Transforming MBSE Models into Formally-Verifiable Language to Support Test and Evaluation as a Continuum

Jerome Hugues and Dio de Niz

Software Solution Division

Assuring Cyber-Physical Systems

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Formal MBSE for Test-as-a-Continuum

Shift Testing Left with MBSE

- Reduces test activities iff models used across whole system V

- *Today:* gap between high-level and low-level models

- *Future:* joint formal model/analysis capabilities across system lifecycle

Lessons Learned – Shifting Left and Supporting Test Objectives

- Lesson 1: virtual integration using SAE AADL

- Lesson 2: combining models and DevOps: ModDevOps

- Lesson 3: prove first, test later

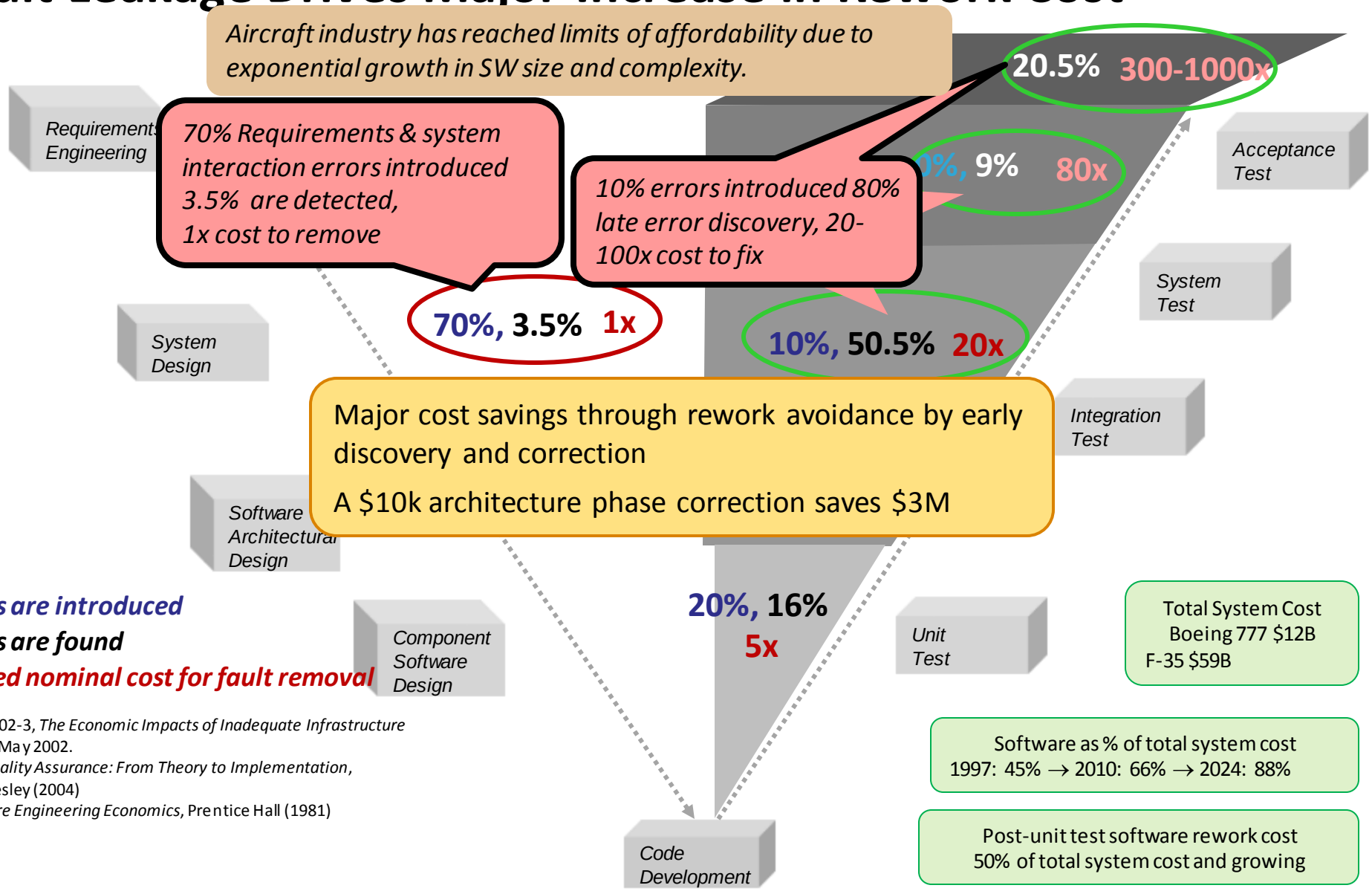
Lessons Learned – Shifting Left and Supporting Test Objectives

- Lesson 1: virtual integration using SAE AADL

- Lesson 2: combining models and DevOps: ModDevOps

- Lesson 3: prove first, test later

High Fault Leakage Drives Major Increase in Rework Cost



Sources:

NIST Planning report 02-3, *The Economic Impacts of Inadequate Infrastructure for Software Testing,* May 2002.

D. Galin, *Software Quality Assurance: From Theory to Implementation*, Pearson/Addison-Wesley (2004)

B.W. Boehm, *Software Engineering Economics*, Prentice Hall (1981)

The Safety Critical Embedded Software System Challenge

Problem:

- Software increasingly dominates safety and mission critical system development cost
- **80% of issues discovered post unit test**

Inception:

Model-based virtual testbench: joint virtual integration testing and incremental analytical assurance

Solution:

- Technology based on SAE International standard matured into practice through pilot projects and industry initiatives
- Open source research prototyping platform continually enhances analysis, verification, and generation capabilities
- Direct alignment with DoD Digital Engineering Strategy

Before You Even Write a Line of Code…

- AADL: design the entire system and identify problems
- Redesign to eliminate errors
- Perform virtual integration of software, hardware, and system to identify problems early

About AADL

- SAE Avionics AADL standard adopted in 2004
- Focused on embedded software system modeling, analysis, and generation
- Strongly typed language with well-defined semantics
- Used for critical systems in domains such as avionics, aerospace, medical, nuclear, automotive, and robotics

RAH-66 COMANCHE SOFTWARE REWORK & INTEGRATION COSTS

Photo Credit: Boeing-Sikorsky

Two major software (SW) rebuilds occurred during development indicating significant integration issues

