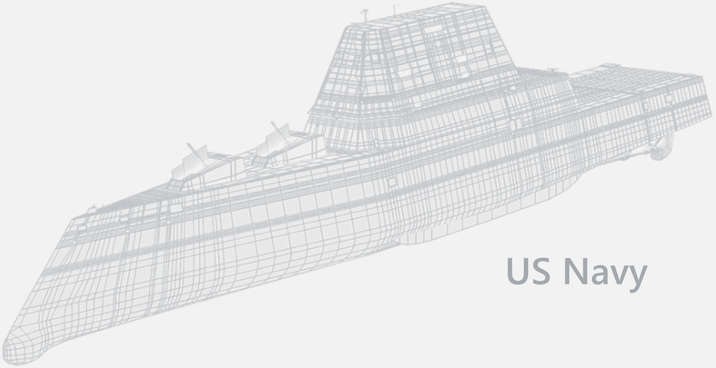
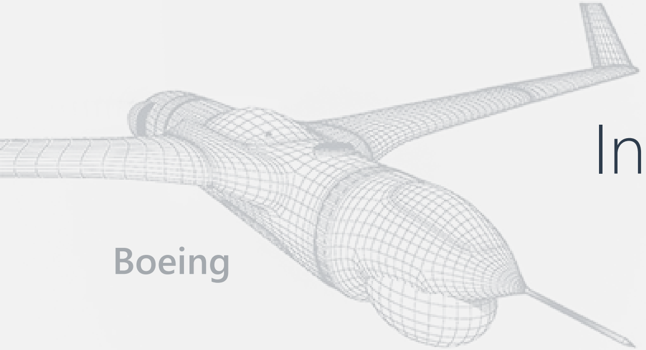
# Using AADL for Microarchitecture Models: Initial Experiences & Suggestions

Mark Brown
Lynx Software Technologies

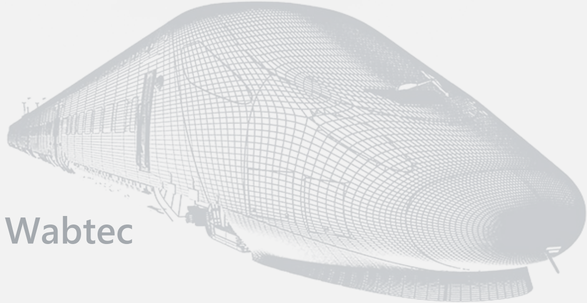
Make it simple.
Make it last.



US Navy



Boeing



Wabtec

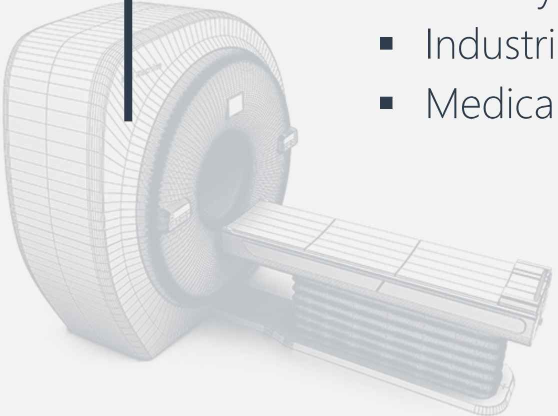
# Industries **Served**

## Security

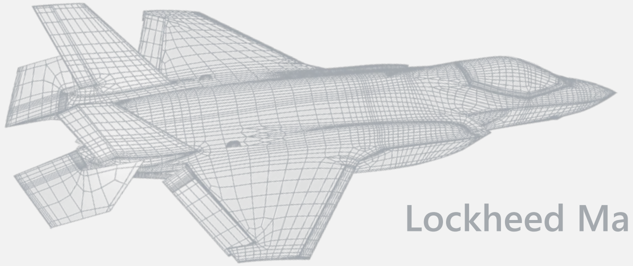
- Military
- Government
- Critical National Infrastructure
- Enterprise
- FinTech
- IIoT

## Safety

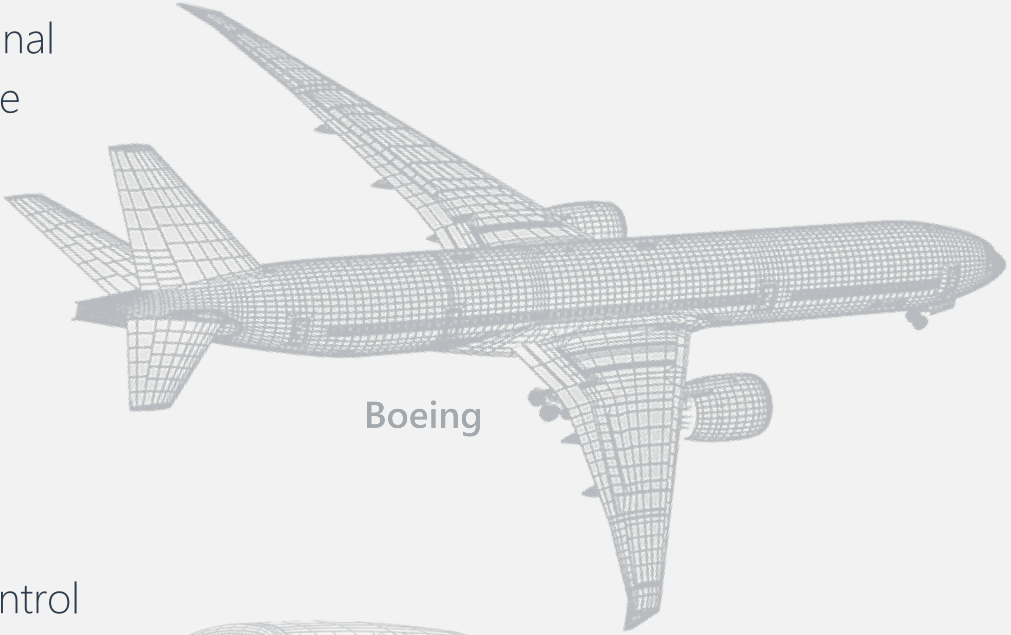
- Avionics
- Automotive
- Railway
- Industrial Control
- Medical



