AVIONICS SYSTEM OF SYSTEMS SIMULATION AND MODELING TOOL CHAIN (ASSIST)

February 3-4, 2021
AADL User Meeting

Prachee Sharma, Dhruv Monga
{psharma, dmonga}@poc.com

This work is performed under Army SBIR contract #'s:
PHASE I: W911W6-18-C-0047, W911W6-18-C-0012
PHASE II: W911W6-19-C-0015, W911W6-19-C-0038

Distribution Statement A: Approved For Public Release; Distribution Unlimited.

PHYSICAL OPTICS CORPORATION – A MERCURY COMPANY

Company Background (POC only a.k.a. Mission - Torrance)

Founded

Founded in 1985, POC was a small business, employee-owned company until acquired by Mercury Systems Inc. Jan 2021

Revenue

Financially Strong & Profitable
Revenue Projection for 2020 is $126M

Employees

332 Employees – 27 Ph.Ds,
156 Engineers
Mercury Systems Inc. 2,000+ Employees

Torrance, CA

170,000 sq. ft. facilities, 6 buildings.
27,000 sq. ft expansion in 2020

Patents

Over 160 issued patents – 60 technologies

Strategic Advisory Board

Outside Board Members
Independent Reviews

Certifications

AS9100, AS9110
Test Laboratory Accreditation
CMMI V1.3

ISOQAR QUALITY in AEROSPACE AS9100

ISOQAR QUALITY in AEROSPACE AS9110

ANAB ACCREDITED ISO/IEC 17025 TESTING LABORATORY

CMMIDEV/3 SM

Mission – Torrance Areas of Focus

Airborne

DTU
Data Transfer Unit

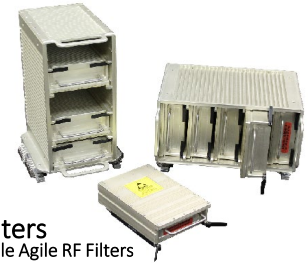
HDVR
High Definition Data & Video Recorder

AMCS
Aviation Mission Common Server

MLS – NAS
Network Attached Storage

JARVIS
Mission Computer – Distributed & Reconfigurable

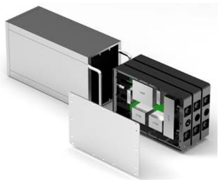
RF/EW



RF Filters
High Power Tunable Agile RF Filters

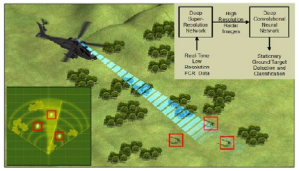
WISDEM
Wideband Intelligent Spectrum

PALM
Predistortion Amplifier



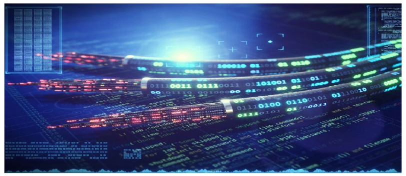
A2D
Interference Canceller

AI/Deep Learning


Deep Super-Resolution Network
High Resolution Radar Images
Deep Convolutional Neural Network
Real-Time Low Resolution FCR Data
Stationary Ground Target Detection and Classification

DEESTAC

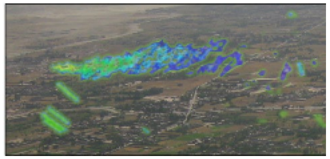
Cyber Security



Encrypted Data

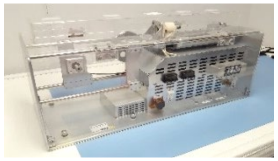
Sensor/Scanner

X-Ray
Non Destructive Corrosion Inspection



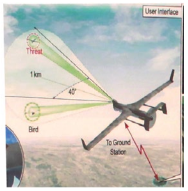
Data Fusion

Emerging Technology



ORFOM
Orbital Fiber Optic
Production Module

DREAM
Digital RF
Countermeasure



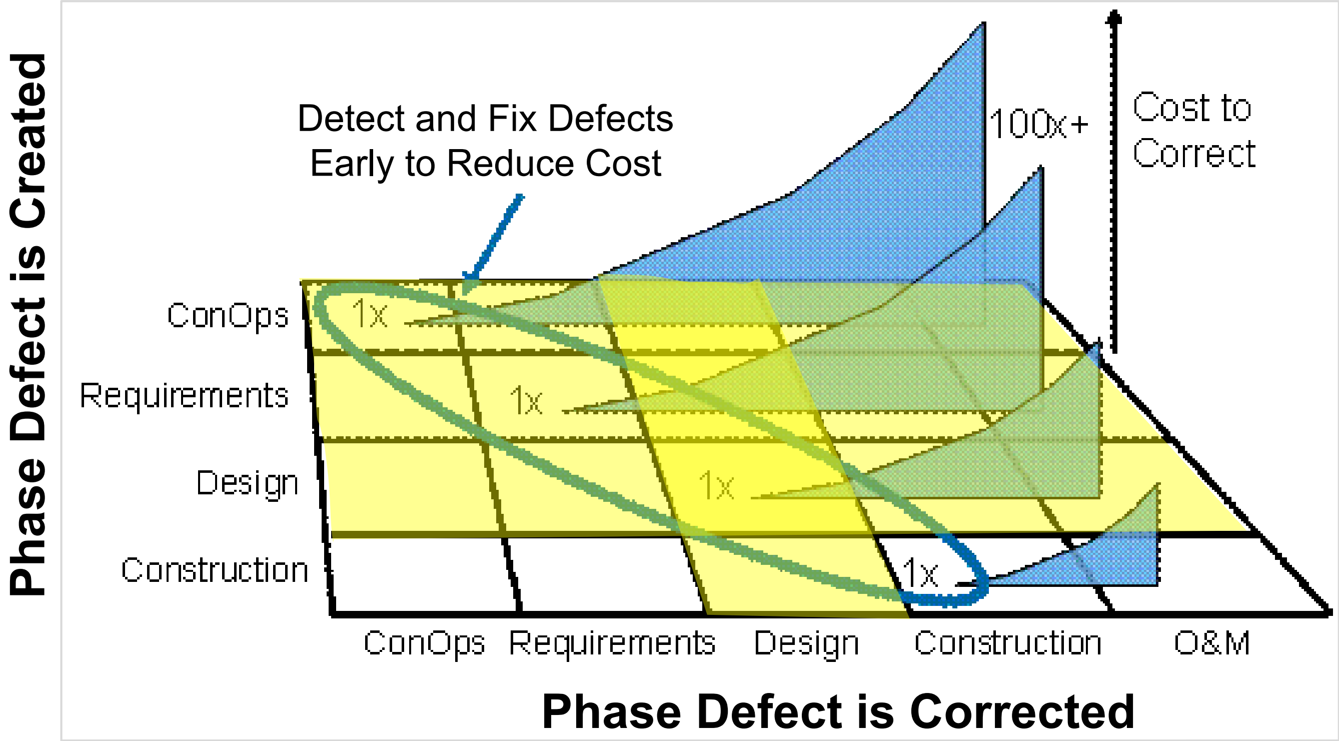
ARGUS
Unmanned
Surveillance

Problem Statement

- ***High Complexity of Avionics Software***
  - *Exponential growth in Source lines of code (SLOC)*
  - *High complexity means*
    - *High development cost and high cost of validation and verification (V&V)*
  - *Affordability of avionics development adversely impacted*
- ***Difficulty in Using Multicore Processors in Avionics***
  - *Difficult due to inability to verify performance during requirements, design and implementation stages*
  - *Analysis of hard real-time and soft real-time requirements needed*
  - *Shared resources make V&V difficult*
- ***Problems exacerbated in presence of high complexity software***
- ***Sustained growth in avionics requires dramatic cost-cutting measures to curtail rising costs***