- **1st increment: 75% of SW replaced**
- **2nd increment: 50% of SW replaced**

- ***In 1983, the Army planned to buy 5,023 vehicles at $12.1 million/copy.***
- ***Test schedule delays and increasing development costs scaled down the planned buy to 650 aircraft at $58.9 million/copy.***
- ***Most testing involved integration of the complete Mission Equipment Package, which incorporated a radar, infrared, and image-intensified television sensors for night flying and target acquisition.***
- ***Technical challenges remained in software development, integration of mission equipment, radar and infrared signatures, and radar perf.***
- ***The first flight had been originally planned to take place during August 1995, but was delayed by a number of structural and software problems that had been encountered.***
- ***Key program elements, including development and integration of certain software capabilities, failed to foster confidence with Army overseers; several capabilities were viewed as having been unproven and risky.***
- ***The anticipated consumption of up to 40% of the aviation budget by the Comanche alone for a number of years was considered to be extreme.***

References:

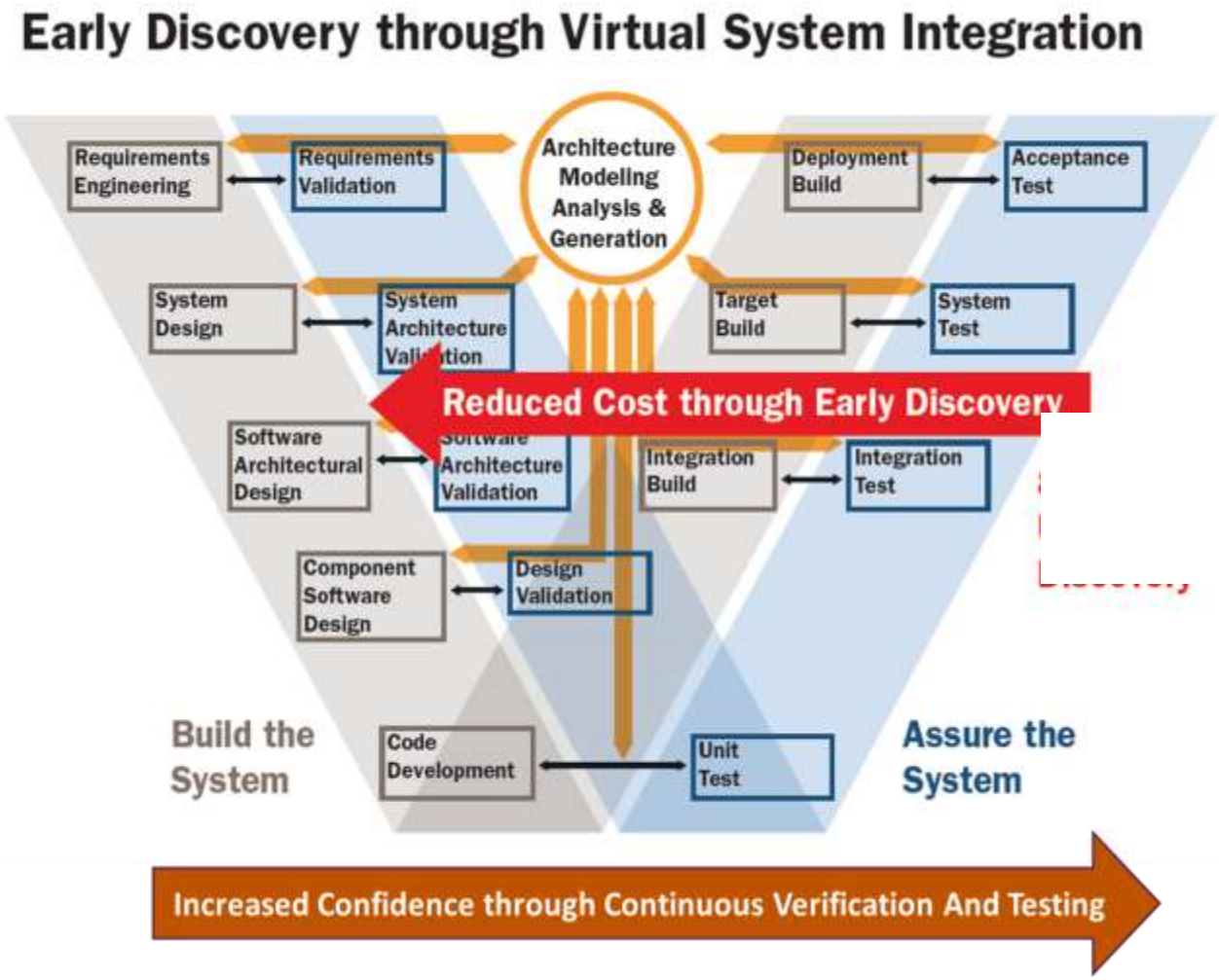
- *http://www.defense-aerospace.com/articles-view/release/3/32273/pentagon-hit-over-comanche-failings-(jan.-23).html*
- *https://en.wikipedia.org/wiki/Boeing%E2%80%93Sikorsky_RAH-66_Comanche#cite_note-26*
- *https://en.wikipedia.org/wiki/Boeing%E2%80%93Sikorsky_RAH-66_Comanche#cite_note-Eden_p139-9)*

Comanche costs were expected to consume up to 40% of US Army Aviation budget resulting in cancellation. Integration and software rework were significant cost contributors.