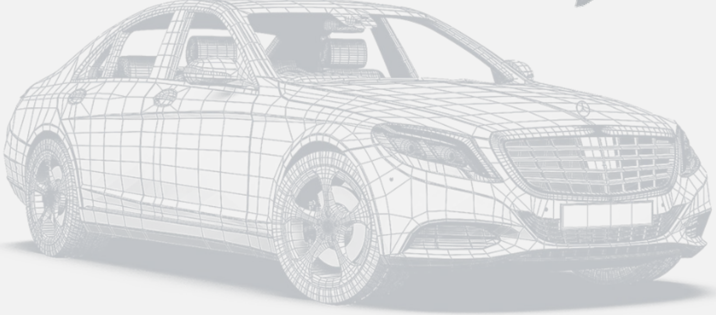
Elekta



Lockheed Martin



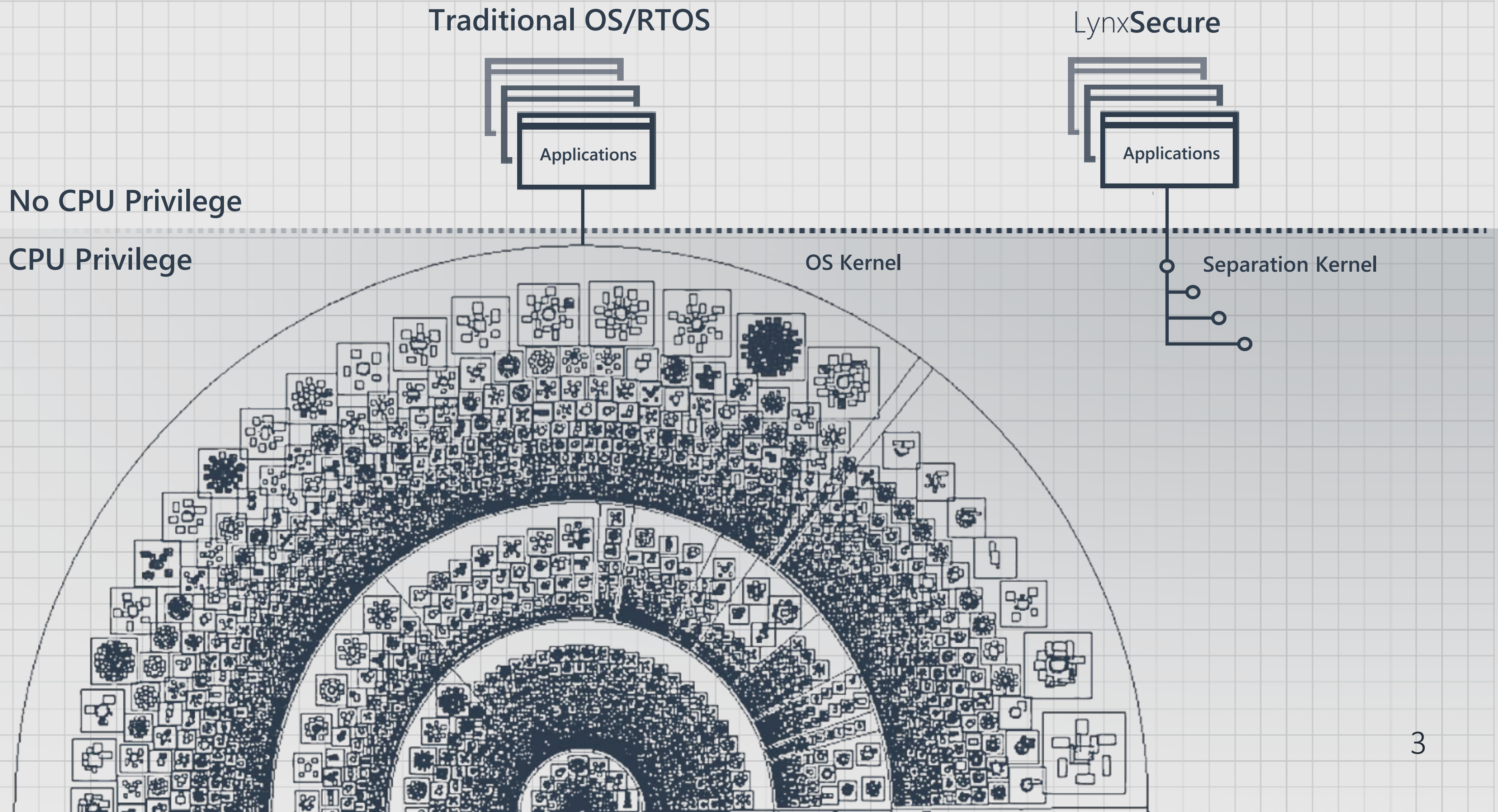
Boeing



Bosch + ETAS

# Assurance

"Kernel" Comparison



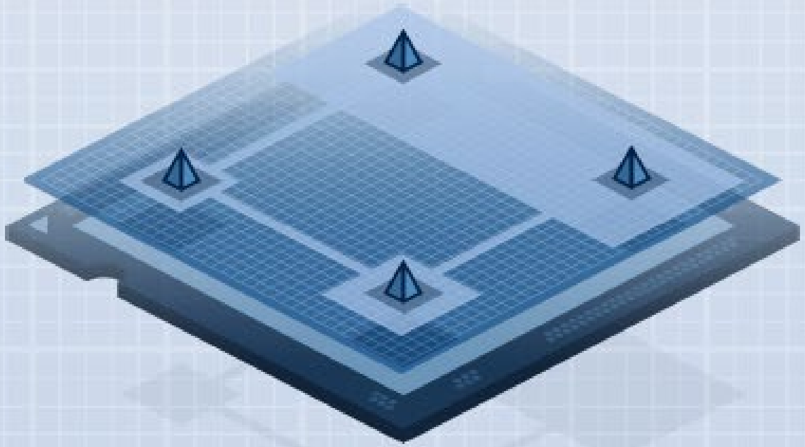
## The answer lies within **Architecture**

**Software Architecture has the greatest influence on:**

- Safety and Security Properties
- Performance
- System Comprehensibility
- Design Adaptability & Reuse
- Development Efficiency
- Novice Adoption