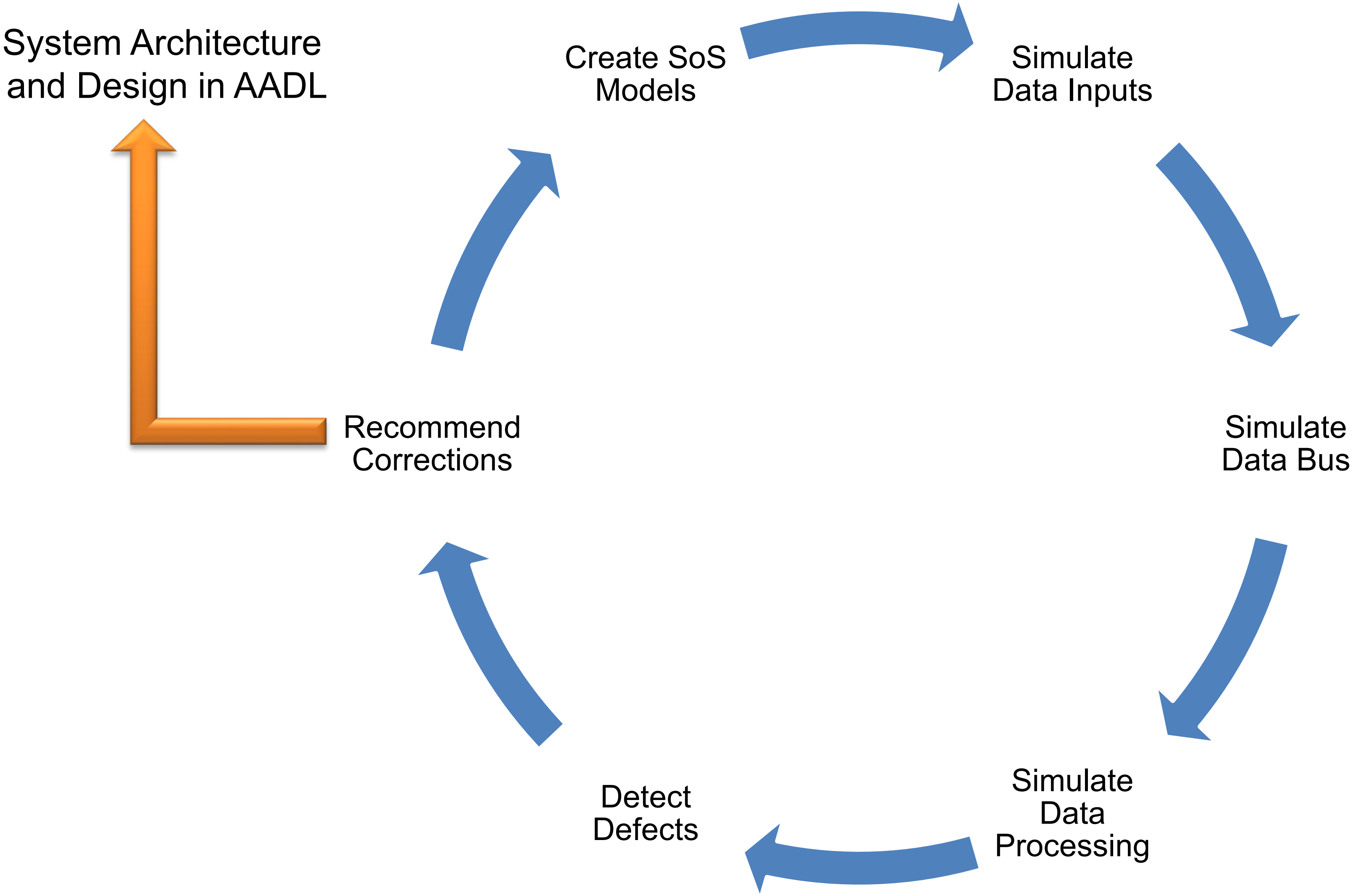
Solution

- ***ASSIST: Avionics System of Systems Simulation and Modeling Tool Chain***
- ***Cost reduction by***
  - *Detect defects created during CONOPS, Requirements and Design*
  - *Correct defects during avionics design*

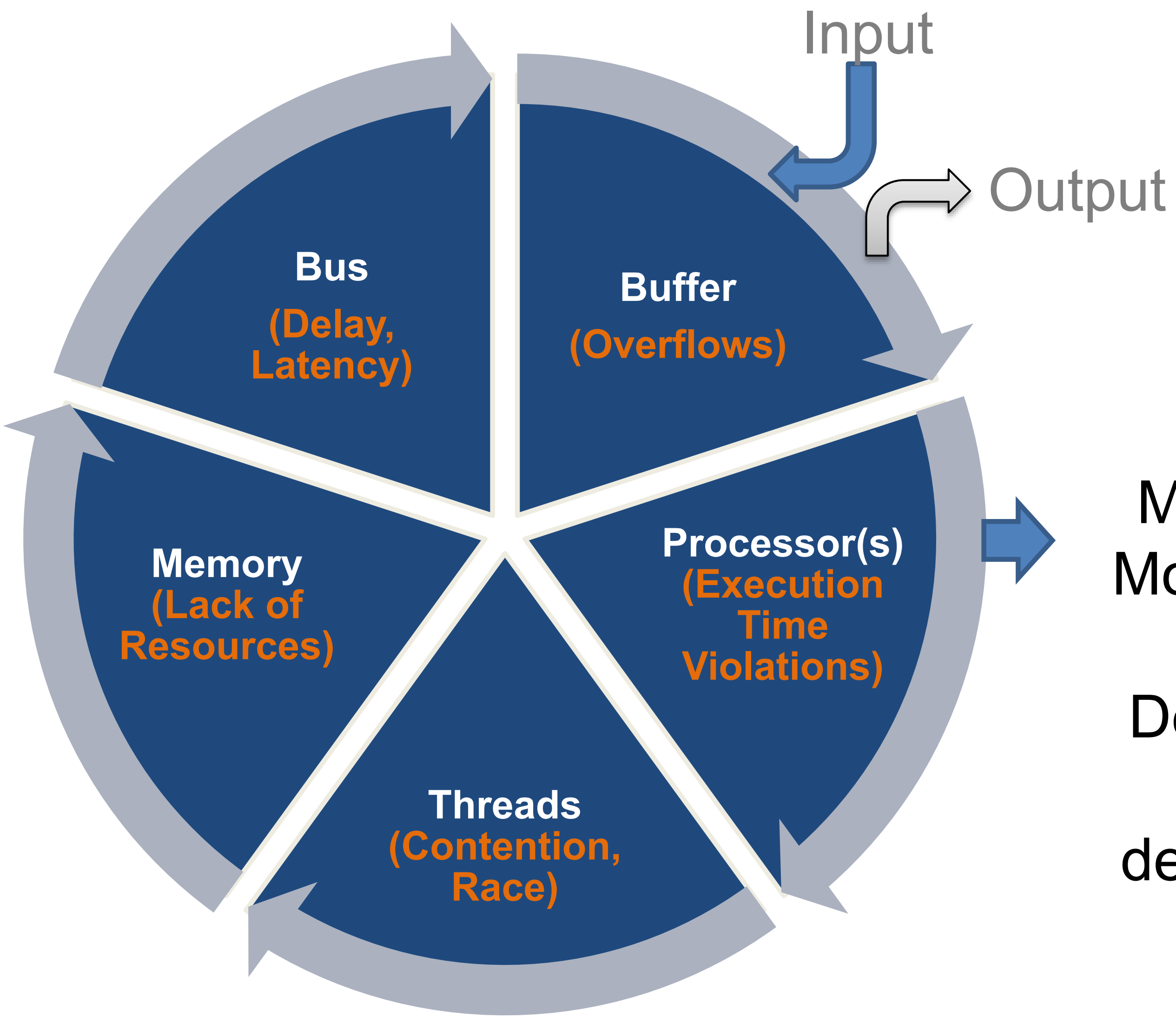


https://ops.fhwa.dot.gov/publications/seitsguide/section3.htm

ASSIST Approach



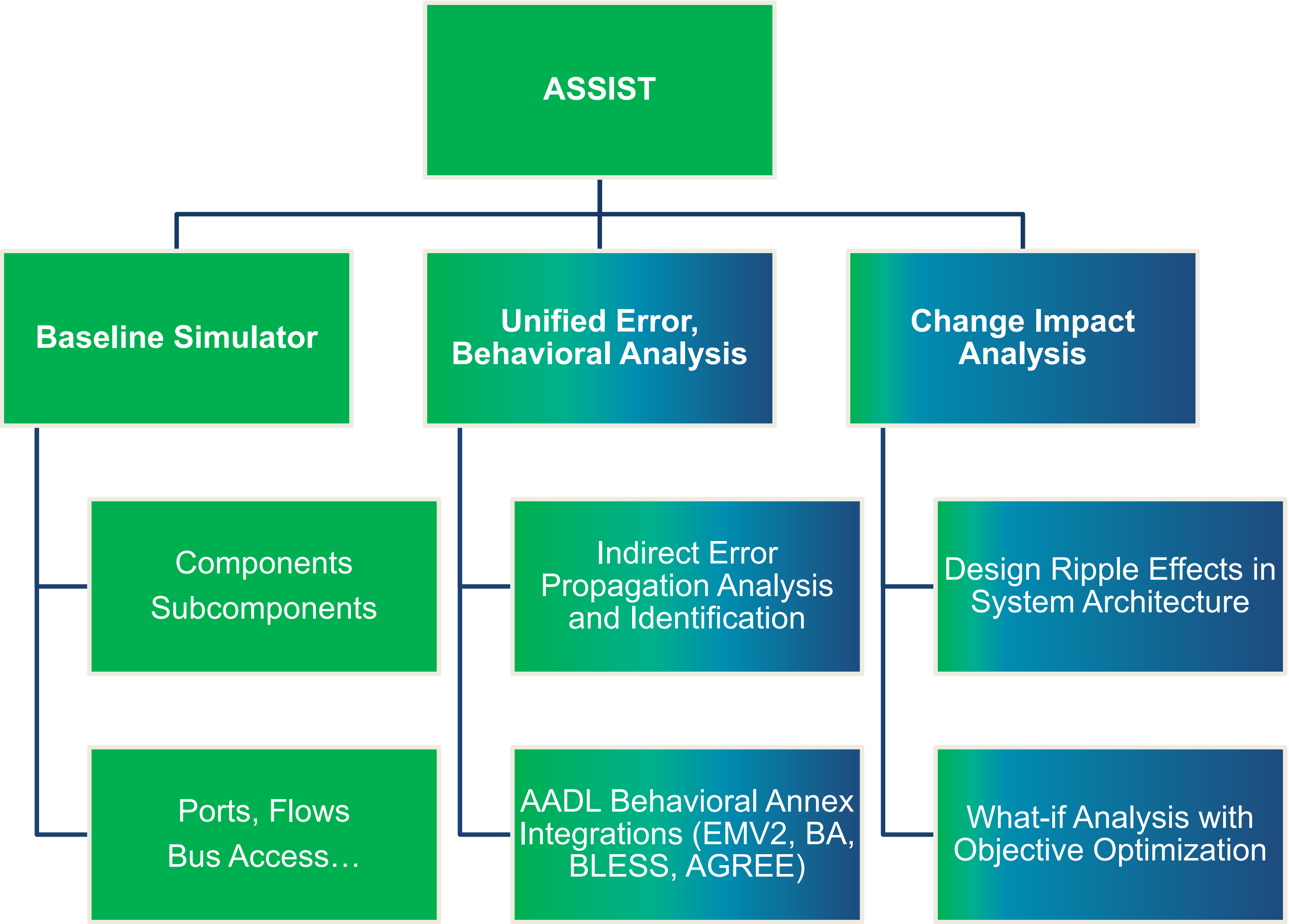
Ex.: Defects Detected, Corrected via M&S