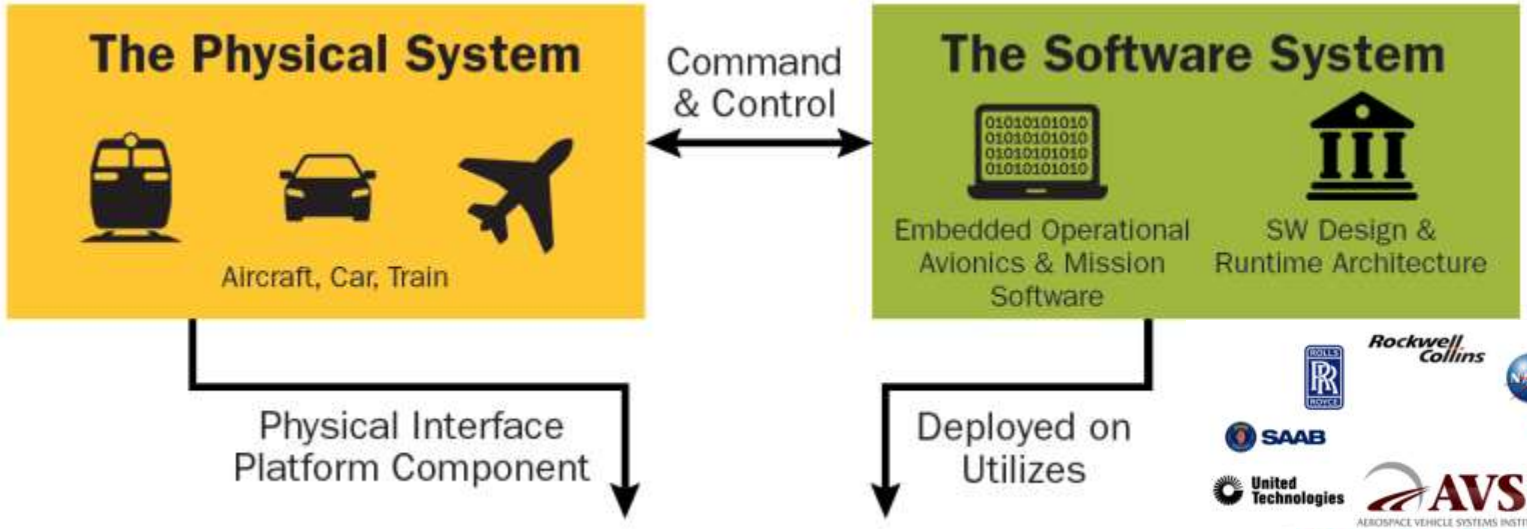
APPROVED FOR PUBLIC RELEASE

Architecture-Centric Virtual Integration Practice

- Supports verification, airworthiness, safety, and cybersecurity
- Targets software-intensive parts of real-time safety- and security-critical computing systems
- Provides standard analyzable and processable architecture description for embedded systems
- Acquisition and Engineering handbooks
- Enabler for MOSA



AADL Standard Targets Embedded Software Systems



SAE International

AS 5506 Standard Suite

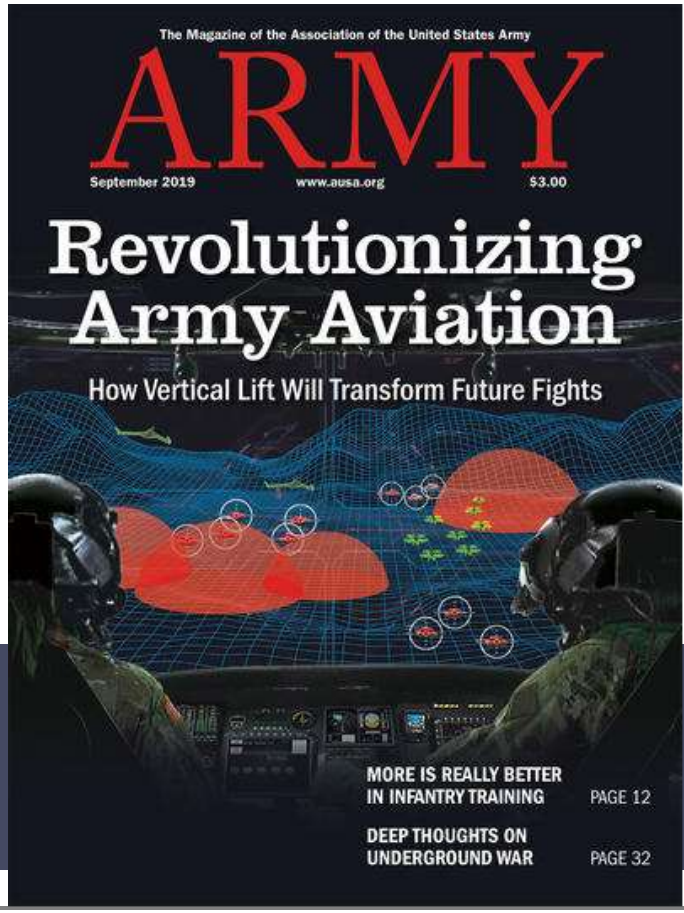
Standards provide long-term industry-wide solutions to support multi-organization model-based engineering

In 2008 Aerospace industry initiative chose AADL over SysML and other notations as it specifically addresses embedded software systems

AADL captures mission and safety critical embedded software system architectures in virtually integrated analyzable models to discover system level problems early and construct implementations from verified models

Revolutionizing Army Aviation through Vertical Lift Challenge

- Decreased fielding time: found problems early
- Early risk reduction: discovered performance issues early
- Increased cybersecurity by improving system security
- Decreased development costs and MOSA support



Virtual integration of software, hardware, and system supports verification, airworthiness, safety, and cybersecurity certification

Revolutionizing Army Aviation through Vertical Lift Challenge

Finding Problems Early (AMRDEC/SEI)

Summary: 6 week virtual integration of health monitoring system on CH47 using AADL

Result: Identified 20 major integration issues early

Benefit: Avoided 12-month delay on 24 month program

CH47 Chinook

DARPA

High Assurance Cyber Military Systems (HACMS)

Unmanned Quadcopter

Unmanned Little Bird

TARDEC Autonomous Truck

Improving System Security (DARPA/AFRL)

Summary: AADL applied to unmanned aerial vehicles & autonomous truck

Result: AADL models enforced security policies and were used to auto-build the system

Benefit: Combined with formal methods verification, prevented security intrusion by a red team

---

Transforming procurement (Joint Multi-Role)