# Product Technologies

## Hardware Virtualization



LYNX | Secure®

## RTOS

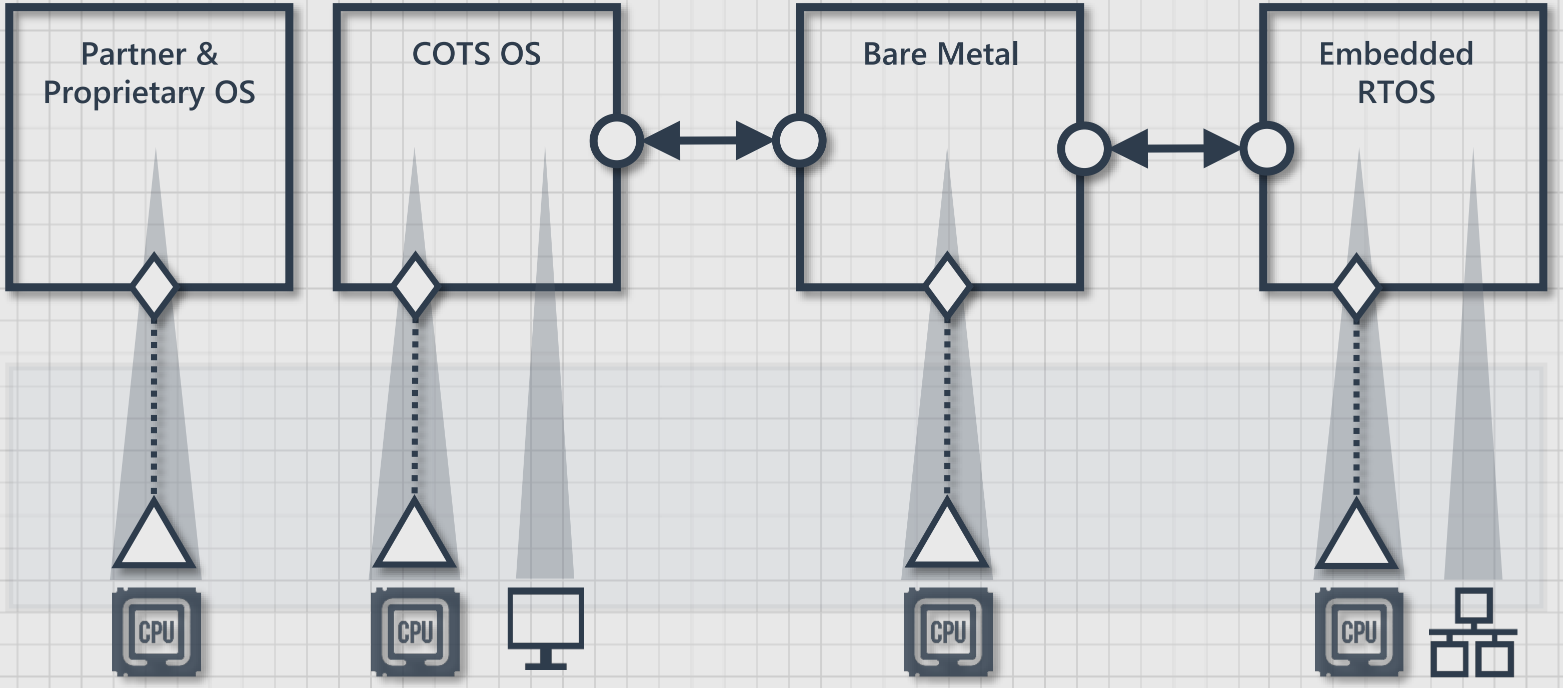
LYNX | OS-178®

## Modular Development Framework

LYNX MOSA.ic™

# LYNX Simplified Modular Integration **Framework**

- **Rooms** (AADL 'system' – with 'processor' / 'virtual processor', IO 'device', 'process' &c.)
- **Core Software** (An OS can be "virtualized" to run as a Guest Operating System)
- **Passageways** (lock-free communications for 'port', e.g., ARINC653 protocols)
- **IO/Resource Assignment** (static 'binding' of hardware resources to each Room/'system')



# AADL model of *uHardware?*

- Trying out AADL/ReqSpec for certification
- Can it express detail for MMU hardware?
  - Isn't that just a 'device'?
  - With a 'bus' and a 'processor'?
  - Can I build a 'port' for pagetable 'data'?
  - Etc.
- Asked two more questions
  - MultiCore Processor (CAST-32A): "Hardware Interference Channels"?
  - Fetching and Speculative Execution (Spectre): "Hardware bugs?"
- Want a model for *micro-architecture*

# MCP Problem Focus

## WCET *despite* Memory Latency Non-Determinism

SCP: Tightly bunched histogram

MCP: Widely spread histogram (non-deterministic)

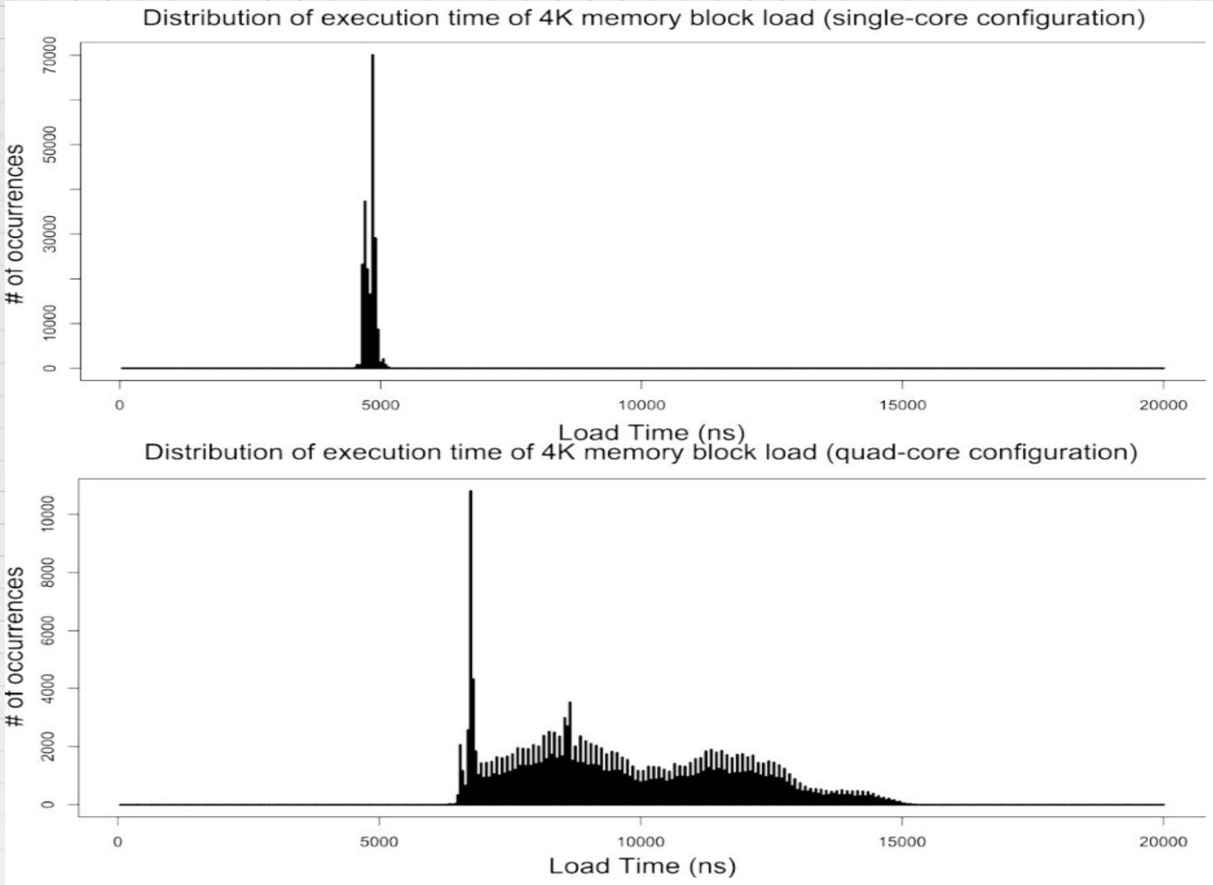
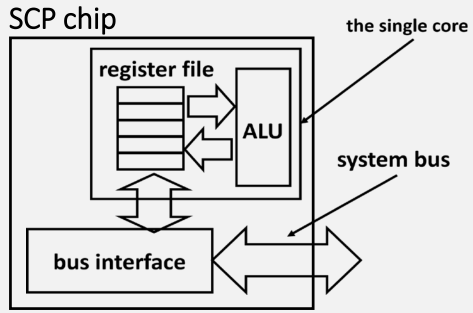