Mutual Dependencies Modeled and Simulated

Defects caused due to mutual coupling detected and corrected

ASSIST Features



ASSIST

Baseline Simulator

- Components Subcomponents
- Ports, Flows Bus Access…

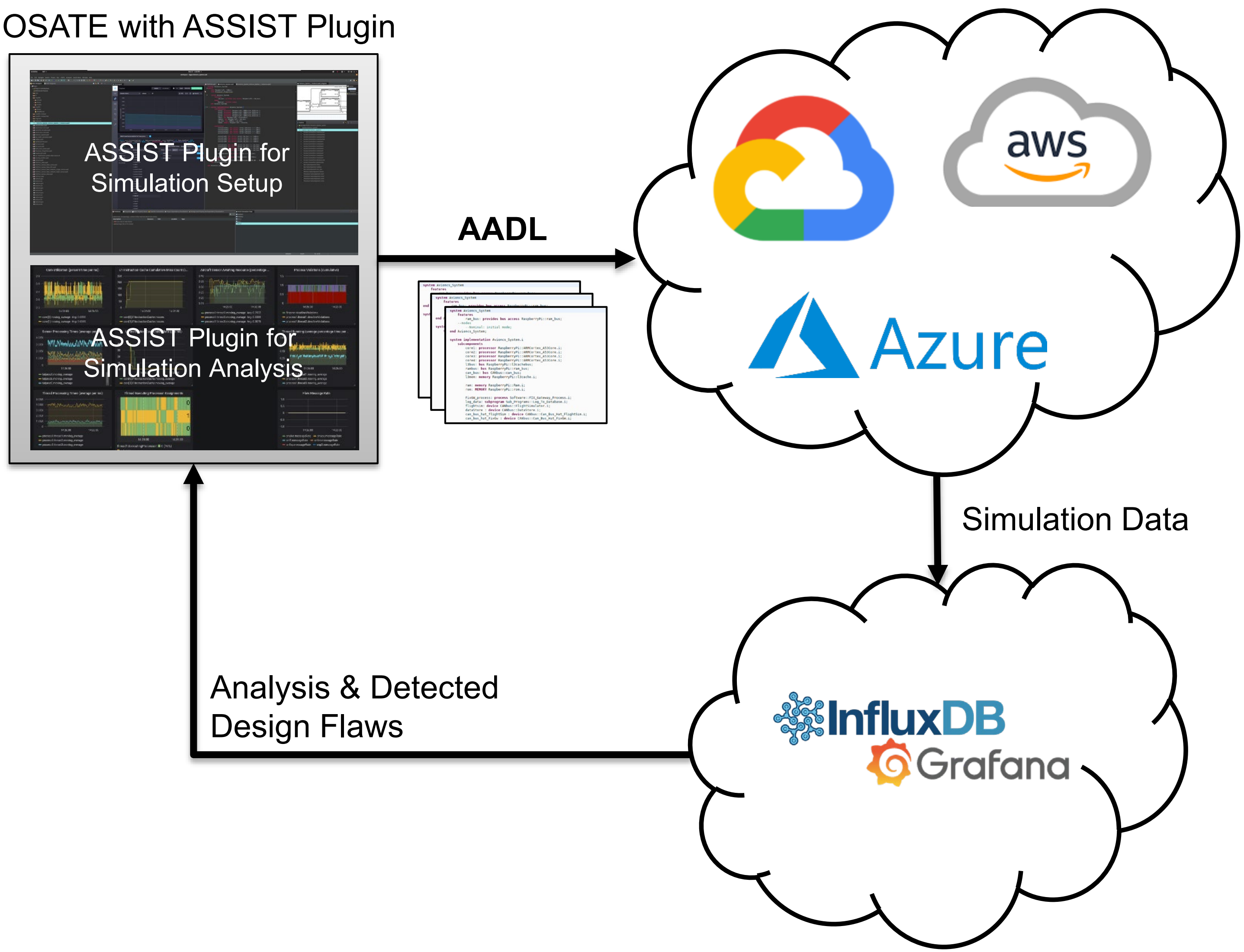
Unified Error, Behavioral Analysis

- Indirect Error Propagation Analysis and Identification
- AADL Behavioral Annex Integrations (EMV2, BA, BLESS, AGREE)

Change Impact Analysis

- Design Ripple Effects in System Architecture
- What-if Analysis with Objective Optimization

Cloud to Simulate Large Avionics Systems



OSATE with ASSIST Plugin

ASSIST Plugin for Simulation Setup

ASSIST Plugin for Simulation Analysis

AADL

Simulation Data

Analysis & Detected Design Flaws

Distributed Simulations

- Powerful machines for complex systems
- Parallel simulations for evaluating multiple designs

Secured Time Series Database

- Low latency time-series logging
- Built-in analysis and visualization
- Secured access to protect data privacy

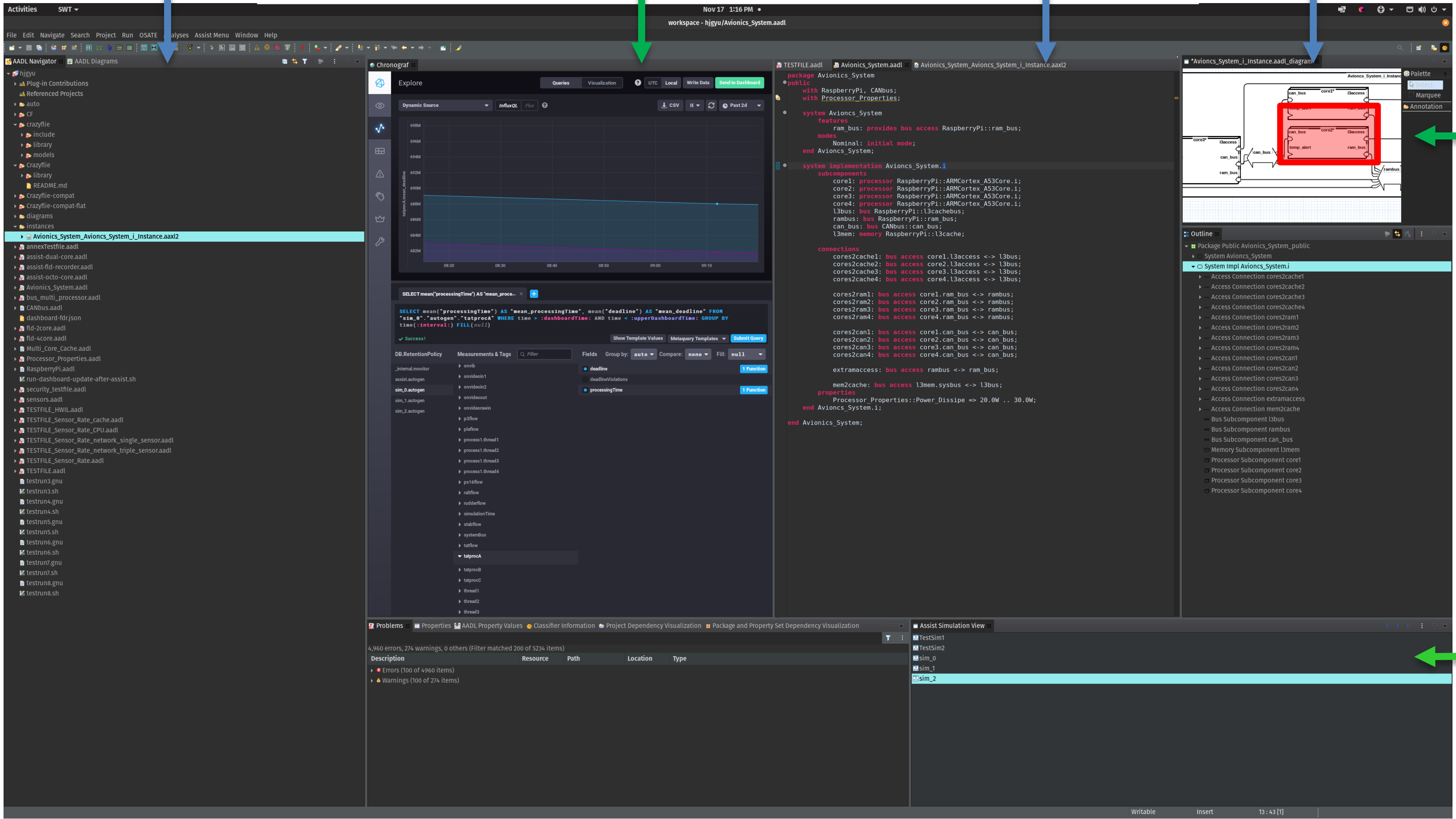
ASSIST is Integrated with OSATE Framework