Summary: Industry/DoD process demonstration

Result: Pre-integration fault identification

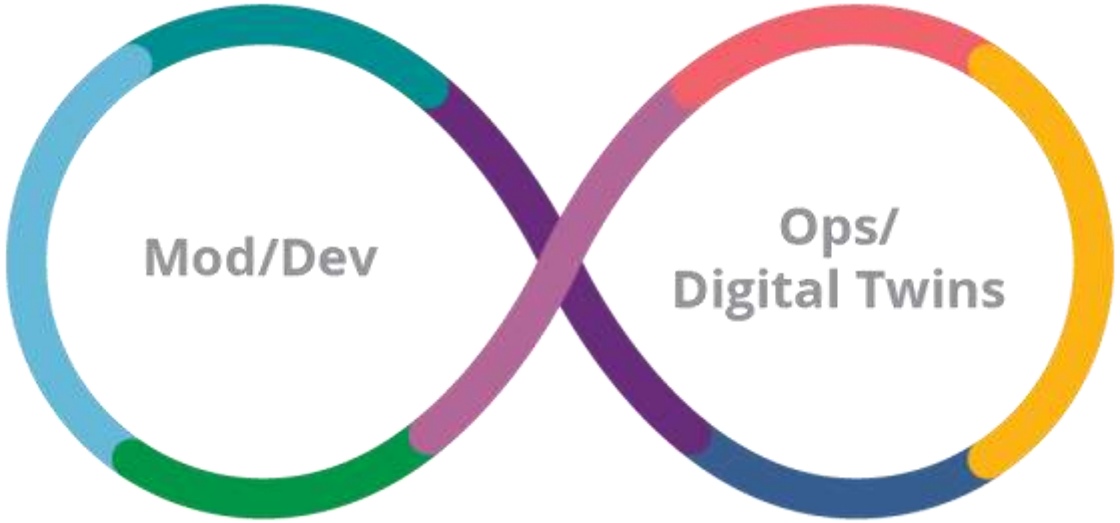
Benefit: 10X reduction integration test cost

Makes complex capabilities possible through Agile analytic and virtual integration of real-time safety and security critical cyber physical embedded systems

Lessons Learned – Shifting Left and Supporting Test Objectives

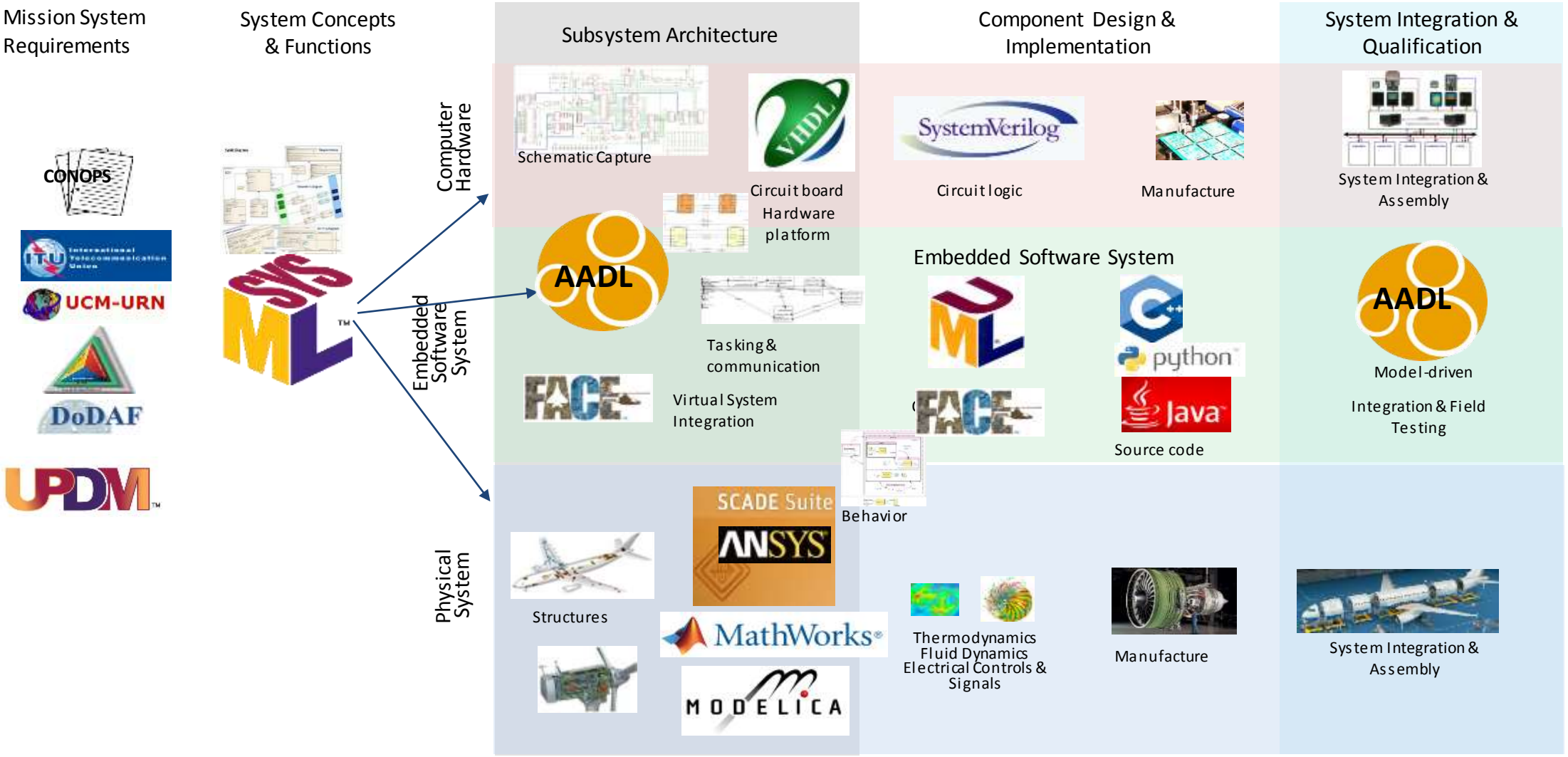
- Lesson 1: virtual integration using SAE AADL