**Figure 15. Example of impact of interferences on 4K memory load operations (1 vs. 4 cores)**

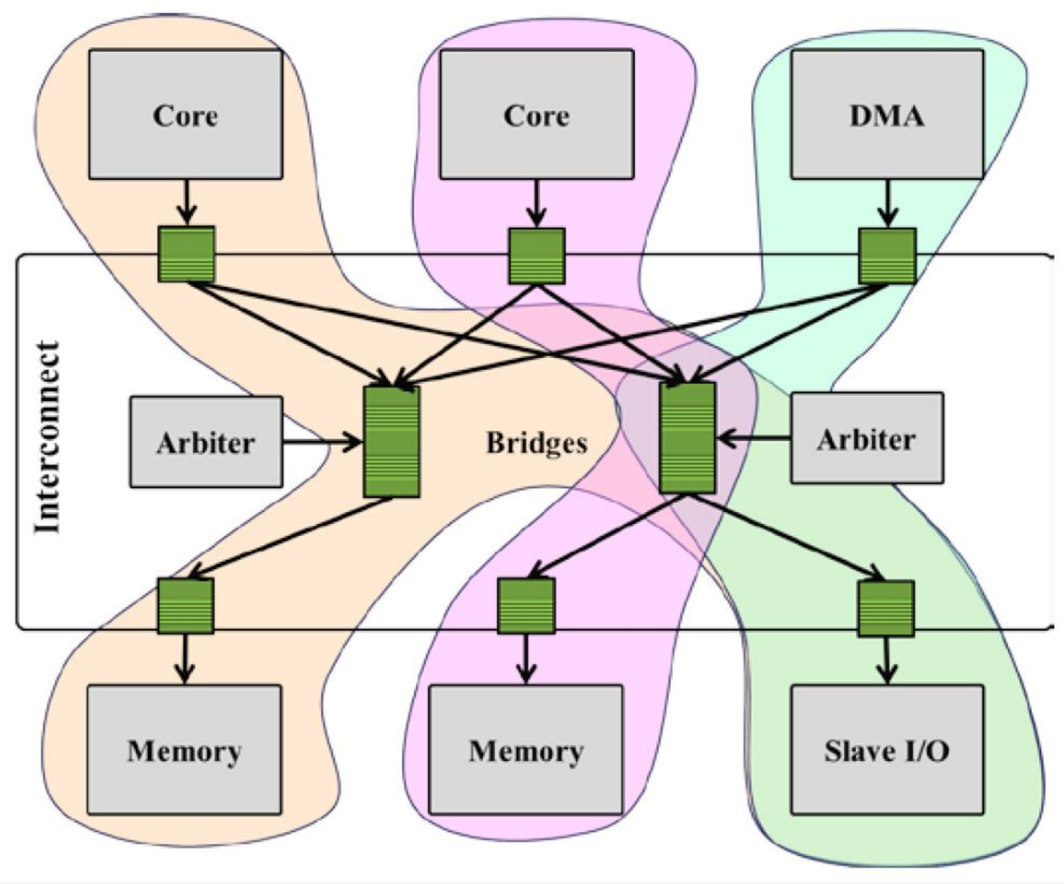
*"For both cases, we collect memory access time and application's execution time* ***while one core executes the benchmark and others execute a stressing benchmark over the same DRAM controller."***
*(FAA TC-16/51, p. 48)*

# What is the MCP Problem?

## *A Better View*



Single Core Processor

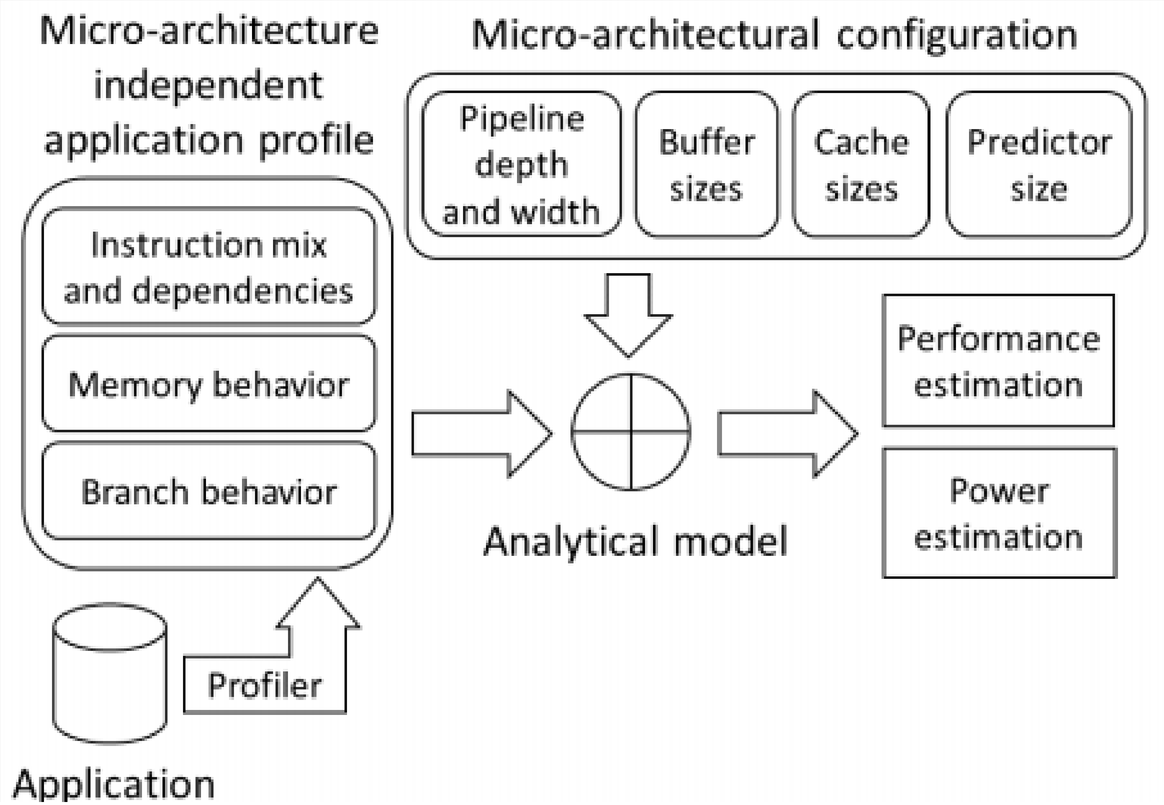


Multi-Core Processor

- Complex interconnect
  - Roundtable
  - Protocols, queues, caches, expensive, growing
  - Congestion
  - Unpredictable initiations
  - Uncontrolled initiations
- Memory competition
  - Active cores
  - Active DMA
  - Preferred lanes