Manage Projects

View Simulation Data and Real-time Analysis

Edit AADL

Visualize System



```
package Avionics_System
public
    with RaspberryPi, CANbus;
    with Processor_Properties;

    system Avioncs_System
        features
            ram_bus: provides bus access RaspberryPi::ram_bus;
        modes
            Nominal: initial mode;
    end Avioncs_System;

    system implementation Avioncs_System.i
        subcomponents
            core1: processor RaspberryPi::ARMCortex_A53Core.i;
            core2: processor RaspberryPi::ARMCortex_A53Core.i;
            core3: processor RaspberryPi::ARMCortex_A53Core.i;
            core4: processor RaspberryPi::ARMCortex_A53Core.i;
            l3bus: bus RaspberryPi::l3cachebus;
            rambus: bus RaspberryPi::ram_bus;
            can_bus: bus CANbus::can_bus;
            l3mem: memory RaspberryPi::l3cache;

        connections
            cores2cache1: bus access core1.l3access <-> l3bus;
            cores2cache2: bus access core2.l3access <-> l3bus;
            cores2cache3: bus access core3.l3access <-> l3bus;
            cores2cache4: bus access core4.l3access <-> l3bus;

            cores2ram1: bus access core1.ram_bus <-> rambus;
            cores2ram2: bus access core2.ram_bus <-> rambus;
            cores2ram3: bus access core3.ram_bus <-> rambus;
            cores2ram4: bus access core4.ram_bus <-> rambus;

            cores2can1: bus access core1.can_bus <-> can_bus;
            cores2can2: bus access core2.can_bus <-> can_bus;
            cores2can3: bus access core3.can_bus <-> can_bus;
            cores2can4: bus access core4.can_bus <-> can_bus;

            extramaccess: bus access rambus <-> ram_bus;

            mem2cache: bus access l3mem.sysbus <-> l3bus;
        properties
            Processor_Properties::Power_Dissipe => 20.0W .. 30.0W;
    end Avioncs_System.i;

end Avionics_System;
```

```
SELECT mean("processingTime") AS "mean_processingTime", mean("deadline") AS "mean_deadline" FROM
"sim_0"."autogen"."tatprocA" WHERE time > :dashboardTime: AND time < :upperDashboardTime: GROUP BY
time(:interval:) FILL(null)
```

```
4,960 errors, 274 warnings, 0 others (Filter matched 200 of 5234 items)
Errors (100 of 4960 items)
Warnings (100 of 274 items)
```