- Lesson 2: combining models and DevOps: ModDevOps



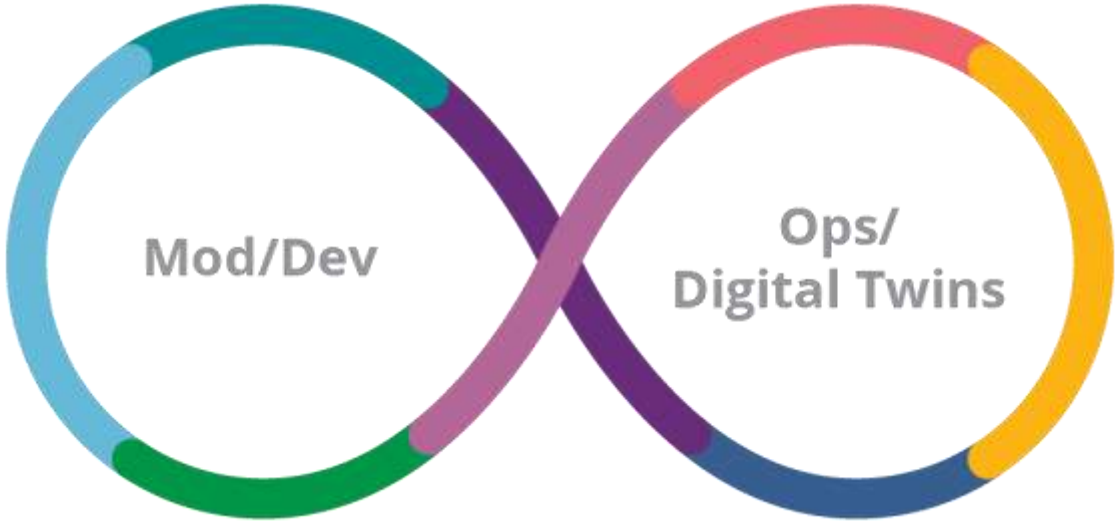
- Lesson 3: prove first, test later

Filling the Modeling and Analysis Gap for Embedded Software Systems



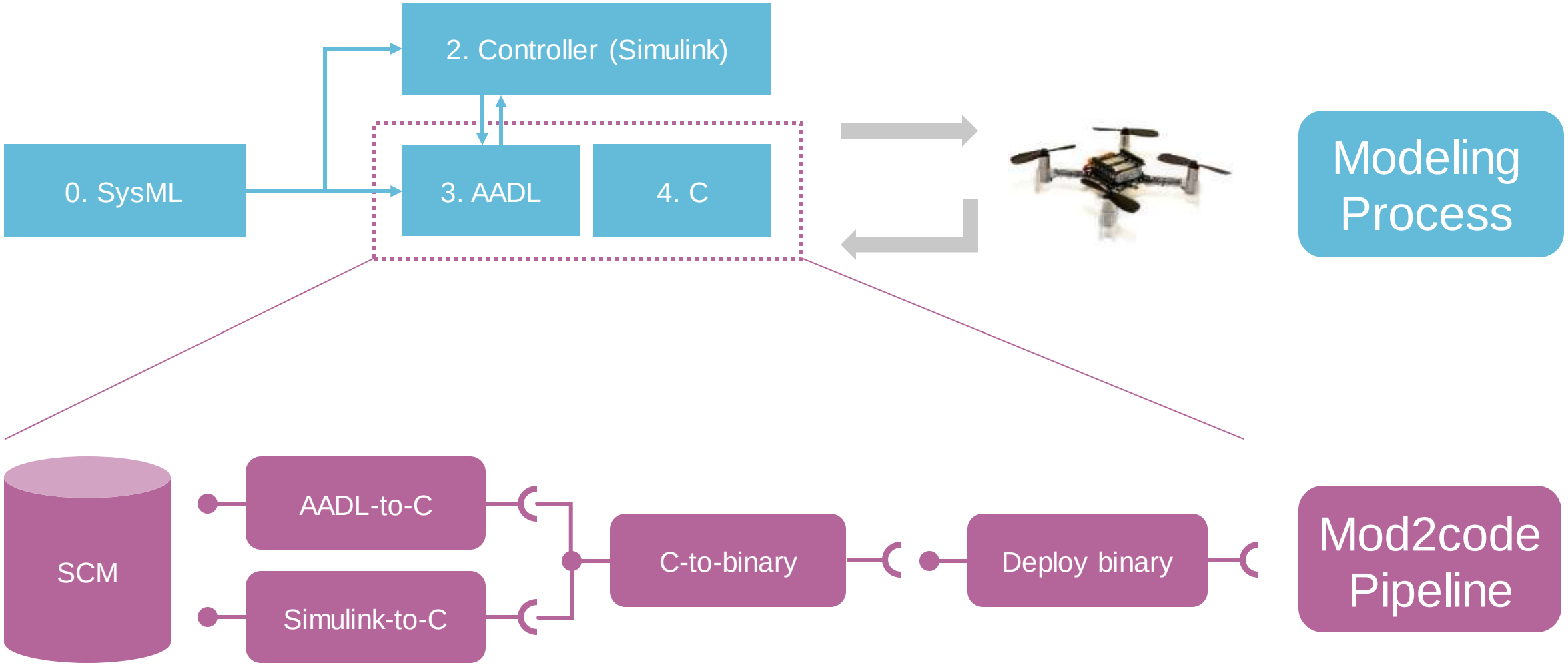
From DevOps to ModDevOps

- DevOps delivers software faster with increased quality
  - Continuous integration and development
  - Containerized systems
- DevOps is adaptable to systems