Copyright Lynx Software Technologies 2018

# MCP uHW Proposed Model (Evenblij 2017)



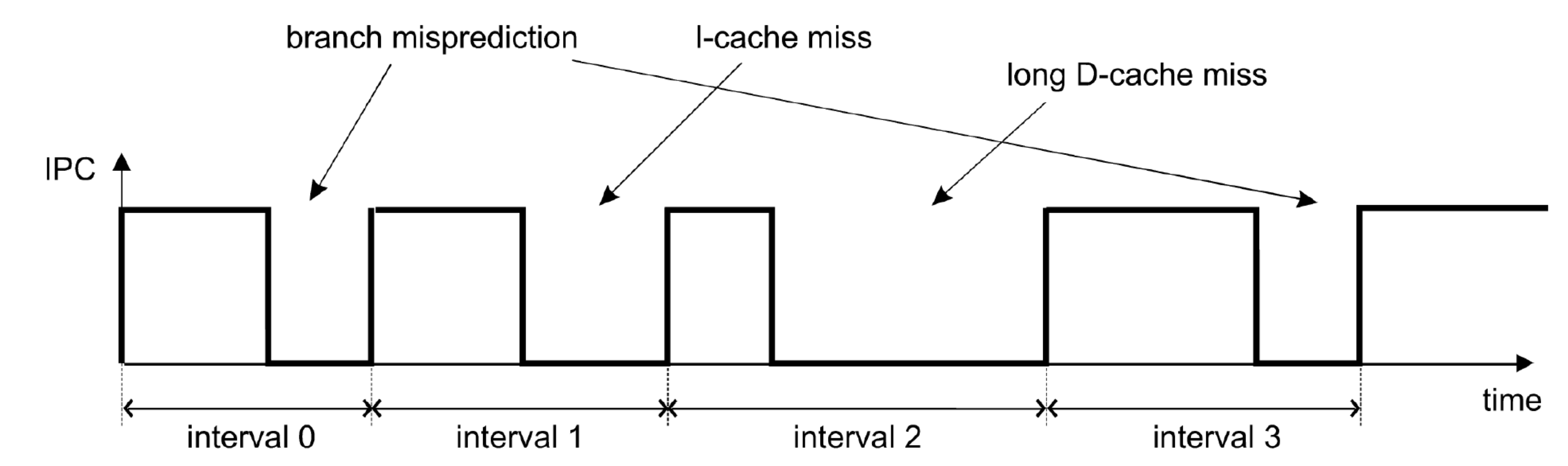
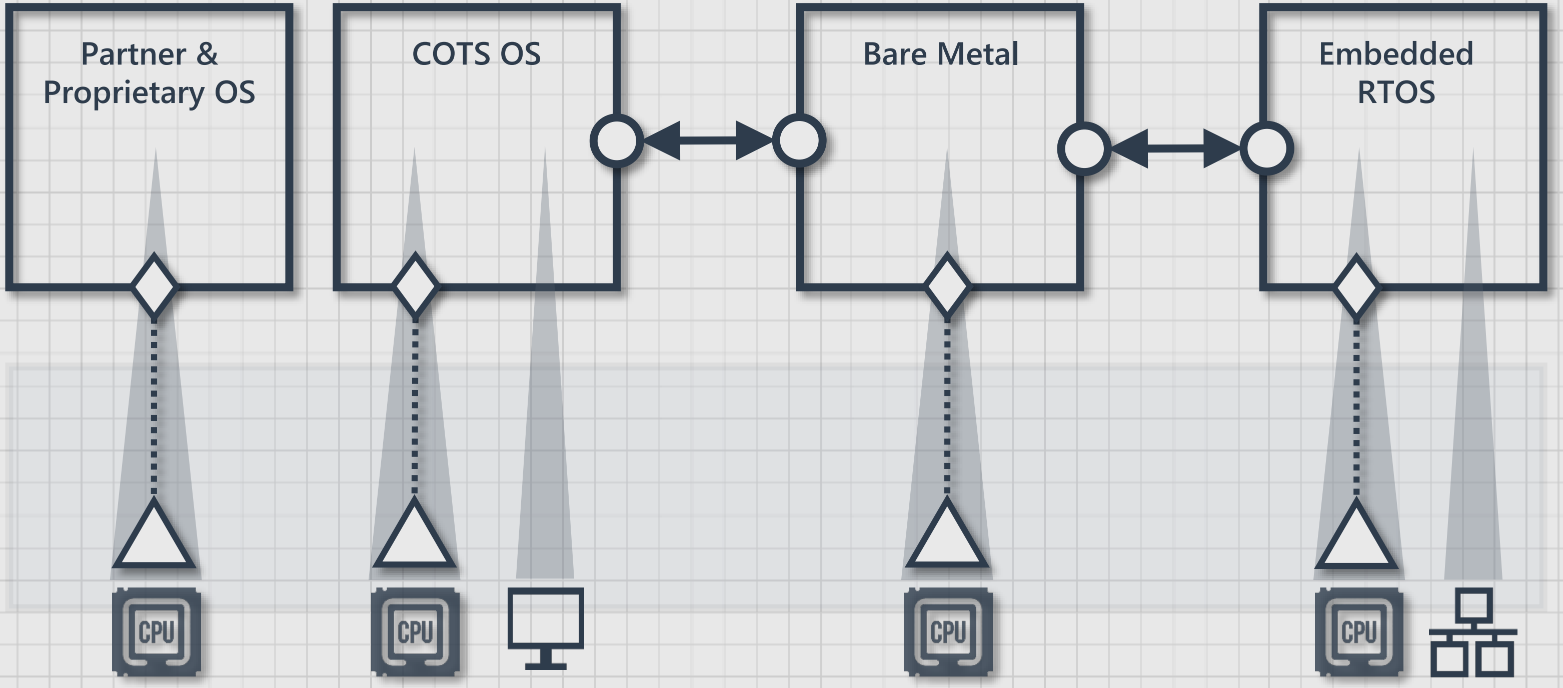


Figure 2.2: Interval analysis divides execution time in intervals between miss events [1].