Visualize Design Flaws, Errors

Organize. Manage Simulations

Legend

Built-in OSATE Capabilities

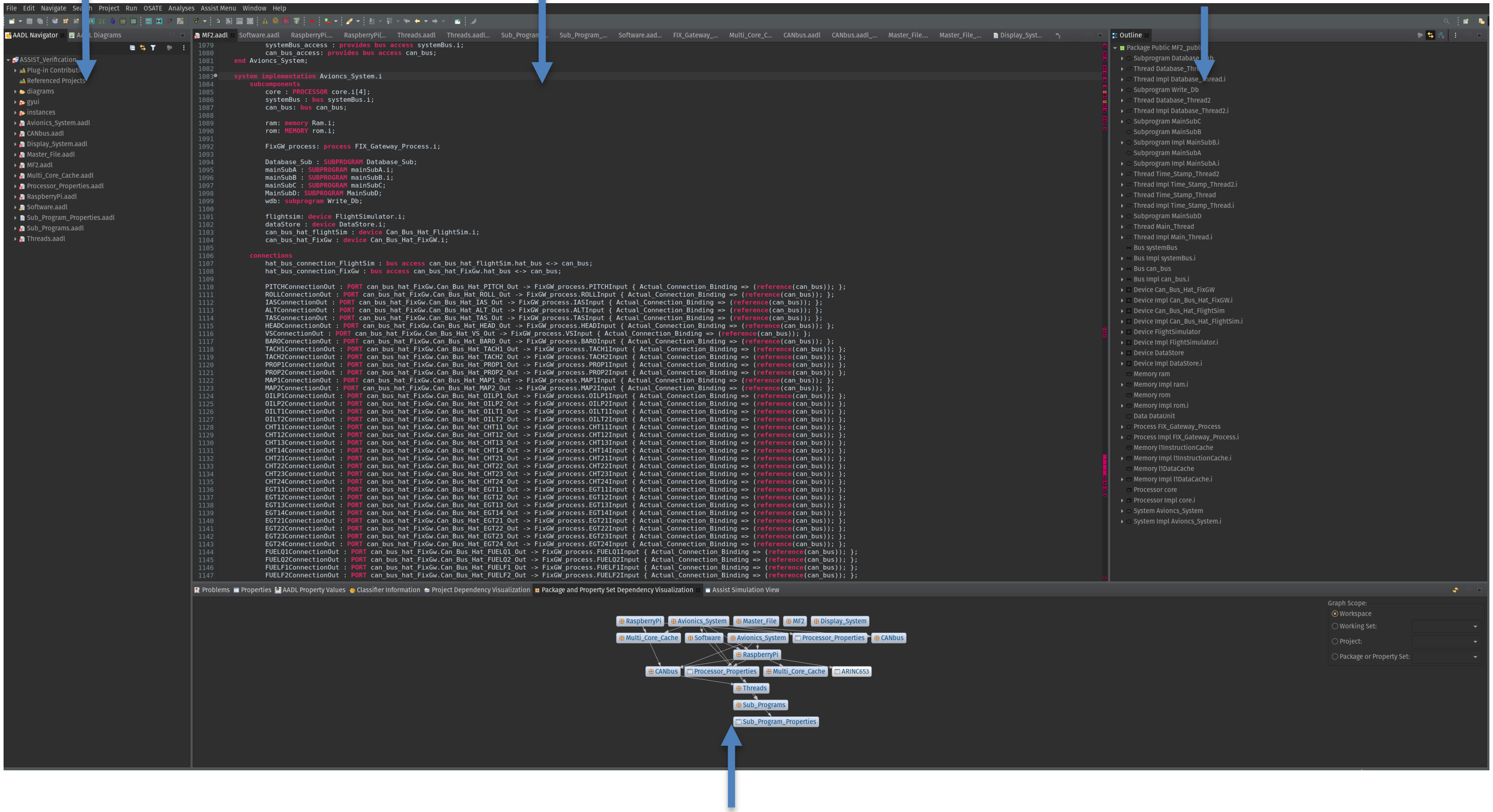
POC-developed Plugin Capabilities

Managing and Editing AADL Projects

Manage Projects

Edit AADL

View System Outline



```
            systemBus_access : provides bus access systemBus.i;
            can_bus_access: provides bus access can_bus;
    end Avioncs_System;

    system implementation Avioncs_System.i
        subcomponents
            core : PROCESSOR core.i[4];
            systemBus : bus systemBus.i;
            can_bus: bus can_bus;

            ram: memory Ram.i;
            rom: MEMORY rom.i;

            FixGW_process: process FIX_Gateway_Process.i;

            Database_Sub : SUBPROGRAM Database_Sub;
            mainSubA : SUBPROGRAM mainSubA.i;
            mainSubB : SUBPROGRAM mainSubB.i;
            mainSubC : SUBPROGRAM mainSubC;
            MainSubD: SUBPROGRAM MainSubD;
            wdb: subprogram Write_Db;

            flightsim: device FlightSimulator.i;
            dataStore : device DataStore.i;
            can_bus_hat_flightSim : device Can_Bus_Hat_FlightSim.i;
            can_bus_hat_FixGw : device Can_Bus_Hat_FixGW.i;

        connections
            hat_bus_connection_FlightSim : bus access can_bus_hat_flightSim.hat_bus <-> can_bus;
            hat_bus_connection_FixGw : bus access can_bus_hat_FixGw.hat_bus <-> can_bus;

            PITCHConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_PITCH_Out -> FixGW_process.PITCHInput { Actual_Connection_Binding => (reference(can_bus)); };
            ROLLConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_ROLL_Out -> FixGW_process.ROLLInput { Actual_Connection_Binding => (reference(can_bus)); };
            IASConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_IAS_Out -> FixGW_process.IASInput { Actual_Connection_Binding => (reference(can_bus)); };
            ALTConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_ALT_Out -> FixGW_process.ALTInput { Actual_Connection_Binding => (reference(can_bus)); };
            TASConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_TAS_Out -> FixGW_process.TASInput { Actual_Connection_Binding => (reference(can_bus)); };
            HEADConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_HEAD_Out -> FixGW_process.HEADInput { Actual_Connection_Binding => (reference(can_bus)); };
            VSConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_VS_Out -> FixGW_process.VSInput { Actual_Connection_Binding => (reference(can_bus)); };
            BAROConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_BARO_Out -> FixGW_process.BAROInput { Actual_Connection_Binding => (reference(can_bus)); };
            TACH1ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_TACH1_Out -> FixGW_process.TACH1Input { Actual_Connection_Binding => (reference(can_bus)); };
            TACH2ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_TACH2_Out -> FixGW_process.TACH2Input { Actual_Connection_Binding => (reference(can_bus)); };
            PROP1ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_PROP1_Out -> FixGW_process.PROP1Input { Actual_Connection_Binding => (reference(can_bus)); };
            PROP2ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_PROP2_Out -> FixGW_process.PROP2Input { Actual_Connection_Binding => (reference(can_bus)); };
            MAP1ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_MAP1_Out -> FixGW_process.MAP1Input { Actual_Connection_Binding => (reference(can_bus)); };
            MAP2ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_MAP2_Out -> FixGW_process.MAP2Input { Actual_Connection_Binding => (reference(can_bus)); };
            OILP1ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_OILP1_Out -> FixGW_process.OILP1Input { Actual_Connection_Binding => (reference(can_bus)); };
            OILP2ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_OILP2_Out -> FixGW_process.OILP2Input { Actual_Connection_Binding => (reference(can_bus)); };
            OILT1ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_OILT1_Out -> FixGW_process.OILT1Input { Actual_Connection_Binding => (reference(can_bus)); };
            OILT2ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_OILT2_Out -> FixGW_process.OILT2Input { Actual_Connection_Binding => (reference(can_bus)); };
            CHT11ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_CHT11_Out -> FixGW_process.CHT11Input { Actual_Connection_Binding => (reference(can_bus)); };
            CHT12ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_CHT12_Out -> FixGW_process.CHT12Input { Actual_Connection_Binding => (reference(can_bus)); };
            CHT13ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_CHT13_Out -> FixGW_process.CHT13Input { Actual_Connection_Binding => (reference(can_bus)); };
            CHT14ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_CHT14_Out -> FixGW_process.CHT14Input { Actual_Connection_Binding => (reference(can_bus)); };
            CHT21ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_CHT21_Out -> FixGW_process.CHT21Input { Actual_Connection_Binding => (reference(can_bus)); };
            CHT22ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_CHT22_Out -> FixGW_process.CHT22Input { Actual_Connection_Binding => (reference(can_bus)); };
            CHT23ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_CHT23_Out -> FixGW_process.CHT23Input { Actual_Connection_Binding => (reference(can_bus)); };
            CHT24ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_CHT24_Out -> FixGW_process.CHT24Input { Actual_Connection_Binding => (reference(can_bus)); };
            EGT11ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_EGT11_Out -> FixGW_process.EGT11Input { Actual_Connection_Binding => (reference(can_bus)); };
            EGT12ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_EGT12_Out -> FixGW_process.EGT12Input { Actual_Connection_Binding => (reference(can_bus)); };
            EGT13ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_EGT13_Out -> FixGW_process.EGT13Input { Actual_Connection_Binding => (reference(can_bus)); };
            EGT14ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_EGT14_Out -> FixGW_process.EGT14Input { Actual_Connection_Binding => (reference(can_bus)); };
            EGT21ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_EGT21_Out -> FixGW_process.EGT21Input { Actual_Connection_Binding => (reference(can_bus)); };
            EGT22ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_EGT22_Out -> FixGW_process.EGT22Input { Actual_Connection_Binding => (reference(can_bus)); };
            EGT23ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_EGT23_Out -> FixGW_process.EGT23Input { Actual_Connection_Binding => (reference(can_bus)); };
            EGT24ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_EGT24_Out -> FixGW_process.EGT24Input { Actual_Connection_Binding => (reference(can_bus)); };
            FUELQ1ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_FUELQ1_Out -> FixGW_process.FUELQ1Input { Actual_Connection_Binding => (reference(can_bus)); };
            FUELQ2ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_FUELQ2_Out -> FixGW_process.FUELQ2Input { Actual_Connection_Binding => (reference(can_bus)); };
            FUELF1ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_FUELF1_Out -> FixGW_process.FUELF1Input { Actual_Connection_Binding => (reference(can_bus)); };
            FUELF2ConnectionOut : PORT can_bus_hat_FixGw.Can_Bus_Hat_FUELF2_Out -> FixGW_process.FUELF2Input { Actual_Connection_Binding => (reference(can_bus)); };
```

Graph Scope:
- ● Workspace
- ○ Working Set:
- ○ Project:
- ○ Package or Property Set:

Visualize Project Dependencies

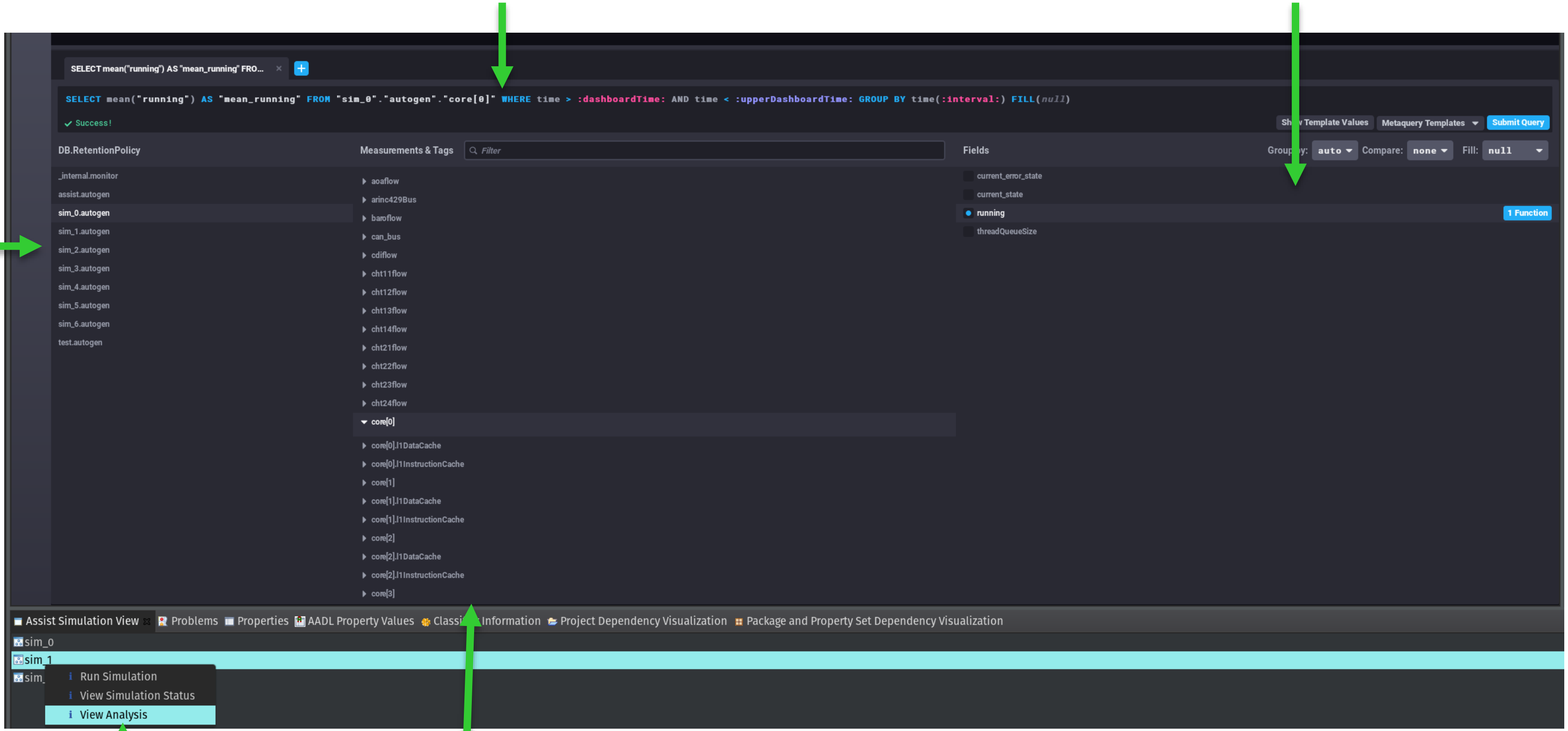
Legend
- Built-in OSATE Capabilities
- POC-developed Plugin Capabilities

Organize, Manage Simulations

Dynamically configure query for custom analysis

Select multiple analytics Per statistic

Browse and view analysis for multiple simulations



```
SELECT mean("running") AS "mean_running" FROM "sim_0"."autogen"."core[0]" WHERE time > :dashboardTime: AND time < :upperDashboardTime: GROUP BY time(:interval:) FILL(null)
```

✓ Success!