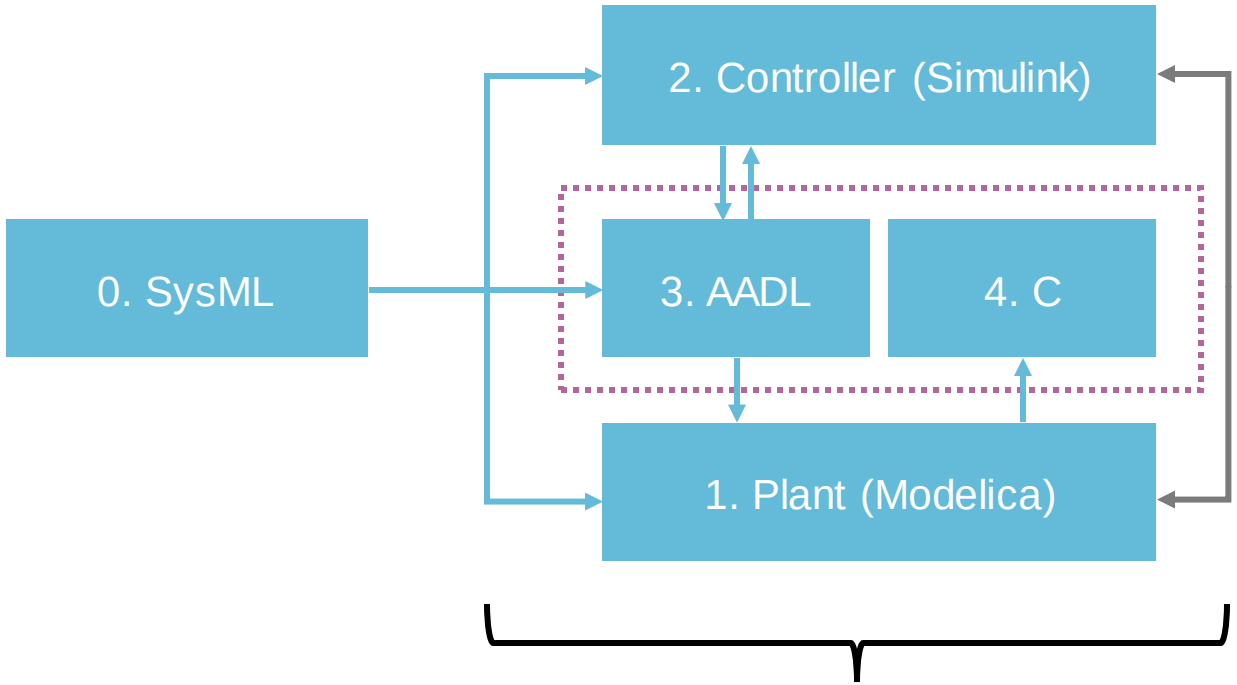


*"ModDevOps is **a systems/software** co-engineering culture and practice that aims at unifying **systems engineering (Mod),** software development (Dev) and software operation (Ops). The main characteristic of the ModDevOps is to strongly advocate **abstraction,** automation, and monitoring at all steps of system construction."* (adapted from https://software.af.mil/training/devops/)

ModDevOps in Action: Modeling Process



From ModDevOps to TwinOps



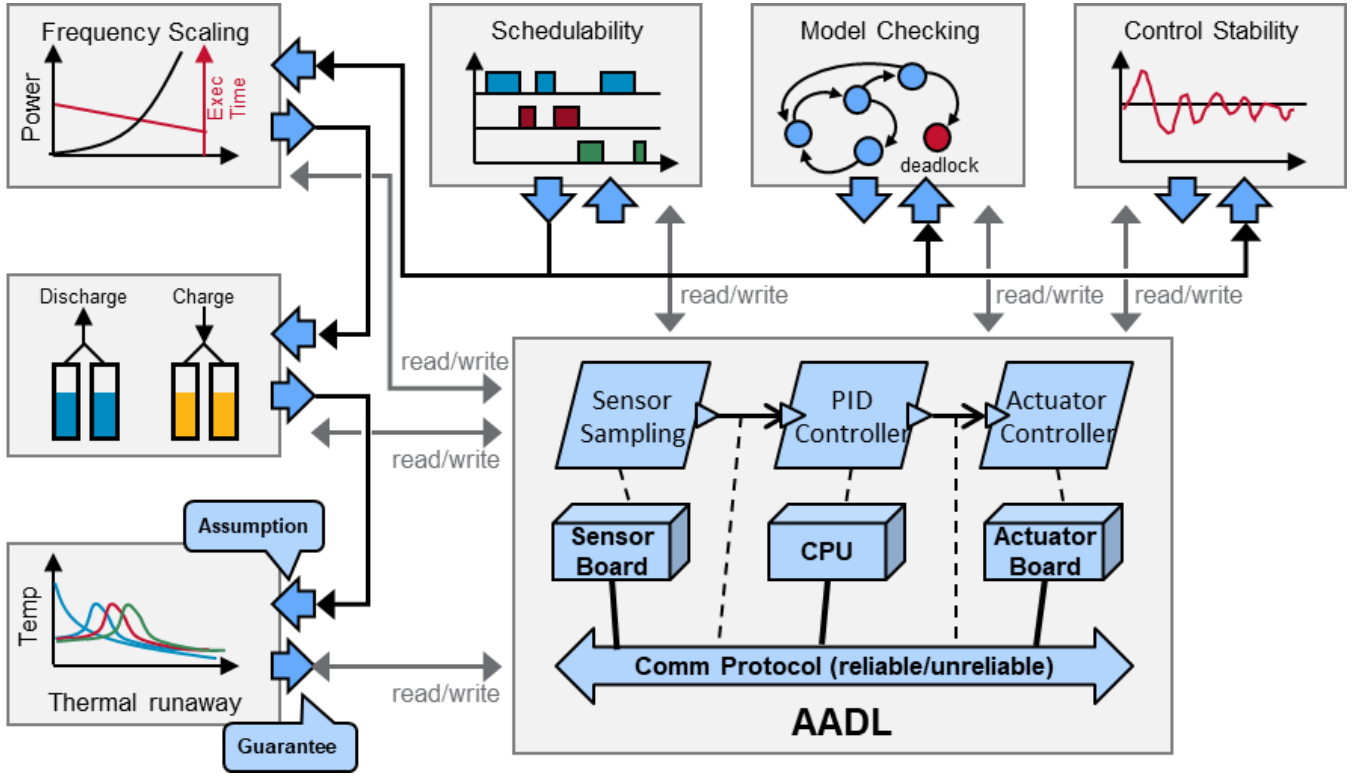
Digital Twins

1-2-3-4: “mega-modeling” V&V
- 1-2: HLR validation
- 2-(3+4): validation of LLR
  1+(3+4): virtual integration

Digital Twins of UAV vs. UAV flying: validation of Modelica model, efficiency of the controller (overshoot verification) and timing verification of software.