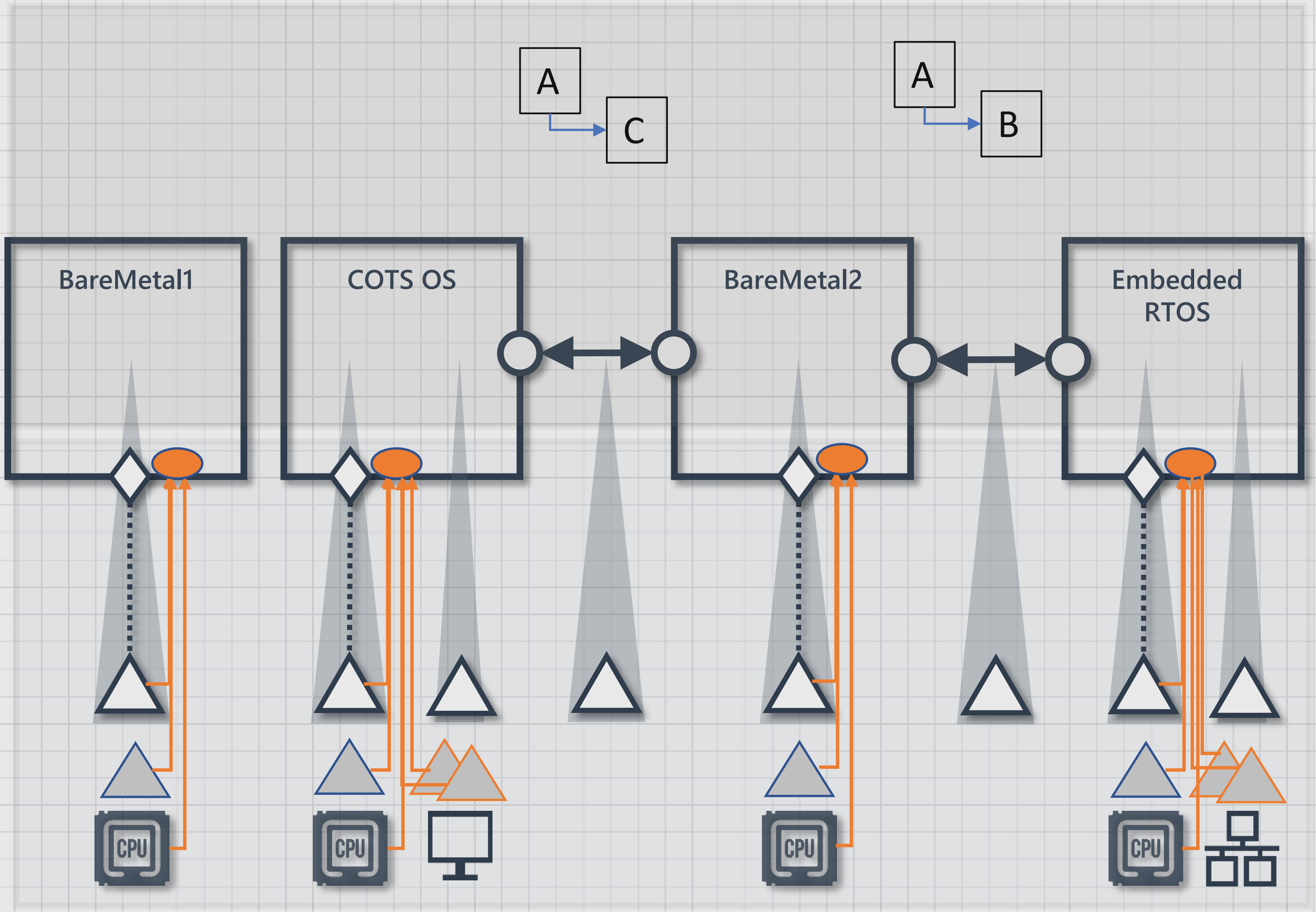
- **Rooms** (AADL 'system' – with 'processor' / 'virtual processor', IO 'device', 'process' &c.)
- **Core Software** (An OS can be "virtualized" to run as a Guest Operating System)
- **Passageways** (lock-free communications for 'port', e.g., ARINC653 protocols)
- **IO/Resource Assignment** (static 'binding' of hardware resources to each Room/'system')

# LYNX Simplified Modular Integration **Framework**



# LYNX Simplified Modular Integration **uHardware** Framework

## Health Monitor / Error Model (types, propagation, behavior, repair,…)



Data Model (types)

Behavior Model (types)

Config'd uHWEvents

uHardware Mapping DataModel & Configs

VM Mapping uHardware

Config'd HW IOdevs

Processor uConfigs

# AADL model of *uHardware?*

- Two questions
  - **MultiCore Processor** (CAST-32A): "Hardware Interference Channels"?
  - **Spectre**: "Hardware bugs?"
- AADL *micro-architecture* model:
  - '**system**' – one per Room; configured container
  - '**processor**' – "contains" nearly all of the problems
  - '**bus**' – use these to compose uHW into graph (topology)
  - '**device**' – uHW; composed using buses
  - '**data**' – used to configure uHW, e.g., pagetable
- *Not* HDL code (right?)

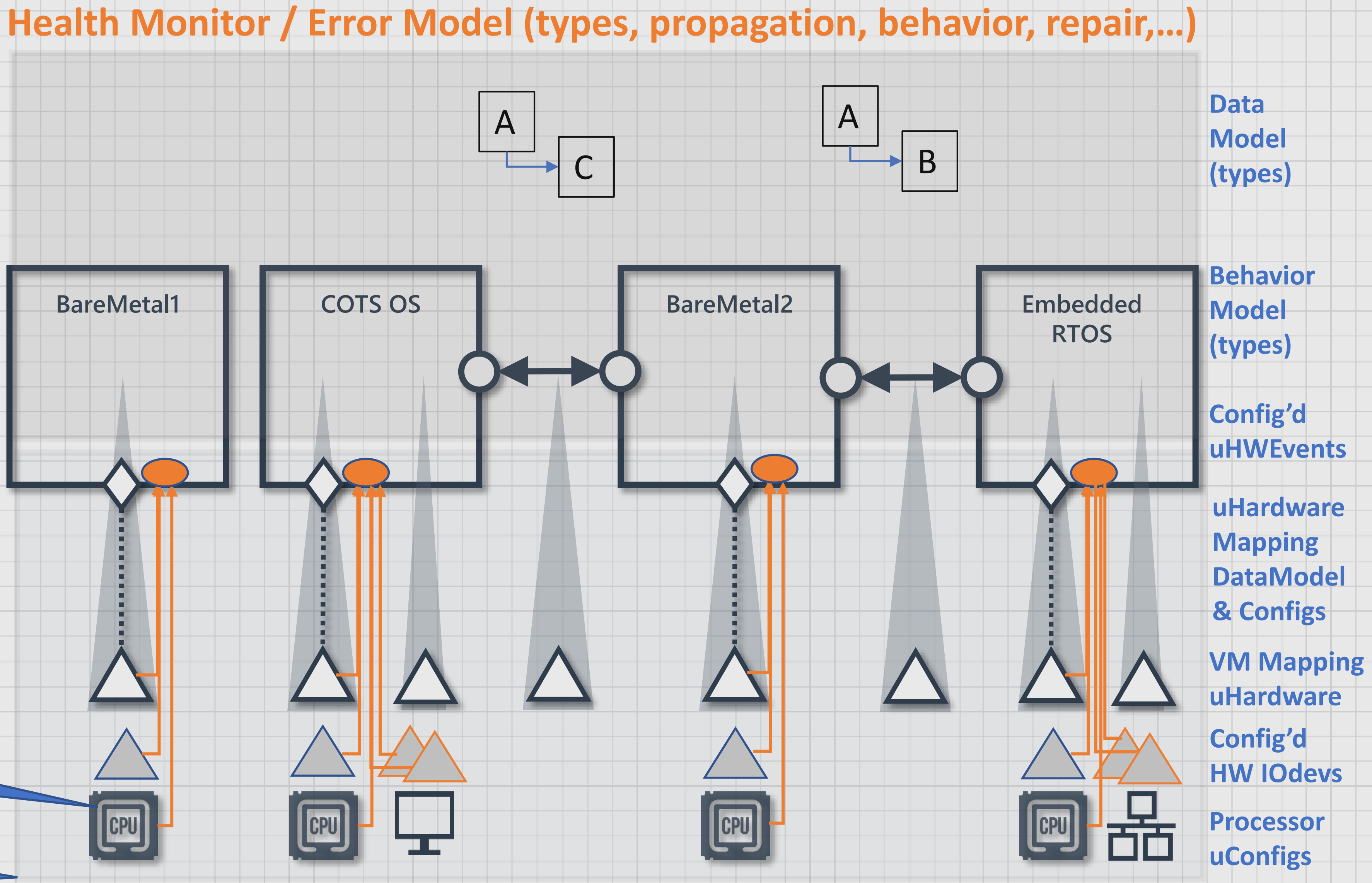
# Initial Requirements (1)