Open analysis window for simulations

Select multiple statistics to view

Legend

Built-in OSATE Capabilities

POC-developed Plugin Capabilities

View Simulation Data and Real-Time Analysis

Cumulative cache misses (top)
Cache miss count rate per ms (middle)

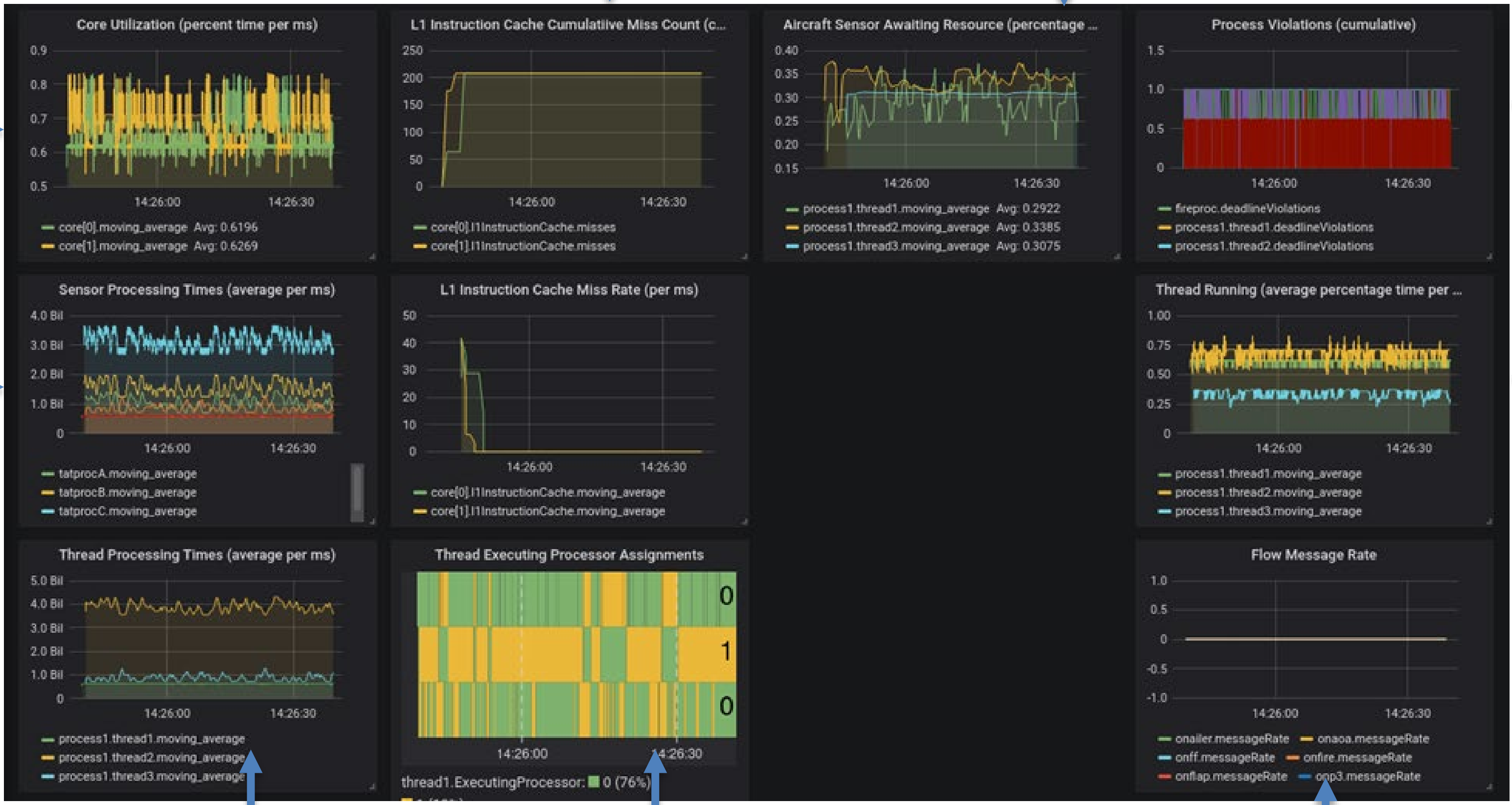
Time spent awaiting resources per ms per process

Core utilization (per ms)

Deadline violations per task

Total processing time per sensor data message

% time spent in running state per thread


Core Utilization (percent time per ms)
core[0].moving_average Avg: 0.6196
core[1].moving_average Avg: 0.6269
L1 Instruction Cache Cumulative Miss Count (c...
core[0].l1InstructionCache.misses
core[1].l1InstructionCache.misses
Aircraft Sensor Awaiting Resource (percentage ...
process1.thread1.moving_average Avg: 0.2922
process1.thread2.moving_average Avg: 0.3385
process1.thread3.moving_average Avg: 0.3075
Process Violations (cumulative)
fireproc.deadlineViolations
process1.thread1.deadlineViolations
process1.thread2.deadlineViolations
Sensor Processing Times (average per ms)
tatprocA.moving_average
tatprocB.moving_average
tatprocC.moving_average
L1 Instruction Cache Miss Rate (per ms)
core[0].l1InstructionCache.moving_average
core[1].l1InstructionCache.moving_average
Thread Running (average percentage time per ...
process1.thread1.moving_average
process1.thread2.moving_average
process1.thread3.moving_average
Thread Processing Times (average per ms)
process1.thread1.moving_average
process1.thread2.moving_average
process1.thread3.moving_average
Thread Executing Processor Assignments
0
1
0
thread1.ExecutingProcessor: 🟩 0 (76%)
Flow Message Rate
onailer.messageRate
onaoa.messageRate
onff.messageRate
onfire.messageRate
onflap.messageRate
onp3.messageRate
14:26:00
14:26:30

