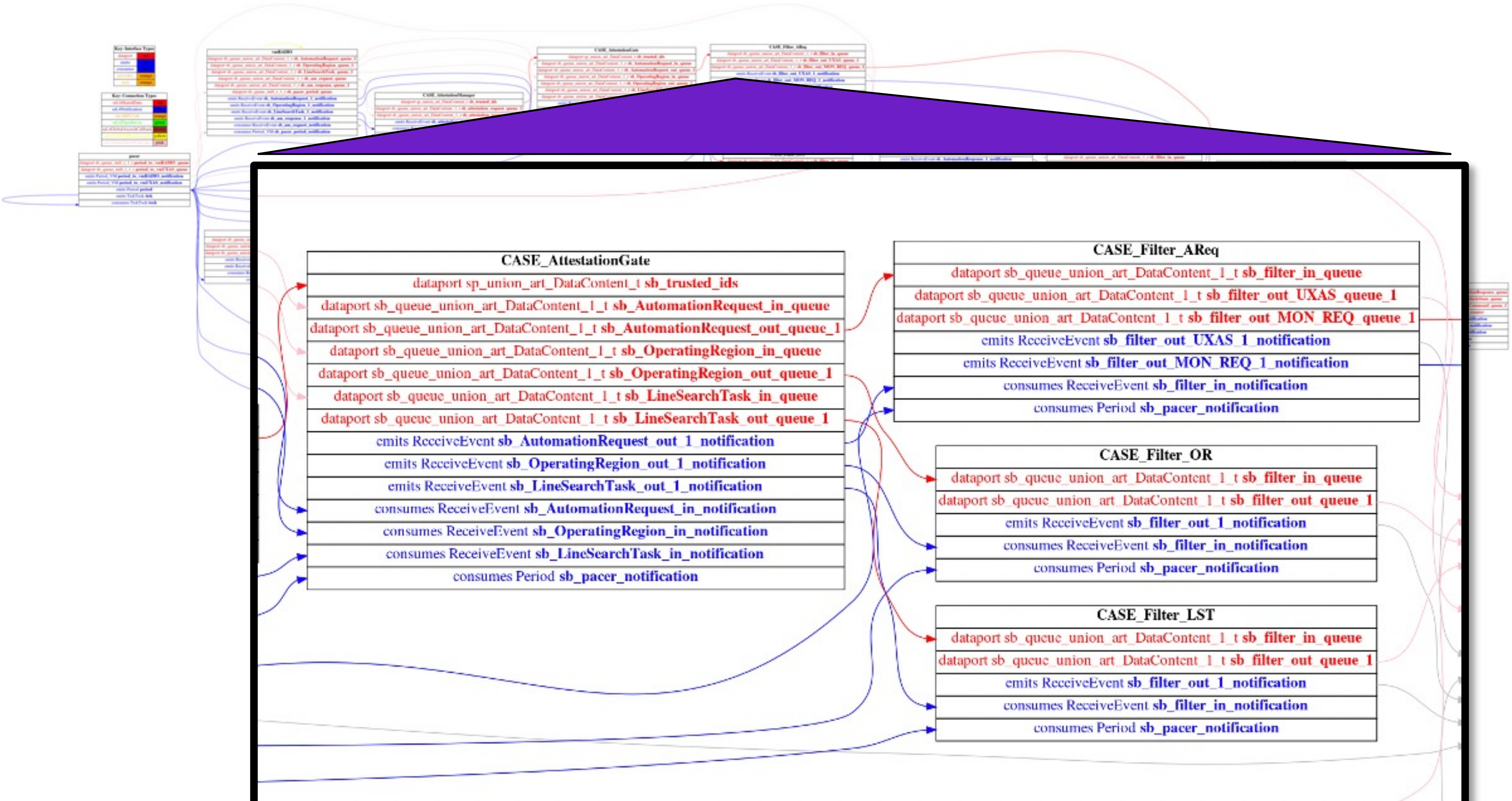
Click on map output port of ground station with "forward propagation" option



Immediately see results of across different subsystems.