- CAST-32A Hardware Inteference
  - Model a *HW Topology* for memory access
    - from processor core
    - through caches
    - through interconnect (cache coherent fabric)
    - through chokepoints
    - to DRAM device
  - Model the *memory access latency*
    - Cache “hit” means good; negative latency
    - Cache “miss” bad; latency added
    - Latency accumulates per node walked in the graph!
  - Model latency as EMv2 ‘*error*’
    - errors and data ‘propagate’ via a ‘port’
    - total accumulation of latency (store/load round-trip)
    - from request, through HW Topology, to satisfy (port)

# Initial Requirements (2)

- Spectre Hardware Bug (Kocher et al.)
  - Model a *HW Topology* for memory access
    - in a processor core, chip-specific
    - via core-local buses
    - including all relevant core-local uHW
    - query for sufficient conditions-to-trigger
  - Model the *uHW that fails to perform access check*
    - We've landed in "reverse engineering" territory!
    - (i.e., Google, Kocher, et al. found a fault / root cause)
    - Increasingly need to collaborate with origin labs
    - Increasingly need to collaborate with chip makers
  - Model uHW conditions that activate / trigger the fault
    - Some in uHW, e.g., TLB / Cache, Speculative Execution Unit..
    - Some in system-level software state, e.g., MMU page tables

LYNX's **uHardware** AADL Library



Spectre &c.

CAST-32A

# Observations (1)

## What's the *purpose* for uHW modeling?

- Is AADL for Teamwork or for Analysis?
  - published model (from chipmaker TRM)
  - private, reverse-engineered model (probably not right)
  - private AADL model ***query***?
  - Observation: AADL is a standard, which promotes **teams**
- How can AADL standards promote teamwork?
- Does the AADL System Integrator want:
  - Better hardware models?
  - Hardware-resource error models?
  - Pre-built certification templates?
  - Can these expensive "options" align with "faster & cheaper?"

# Observations (2)

## What's the "Customer Demand"

(for a simple, modular, AADL uHW Library)?

- **"Do it for me"** – wants to buy the uHW model
  - uHW AADL 'device', 'bus', 'data'/Data model, 'ports', etc.
  - "Just to look at" – to analyze, query, & re-use
  - "I don't want to pay a big AADL cost-to-entry"
- **"Hold the errors"** – doesn't want any!
  - Some errors are EMv2-repaired ('error sink')
  - Some errors always 'propagate'
  - Some 'error behavior' should be extensible/overridable
  - Observation: hardware resources as a primary 'error source' (e.g., 'ARINC653::HM_Error_ID_Actions', 'Error_Level_Type', etc.)
- **Transparency** – about our LynxSecure requirements
  - **SW**: validation needs to know how well it works (i.e., the partitioning architecture's internal requirements)
  - **HW**: hardware needs software to register the event handler (as an integration requirement)
  - **Data**: A customer can configure partitioning for safety – *or not*! (i.e., configuration data could specify poor requirements)