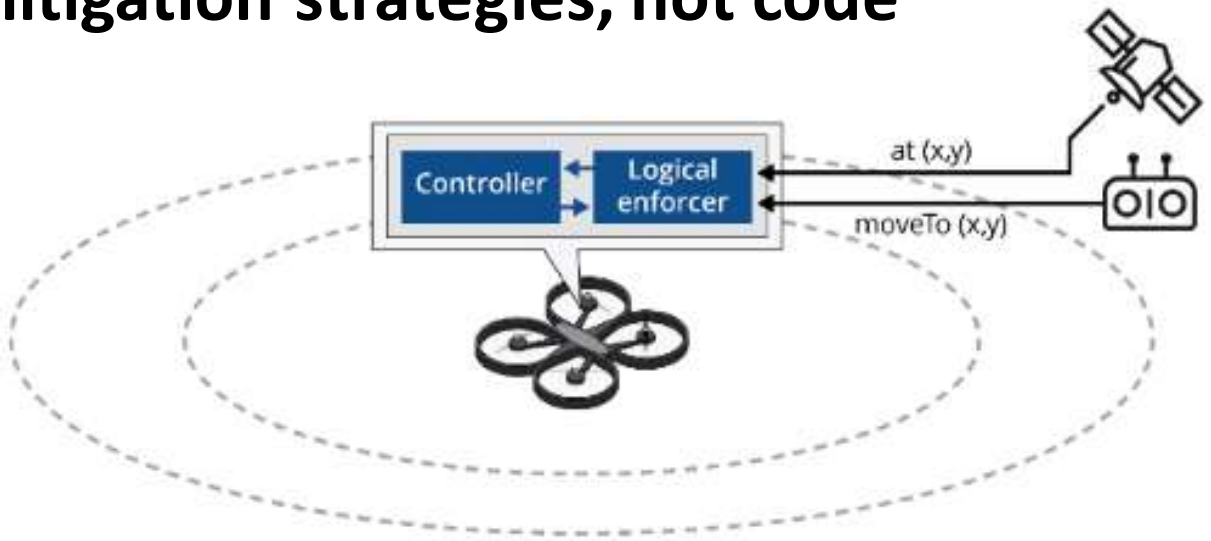
Lessons Learned – Shifting Left and Supporting Test Objectives

- Lesson 1: virtual integration using SAE AADL
- Lesson 2: combining models and DevOps: ModDevOps
- Lesson 3: prove first, test later

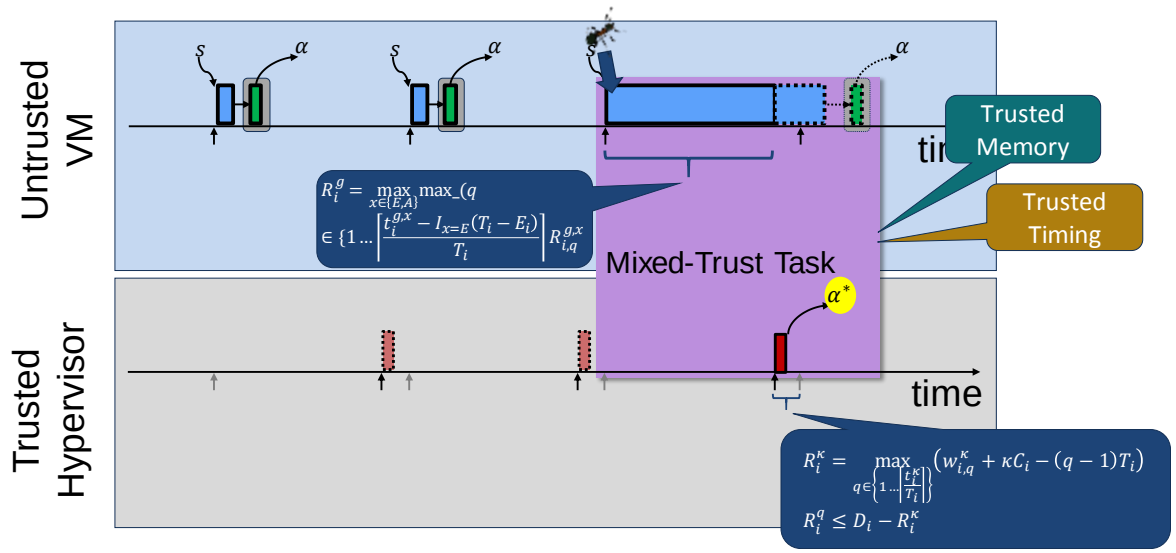


Proofs to reduce test space: verify mitigation strategies, not code

- Problem: hard-to-verify complex controllers



- Solution: isolate and verify unsafe behavior
  - Timing
  - Physical Effects
  - Hypervisor and composition logic

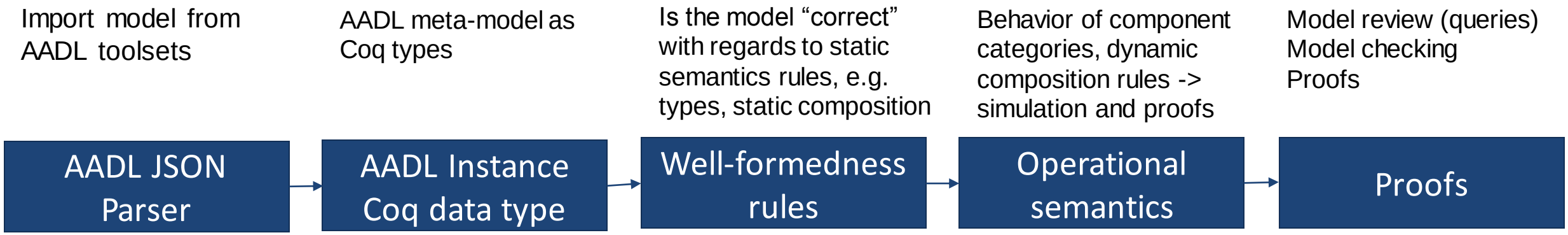


Proof at the Level of MBSE Artefacts

- Problem: semantics not specified in modeling language. How can we formally verify?