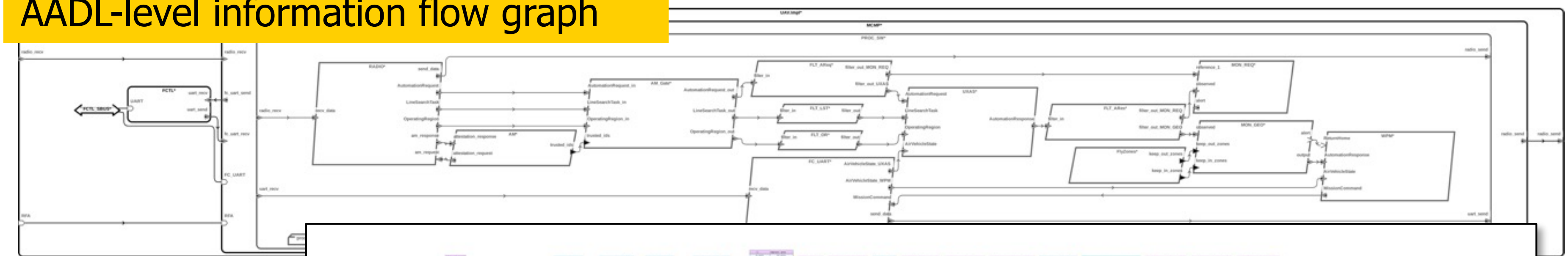
HAMR CAmkES/seL4 Platform Flows

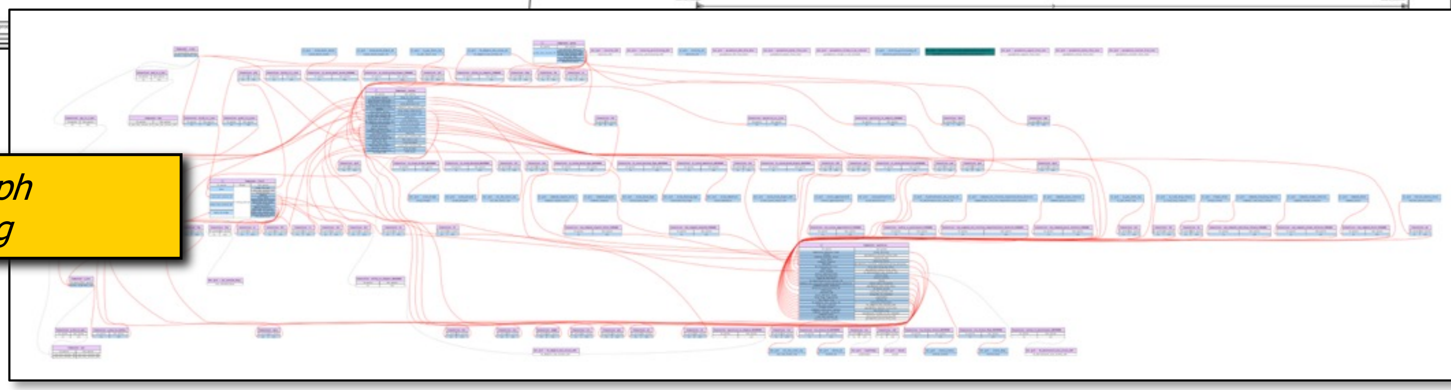
HAMR generates graphs of CAmkES/seL4 platform flows (relying on assurance of CAmkES and seL4) along traceability information showing the correspondence between the model level specifications and realization of flow controls in the deployed code.