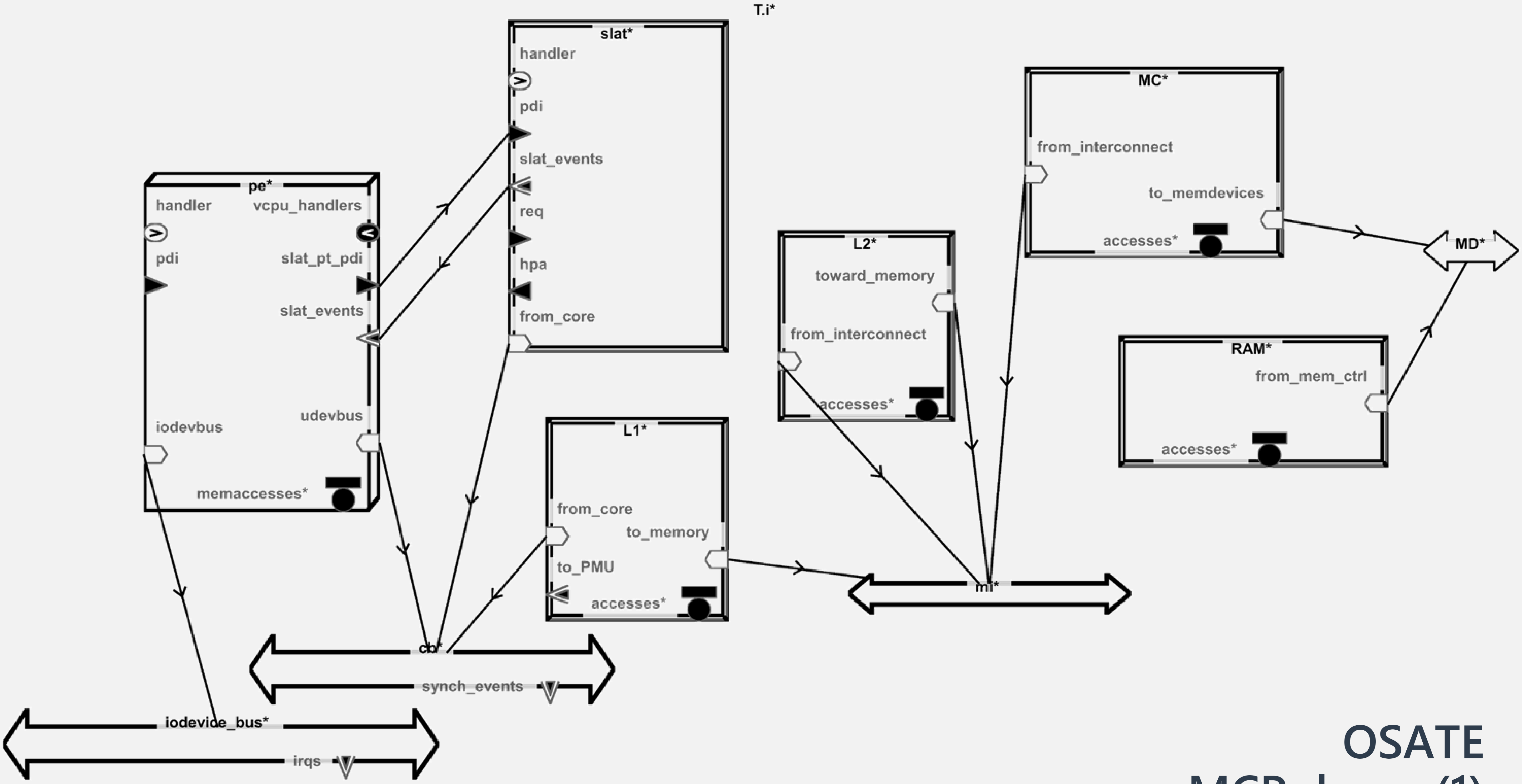
# Opportunities/ Current Goals

Best practices

- OSATE provides formal methods
- Requirements engineering?
- Certification templates?

Prebuilt AADL models “below the HAL” (uHW)

- Prebuilt repository for AADL-style requirements & model
- Prebuilt models for uHW resources
- Model for MCP partitions
- Working error model



# OSATE
# MCP demo (1)

# OSATE demo (2) “partitioning/”

- Abstract Model
  - Hardware (hw)
  - Software (handlers)
  - Data (pdi)
  - Hardware abstraction layer (resources)
  - Abstract HW concepts (control_type)
- ReqSpec requirements
  - for Hardware
  - for Software
  - for Data
  - to define/explain concepts
    - control_type
    - resources

# OSATE demo (3) “aarch64/”

- HW-specific model (aarch64)
  - hw
  - handlers
  - pdi
  - resources
- ReqSpec for aarch64
  - As before: {hw, handlers, pdi}
  - This time, traces to ARM TRMs
  - This time, defines “resources” precisely, in terms of HW TRM definitions
  - Defines what a configured ‘system’ means

# OSATE demo

# What's missing? (1)

## Automated AADL analysis for MCP?

- **Start**: data used to configure uHW blocks
- **Add**: hardware reverse engineering (chip vendor)
  - This is expensive; I cannot share
  - Basic idea is to measure each uHW block
    - What can possibly trigger latency?
    - What's the latency cost, per uHW block?
    - What's the worst case (consecutive losses…)
  - Goal is to measure all uHW blocks (AADL properties)
- **Add**: software memory use metrics (customer)
  - Record memory use profile per thread/SW
  - Find the worst possible competitions, given uHW + SW
- **Deploy**: HW interference mitigations (HW+SW+data)
  - Break up the fights! (dynamic, not AADL-static)
  - Prove that the new worst case is better (how much?)

# What's missing? (2)

## AADL analysis for MCP – by parts

- **Start**: data used to configure uHW blocks
- **Added**: hardware reverse engineering (chip vendor)
- **Measured**: software memory use metrics (customer)
  - Record memory use profile per thread/SW
  - Find the worst possible competitions, given uHW + SW
- **Modeled**: HW interference mitigations (HW+SW+data)
  - Break up the fights! (dynamic, not AADL-static)
  - Estimate: mitigated case is better (how much?)

# Suggestions for AADL Committee (1)

## **Anticipate** Pre-built AADL models

- concept: AADL libraries, with reqspec, etc.
- customer “just wants to build/integrate software”
- reusable by AADLv3 Configuration
- e.g., ‘Customer1 system extends Lynx::Room’
  - was ‘Customer1 system’ (no extends)
  - or was ‘Customer1 system extends Other::System’
- e.g., ‘message1 data extends balsa_data_model::EGI_Data_Platform’

# Suggestions for AADL Committee (2)

**Anticipate** AADL models for HW uArchitecture

- concept: AADL libraries, with reqspec, etc.
- customer “just wants to build/integrate software”
- library provides a model for some questions
  - reusable by inheritance
  - query-able (new formal methods queries)
  - configure-able (AADL configuration to quickly represent a chip or a target board or larger system, but with HiFi uHW)
- Promote teaming
  - reverse engineering of realistic chip models
  - collaboration with HW chip manufacturers, “Please confirm..”
  - encode best practices for/toward solutions
- Collaboration via git – or just a zipfile/tarball

# Suggestions for AADL Committee (3)

## **Anticipate** AADL models as certification repositories

- concept: AADL libraries, with reqspec, etc.
- customer wants the least possible:
  - certification headaches (examples, Configuration, reports)
  - HW / uHW / MCP headaches
  - customer just wants to build software
- library provides a template-model for best practices
  - reusable by structure (directories, packages, etc.)
  - per regime / per set of best practices (Agile-style?)
  - Architecture-led
- Promote education / learning
  - encode best practices for/toward solutions
  - some learn best from an example
- Collaboration via git – or just a zipfile/tarball

# Suggestions for AADL Committee (4)

## How best to model uHW?

- New ‘device’ – inside / outside of ‘processor’?
  - All memory uHW affects every ‘thread’, ‘process’, etc.
  - Every uHW event switches the thread SW
    - A context switch to OS always pollutes the caches
    - A context switch from OS to not-me pollutes caches
- concept: AADL “zoom”
  - out: “just a processor”
  - in: uHW exploded view (breakdown into components)
- virtual vs. (not) processor / bus / memory

# Call to Action

- Do you agree that:
  - AADL can/should be used to model *Hardware microarchitecture*?
  - The AADL standards, together with reqspec & friends, can be used to promote teamwork?
  - Teamwork will help with thorny issues like safety, security, reliability?
- How can you / your organization help?
  - Not everyone has expertise here
  - One solution may not fit all
  - Developing world-class exemplars may help most
- Do you want to
  - team with Lynx as we build up some exemplars?
  - obtain rich, detailed, automatable models of uHW?

Thank You.