Processing time per thread execution

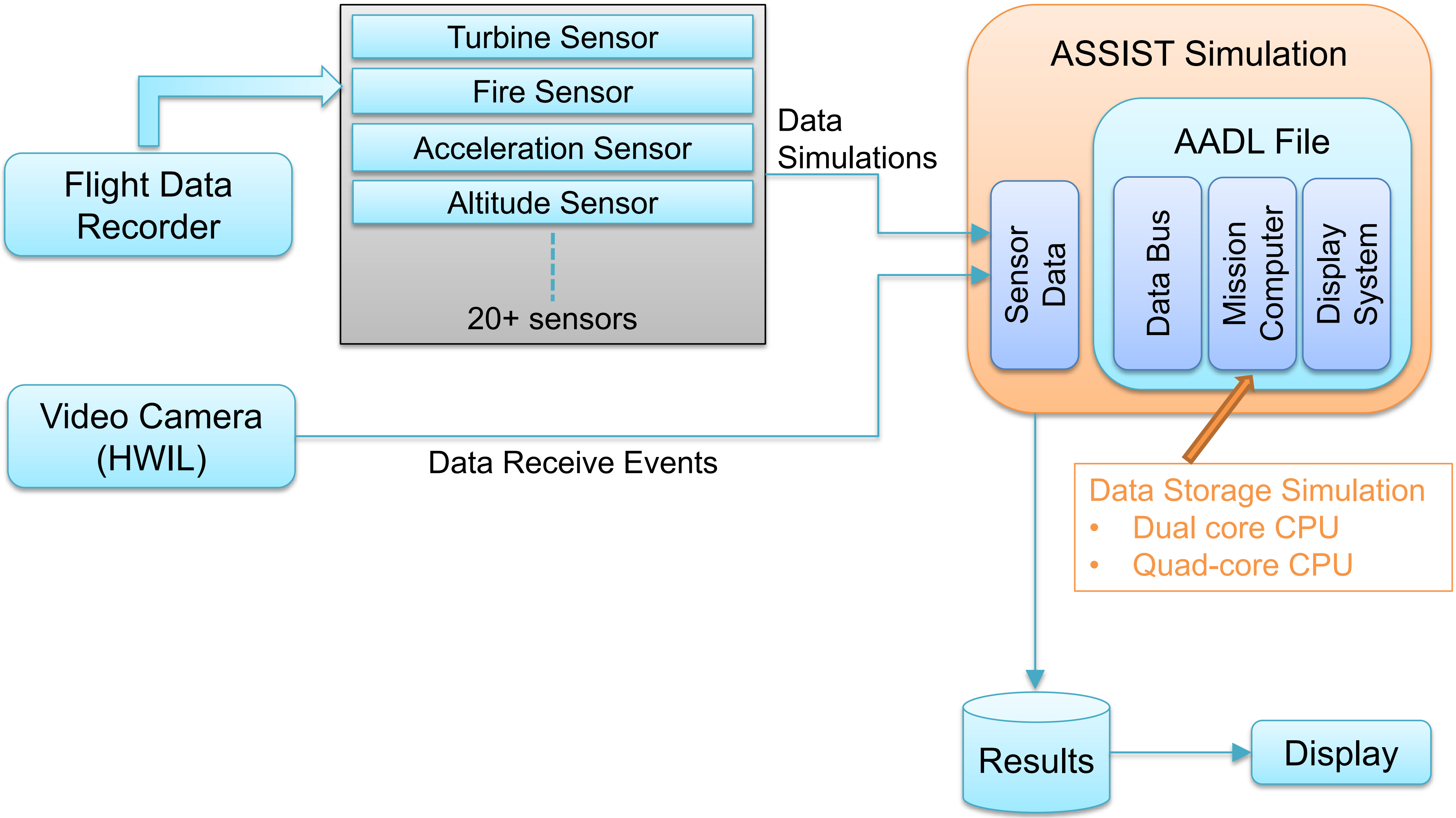
Core assignment per thread

Incoming data rate for messages

ASSIST

USING ASSIST

MultiCore Analysis



Data Storage Simulation

- Dual core CPU
- Quad-core CPU

Simulation Statistics Collected

- ***Data of interest for Multicore Processor Analysis***
  - *Federal Aviation Administration (FAA) Study - Assurance of Multicore Processors in Airborne Systems* *http://www.tc.faa.gov/its/worldpac/techrpt/tc16-51.pdf*
- ***Statistics recommended by FAA and collected by ASSIST:***
  - *Core utilization (% utilized averaged over ms)*
  - *Processing time per sensor message*
  - *Processing time per thread*
  - *Cache miss (+hit) counts and miss (+hit) rates/ms*
  - *Thread execution details:*
    - *Assigned processor*
    - *State transitions (running, executing, waiting on resource, idle)*
  - *Deadline violations*
  - *Flow rates per message*

Comparative Analysis

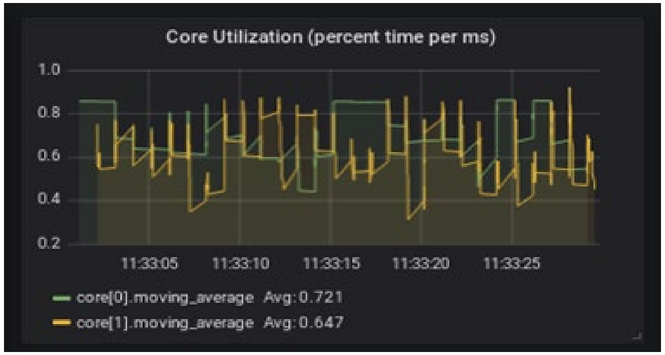
Processor Core Utilization
- Dual Core: 72% / 64%
- Quad Core: 45% / 44% / 78% / 47% (additional headroom) ✓

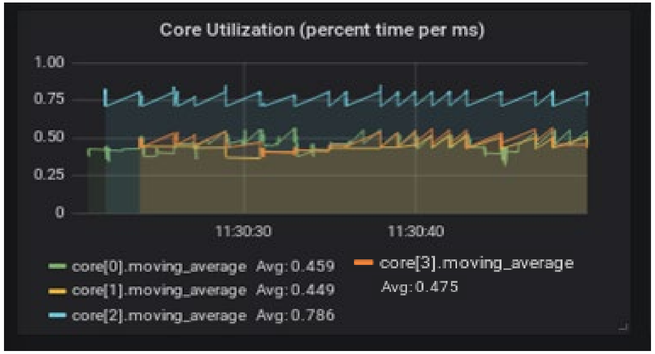
Thread Preemption Latency
- Dual Core: ~15-20 instances preemption exceeds 10%
- Quad Core: 0 instances preemption exceeds 10% ✓

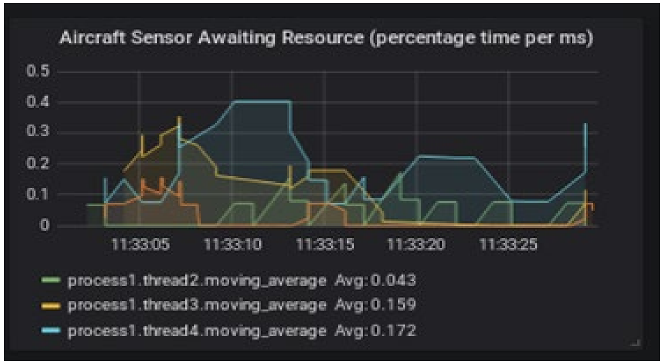
Processing Deadline Violations
- Dual Core: 4 critical threads 100’s violations
- Quad Core: only non-critical video thread ✓

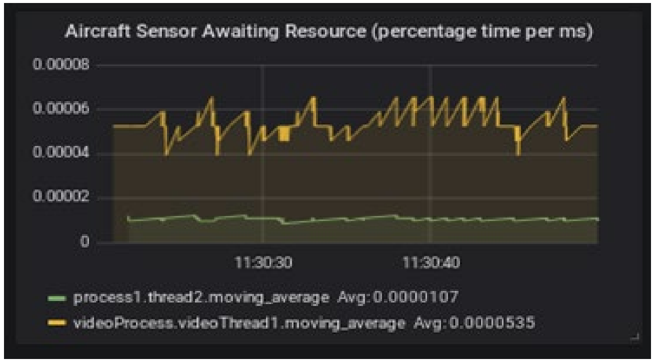
Dual Core

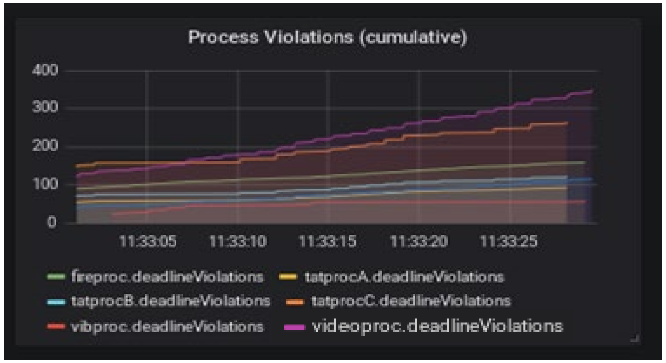
Quad Core

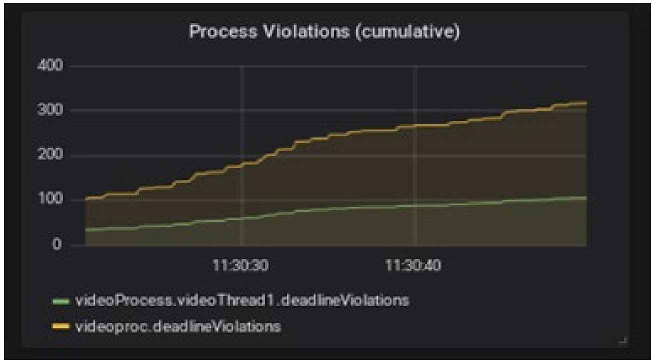




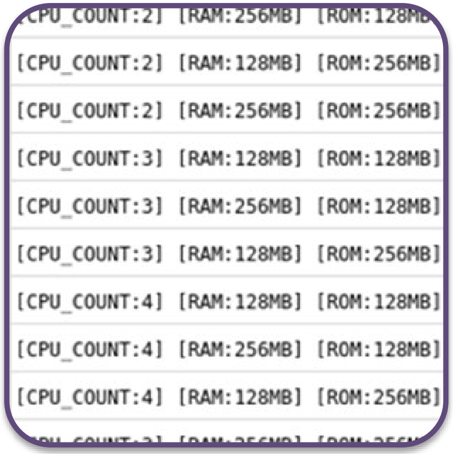








Change Analysis – What if Scenarios



[CPU_COUNT:2] [RAM:256MB] [ROM:128MB

[CPU_COUNT:2] [RAM:128MB] [ROM:256MB]