- Solution: provide an unambiguous formal semantics for AADL

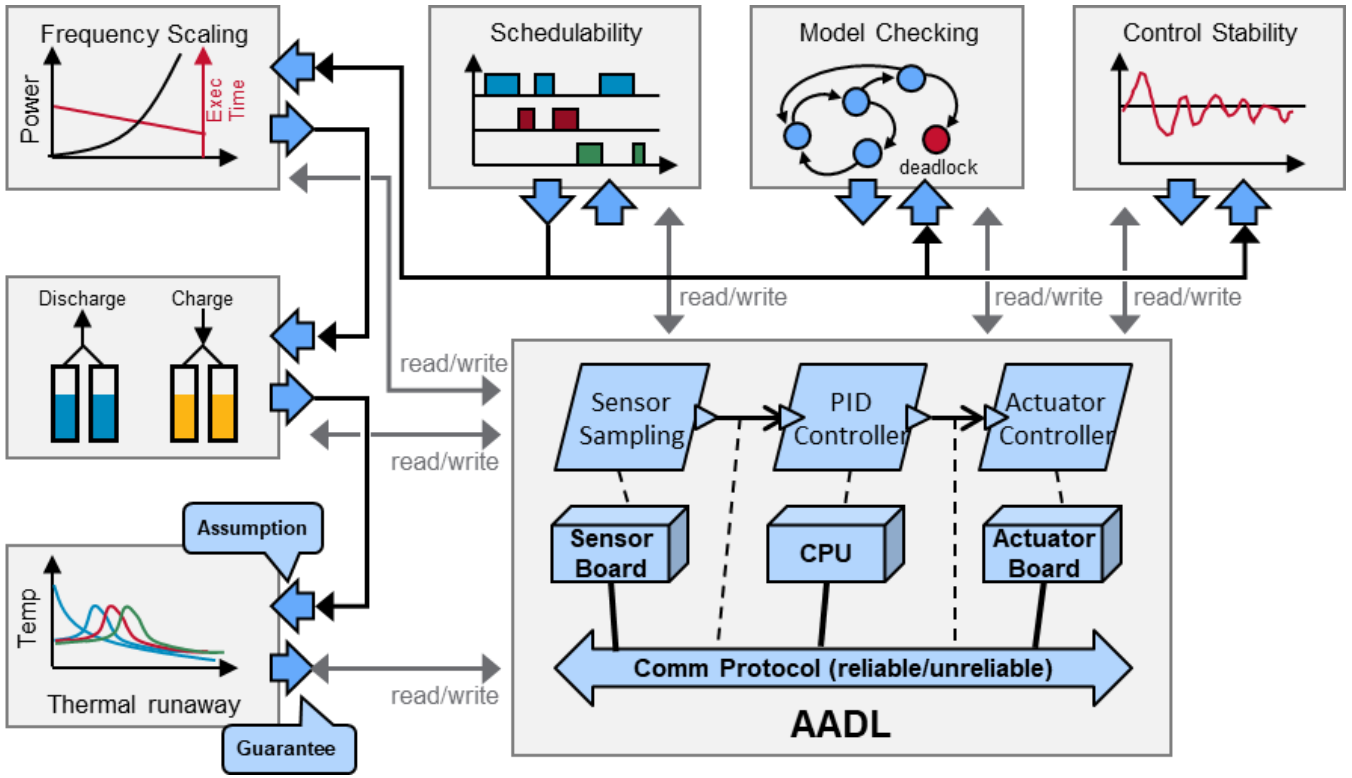
- ***Creates a correct-by-construction verification pipeline***



Proofs to Reduce Artefact Review

Models can be verified using manual review or automated analysis

- Problem: ensure we use the right tool for the right model
- Solution: verify analysis assumptions and guarantees using formal contracts
  - Validate assumptions
  - Combine multiple analyses
  - Resolve assumption conflicts



Wrap-Up

The Science: foundations for combining models; leverage formal methods to "shift left"

	Now	Future
Shift left (parts of) integration testing	AADL and ACVIP	Higher-level models: UAF, SysMLv2, …
Formal MBSE	Coq mechanization of AADL, static analysis	Extend mechanization to a wider set of languages (SysMLv2), analyses (security, …), and simulation
Tool support	Disconnected, ad hoc transformation	Native Digital Thread

The Policies: adoption barriers within DoD

	Now	Future
	Testing separate from design	ModDevOps extended to whole lifecycle
	MBSE not codified per DoD processes	Enterprise-level MBSE aligned with DoD rules

Backup slides

AADL Standard Suite (AS-5506 series)

Core AADL language standard vV1 2004, .. v2.3 2022]

- Focused on embedded software system modeling, analysis, and generation
- Evidence produced as a result of automated tool-supported analysis
  - Performance analysis: worst-case response time, schedulability
  - Safety analysis: eliciting unsafe scenarios, computing fault trees, probability of reaching an unsafe state
  - Automated model review: conformance to modeling guidelines
  - Code generation: generating “correct-by-construction” software

Standardized AADL Annex Extensions

- Error Model language for safety, reliability, security analysis [2006, 2015]
- ARINC653 extension for partitioned architectures [2011, 2015]
- Behavior Specification Language for modes and interaction behavior [2011, 2017]
- Data Modeling extension for interfacing with data models (UML, ASN.1, …) [2011]
- AADL Runtime System & Code Generation [2006, 2015]
- FACE Annex [2019]

AADL capabilities

AADL is highly tunable, with a restricted set of concepts

- Demonstrated many use cases, 1600+ academic publications

AADL as a backbone, federating multiple activities

- analysis through generation of intermediate models + external tools

Non exhaustive list of analysis capabilities

- Integration: SysML, FACE, Simulink, SCADE
- Architectural pattern checks:
  MILS, ARINC, Ravenscar, Synchronous
- Model checking:
  - Timed/Stochastic/Colored Petri Nets
  - Timed automata et al.: UPPAAL, Versa, TASM
- Scheduling: OSATE, CAMET, MAST, Cheddar, CARTS
- Performance evaluation: real-time and network calculus
- Fault analysis: OSATE, COMPASS,
  Mapping to Stochastic Petri Nets, PRISM
- Security: CAMET (DoDI 8510.01)
- Simulation: ADeS, Marzhin
- Energy consumption of SoC: OpenPeople project
- Code generation: SystemC, C, Ada, RTSJ, Lustre
- WCET analysis: mapping to Bound-T

AADL demonstrated its suitability to support various analysis for the real world

AADL commercial and open source toolchains

Multiple AADL toolchains exist, they can be easily combined thanks to the textual syntax.

- **OSATE** (SEI/CMU) https://osate.org/
  - Eclipse-based tools. Reference implementation
  - Textual and graphical editors + various plug-ins for latency, processor utilization, memory utilization, data consistency, security, safety analysis (MIL STD 882E, ARP4761), ARINC653
- **CAMET** (Adventium Lab) https://www.adventiumlabs.com/curated-access-model-based-engineering-tools-camet-library
  - Extensions to OSATE to support other analysis (Multiple Independent Levels of Security (MILS), Framework for Analysis of Schedulability, Timing and Resources (FASTAR)
- **SCADE Architect** (ANSYS Esterel) https://www.ansys.com/products/embedded-software/ansys-scade-suite
  - Eclipse-based tools. Combine SysML, AADL and other formalisms, code generation
- **AADL Inspector** (Ellidiss) https://www.ellidiss.com/products/aadl-inspector/
  - Lightweight editor, model simulation, scheduling analysis