Platform Information Flow Assurance Sketch

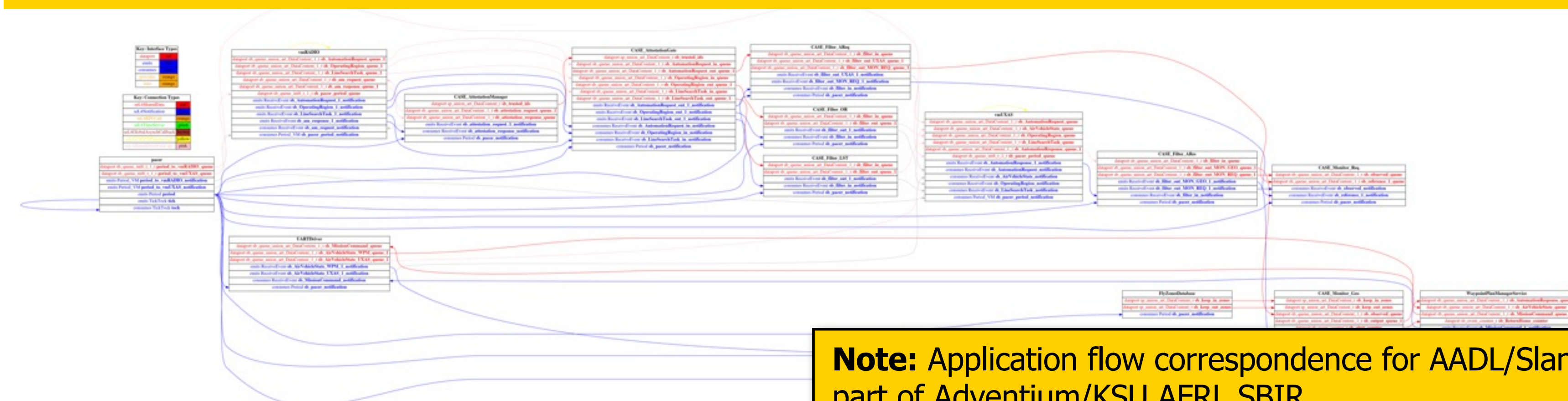
AADL-level information flow graph





SMT-based proof of graph isomorphism/embedding

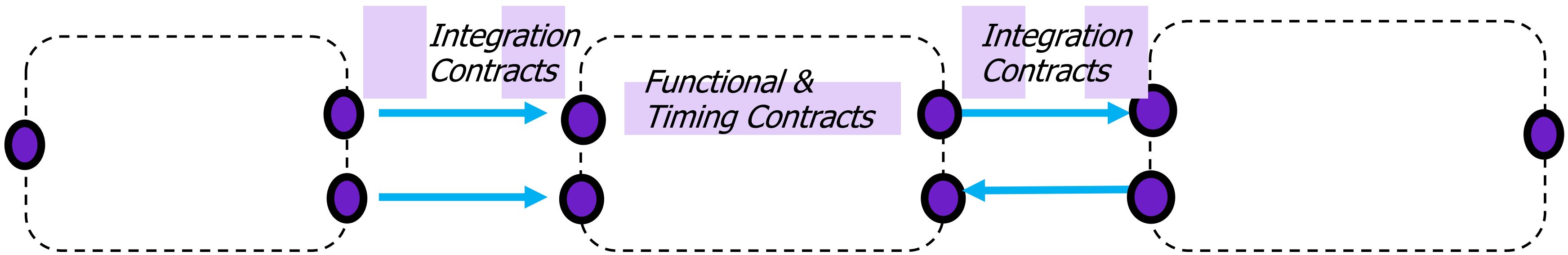
seL4 Platform-level information flow (at granularity of CAmkES topology)



Note: Application flow correspondence for AADL/Slang is part of Adventium/KSU AFRL SBIR

Contracts for Compositional Reasoning

On-going work: supporting compositional verification in AADL and HAMR-generated applications...