[CPU_COUNT:2] [RAM:256MB] [ROM:256MB]

[CPU_COUNT:3] [RAM:128MB] [ROM:128MB]

[CPU_COUNT:3] [RAM:256MB] [ROM:128MB]

[CPU_COUNT:3] [RAM:128MB] [ROM:256MB]

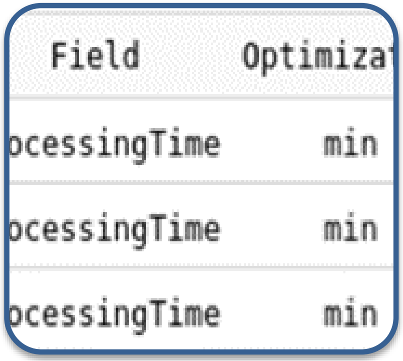
[CPU_COUNT:4] [RAM:128MB] [ROM:128MB]

[CPU_COUNT:4] [RAM:256MB] [ROM:128MB]

[CPU_COUNT:4] [RAM:128MB] [ROM:256MB]

Generate Alternative System Designs

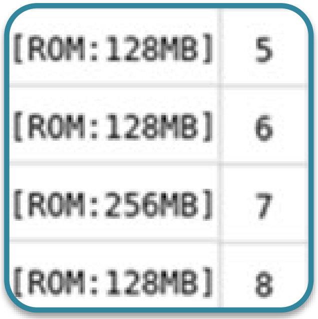
- Variations of baseline system design
- All permutations of parameters are possible
  - ○ # Cores, threads
  - ○ Size of memory
  - ○ Scheduling algorithms
  - ○ Bus types



Field	Optimiza
ocessingTime	min
ocessingTime	min
ocessingTime	min

Specify Design Criteria

- Create minimization and maximization criteria for system performance such as
  - ○ Minimize processing time
  - ○ Maximize CPU utilization
- Prioritize minimization and minimization criteria

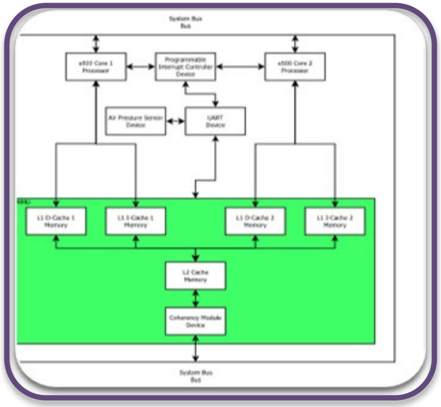


[ROM:128MB]	5
[ROM:128MB]	6
[ROM:256MB]	7
[ROM:128MB]	8

Rank Configurations

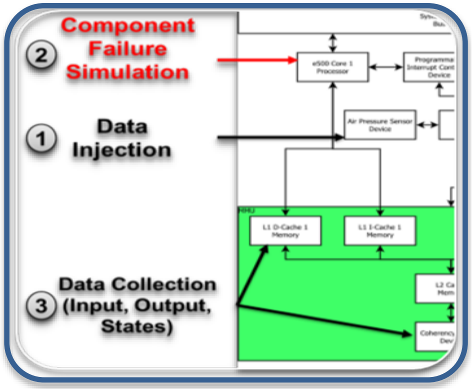
- Simulate alternate designs in parallel using cloud resources
- Generate performance statistics for each alternative
- Rank alternative designs using Multi-Criteria Decision Analysis (MCDA)

Error and Behavioral Analysis



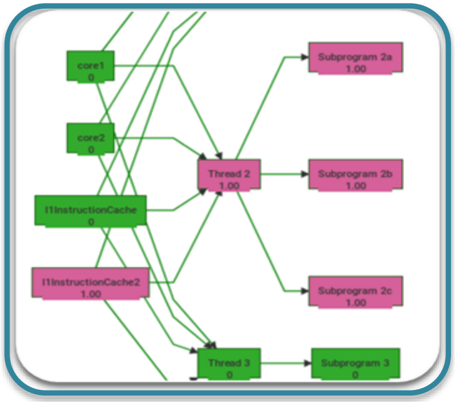
Define Error Propagation Paths and Behavioral Specifications

- Use AADL error annex to track failure propagations across components
- User AADL behavior annex to define behavioral specifications and resource requirements


2 Component Failure Simulation
1 Data Injection
3 Data Collection (Input, Output, States)
e500 Core 1 Processor
Programma Interrupt Cont Device
Air Pressure Sensor Device
L1 D-Cache 1 Memory
L1 I-Cache 1 Memory
L2 Ca Mem
Coherency Dev

Inject Component Failures

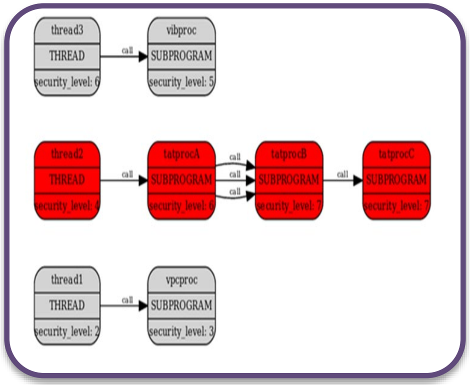
- Induce system failures during simulation
- Component failures can be generated to ensure evaluation of edge cases and common component failure conditions
- Based on the failure conditions for each component defined in AADL, ASSIST can algorithmically generate and simulate permutations of all possible system-wide component failure states



Identify Failure Propagations

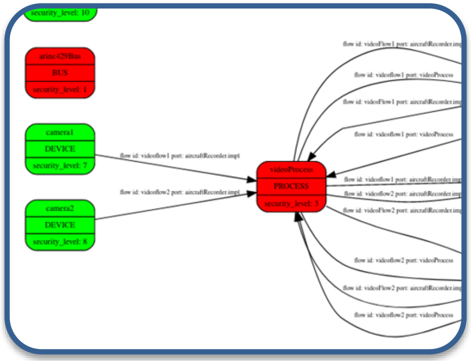
- Identify failure propagations resulting from error propagation paths defined in Annexes
- Identify indirect and induced failure propagations resulting from defined propagation paths

Security Analysis



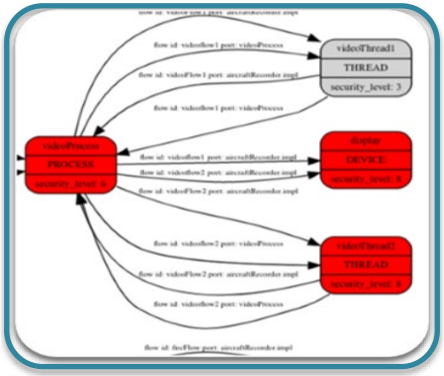
System Defined as Graph

- System components modeled as nodes and connections as edges
- Relationships currently represented are:
  - Subcomponents
  - Data flows
  - Connections
  - Software function calls


security_level: 10
arinc429Bus
BUS
security_level: 1
camera1
DEVICE
security_level: 7
camera2
DEVICE
security_level: 8
flow id: videoflow1 port: aircraftRecorder.impl
flow id: videoflow2 port: aircraftRecorder.impl
videoProcess
PROCESS
security_level: 3
flow id: videoFlow1 port: aircraftRecorder.impl
flow id: videoflow1 port: videoProcess
flow id: videoFlow1 port: aircraftRecorder.impl
flow id: videoflow1 port: videoProcess
flow id: videoflow1 port: aircraftRecorder.impl
flow id: videoflow2 port: aircraftRecorder.impl
flow id: videoFlow2 port: aircraftRecorder.impl
flow id: videoflow2 port: videoProcess
flow id: videoFlow2 port: aircraftRecorder.impl
flow id: videoflow2 port: videoProcess

Assign Security Classifications

- Assign security classifications to each component
- Component can be assigned varying levels of security classification representing secured or unsecured components using AADL property specifications


videoThread1
THREAD
security_level: 3
flow id: videoflow1 port: videoProcess
flow id: videoFlow1 port: aircraftRecorder.impl
flow id: videoflow1 port: videoProcess
videoProcess
PROCESS
security_level: 6
display
DEVICE
security_level: 8
flow id: videoflow1 port: aircraftRecorder.impl
flow id: videoflow2 port: aircraftRecorder.impl
flow id: videoFlow2 port: aircraftRecorder.impl
videoThread2
THREAD
security_level: 8
flow id: videoflow2 port: videoProcess
flow id: videoFlow2 port: aircraftRecorder.impl
flow id: videoflow2 port: videoProcess
flow id: fireFlow port: aircraftRecorder.impl

Identify Vulnerabilities

- Vulnerabilities identified by determining unsecured components connected to secured components through connection and data flow
- Security vulnerability propagation identified through graph analysis
- Security vulnerability generated scores generated for system and components

DISCUSSION

mercury | POC PHYSICAL OPTICS CORPORATION®

INNOVATION THAT MATTERS ™

FOLLOW US ON SOCIAL