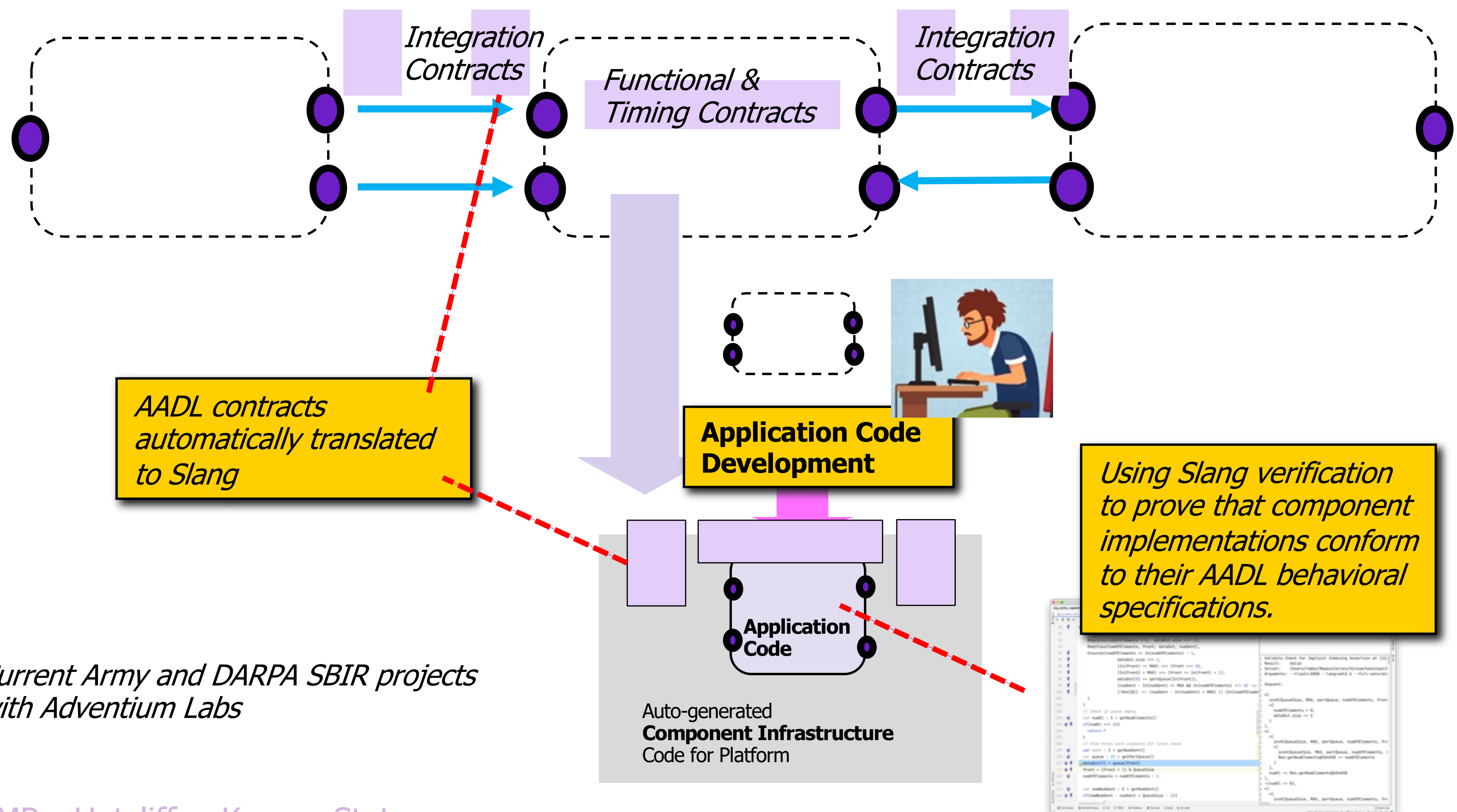


Existing AADL contract languages

- AGREE -- Assume-Guarantee REasoning Environment (used on DARPA CASE)
- BLESS – Behavior Language for Embedded Systems with Software

Contracts for Compositional Reasoning

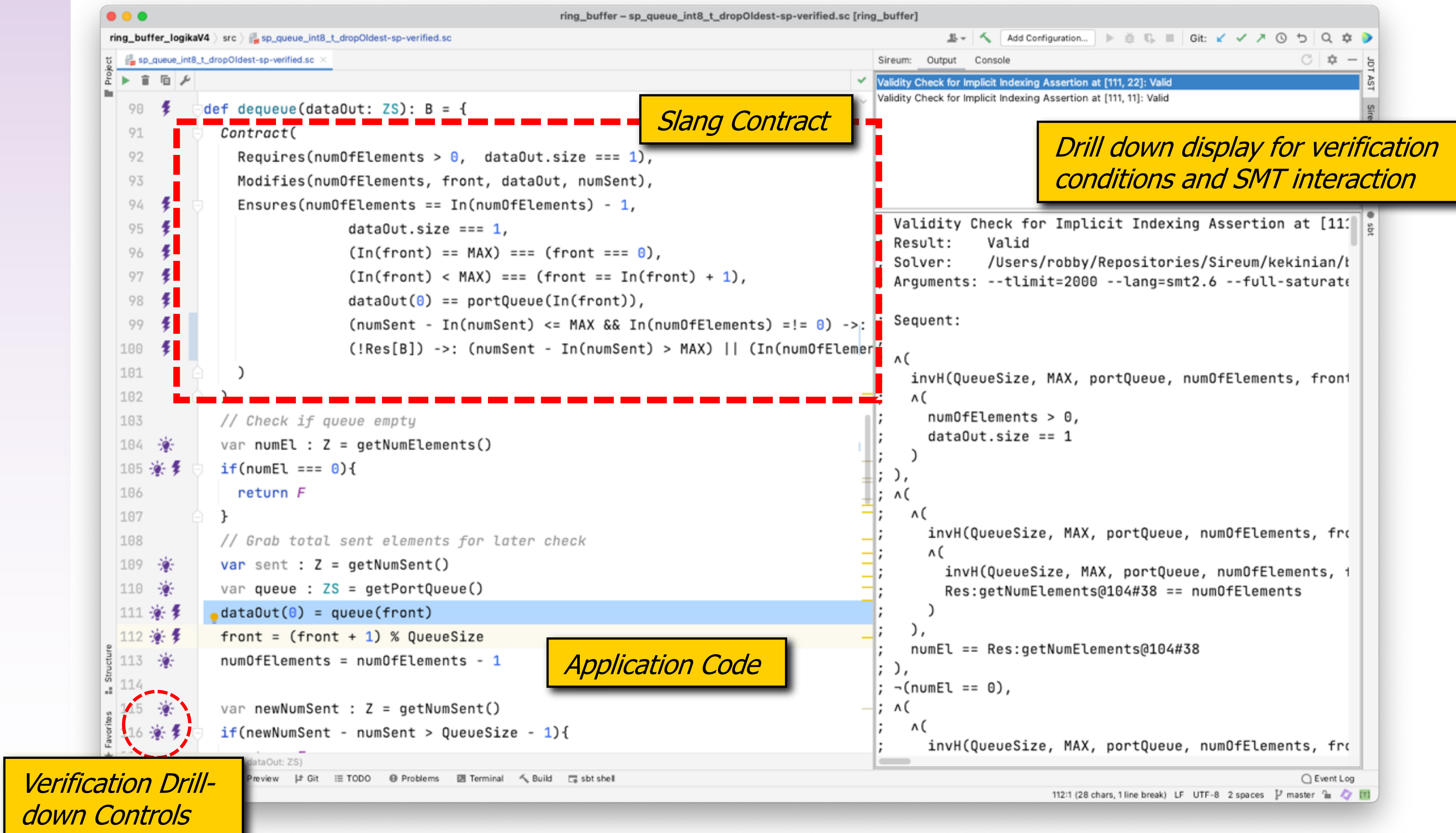
On-going work: supporting compositional verification in AADL and HAMR-generated applications...



Current Army and DARPA SBIR projects with Adventium Labs

Slang Contracts

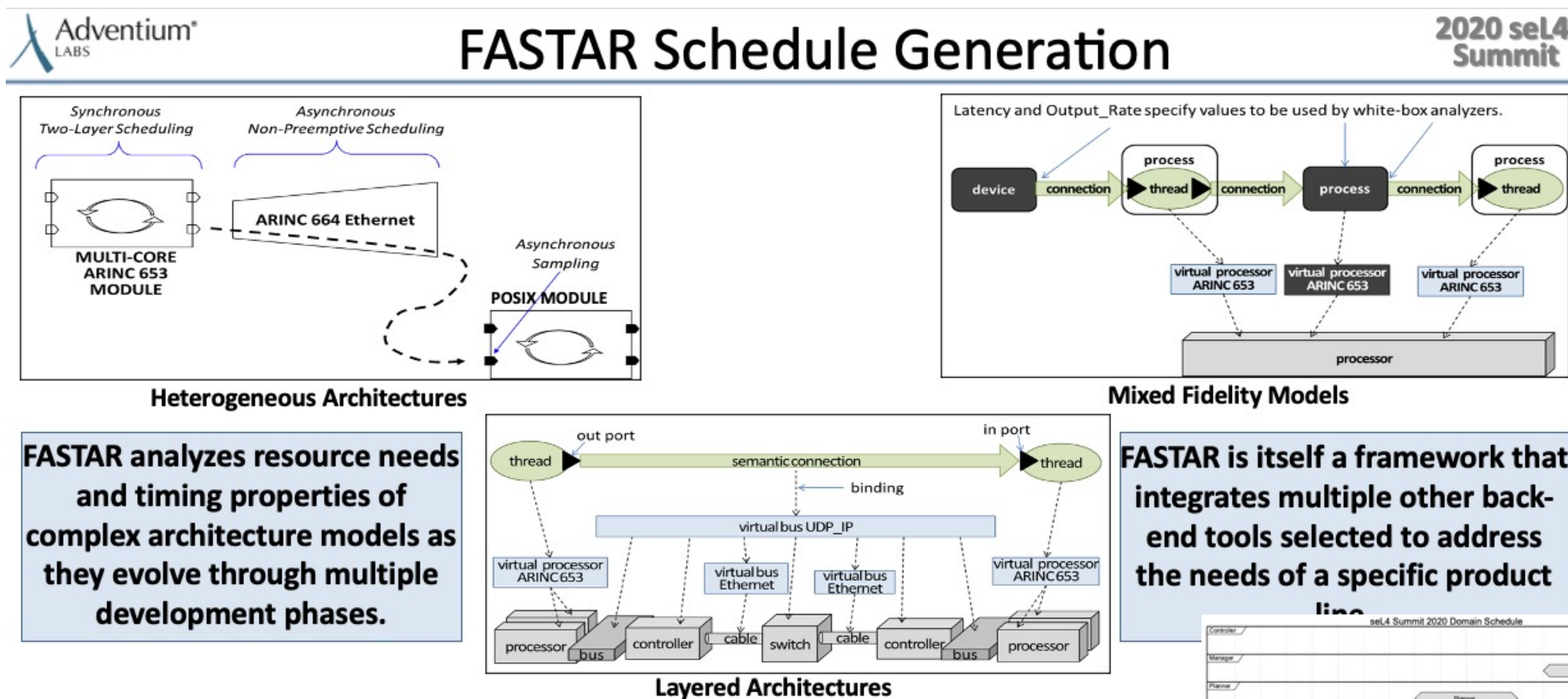
and Automated Verification via Symbolic Execution



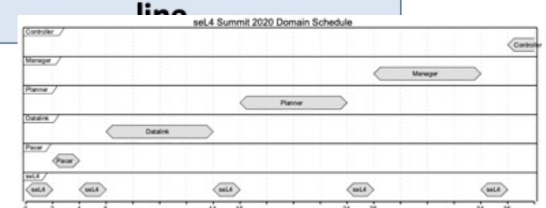
Slang applications can be integrated with Scala and Java and executed on JVM or translated to C (generated C is compatible with verified CompCert compiler)

Automated Scheduling Analysis in AADL

Again, HAMR is leveraging the System Engineering emphasis of AADL – in particular, tools like Adventium’s FASTAR can process AADL model structure and timing annotations to perform schedulability analysis and automatically generate schedules.



From Todd Carpenter’s “Time Enough for seL4” talk in Assured Systems II session.




```
const dschedule_t ksDomSchedule[] = {
  { .domain =  0, .length = 1 },    // 000 - 002 ms: 2 ms seL4
  { .domain =  1, .length = 1 },    // 002 - 004 ms: 2 ms Pacer
  { .domain =  0, .length = 1 },    // 004 - 006 ms: 2 ms seL4
  { .domain =  2, .length = 4 },    // 006 - 014 ms: 8 ms Datalink
  { .domain =  0, .length = 1 },    // 014 - 016 ms: 2 ms seL4
  { .domain =  3, .length = 4 },    // 016 - 024 ms: 8 ms Planner
  { .domain =  0, .length = 1 },    // 024 - 026 ms: 2 ms seL4
  { .domain =  4, .length = 4 },    // 026 - 034 ms: 8 ms Manager
  { .domain =  0, .length = 1 },    // 034 - 036 ms: 2 ms seL4
  { .domain =  5, .length = 2 },    // 036 - 040 ms: 4 ms Controller
};
```

Related Work

HAMR enhances the HACMS Collins/U Minn/Data61 *Trusted Build* concept to provide...

Sireum HAMR

High Assurance Model-based Rapid Engineering of Embedded Systems

- Completely new translation architecture with traceability mechanisms and support for eventual tool chain verification
- Alignment with AADL semantics
- Implementation of standardized AADL run-time services (key abstraction layer)
- Multiple platform support
- Multiple language support (adding Slang and CakeML)
- Automated insertion of virtual machines in seL4
- True one-way communication in seL4 (removing back channels)
- Traceability artifacts and information flow topology preservation proofs
- Temporal separation using seL4 domain scheduler

Other notable works on AADL code generation

- Ocarina – code generation for Ada and C (RT-POSIX threading, Xenomai, RTEMS, ARINC 653))
- RAMES – code generation for C (RT-Posix threading, nxtOSEK (LEGO Mindstorms), POK ARINC653-compliant.

Conclusion

Sireum HAMR

High Assurance Model-based Rapid Engineering of Embedded Systems

- HAMR helps support DARPA CASE goals by adopting multiple design goals that emphasis compositionality
- … leading to a rich model-based systems engineering framework that supports multiple deployment platforms and multiple implementation languages
- ...emphasizing assurance and rigorous development practices

Resources on HAMR web site

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

Questions

Sireum HAMR

High Assurance Model-based Rapid Engineering of Embedded Systems

http://hamr.sireum.org