COLLINS AEROSPACE

USING ACVIP – JMR MSAD CAPSTONE USE CASE

ACVIP USER DAY, 6/2/2022

Stephanie Burns, Avionics
Andrew Muxen, Avionics

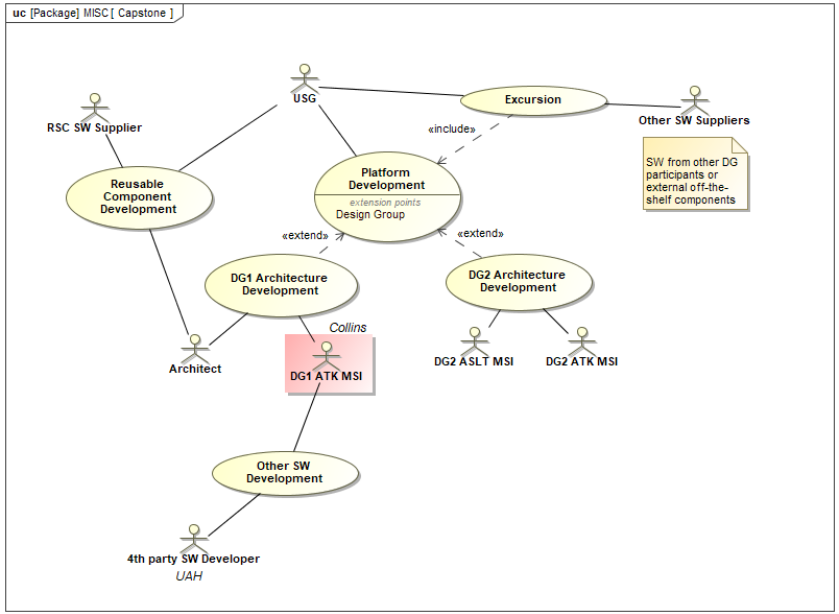
This work was performed under the Army's DEVCOM AvMC
JMR MSAD Capstone Demonstration, #W911W6-19-2-0004

Collins Aerospace

OVERVIEW

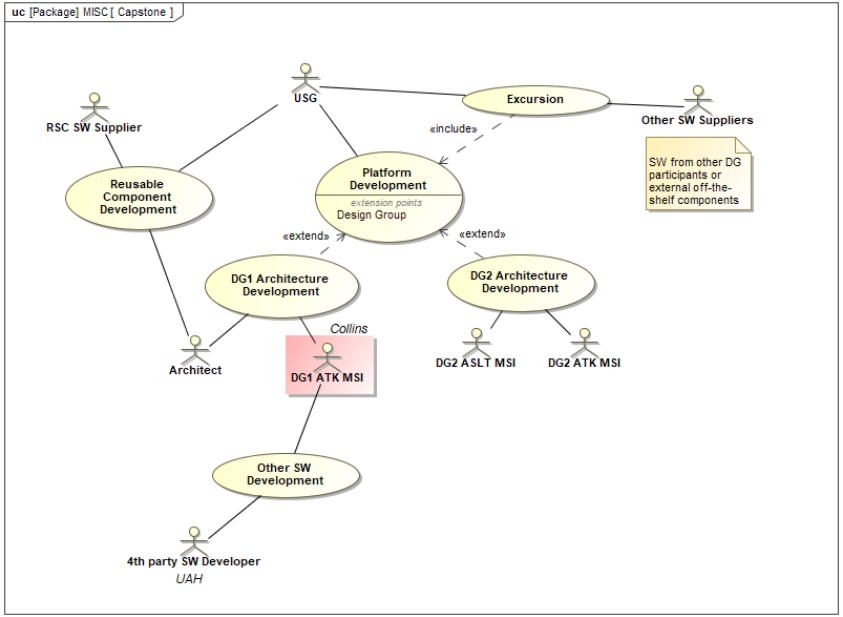
- JMR MSAD Capstone is an Army science and technology program setup specifically to generate lessons learned in the specification and integration of reusable software components

- Three integrators participated using two different organizational approaches
  - External architect role interfacing with integrator
  - Collaborative architecture development between two integrators



OVERVIEW

- RSC SW supplier provided four software components to all MSIs
  - Correlation and Fusion
  - Route Planning and Monitoring
  - Risk Determination
  - Platform Configuration and Performance
- 4th party SW developer provided additional required component and its model
  - Extend our lessons learned
- Excursions modified original integration
  - Evaluate impact of having framework when adding, replacing, or updating software



This briefing summarizes how AADL and ACVIP facilitated this effort

CAPSTONE FRAMEWORK

- SysML models
  - All MSIs received GFI models
    - PEO AVN architecture (RefArch) and performance spec
    - Reusable components SysML model
  - Each design group developed Family of System architecture model (ObjArch)
  - Each MSI developed system architecture model (SysArch)
  - Each MSI develop system design model
- RSC SW supplier and 4th party SW Developer provided
  - AADL models with their own analysis results
  - FACE data models and object code
  - Users guide and source code
    - Limited access to RSC source code internally
- SysML to AADL tool generated overall system AADL model
  - Internal tool used until Adventium tool became available
- Integrated SW Supplier's AADL models into system AADL model for analyses

CAPSTONE ACVIP ACTIVITIES

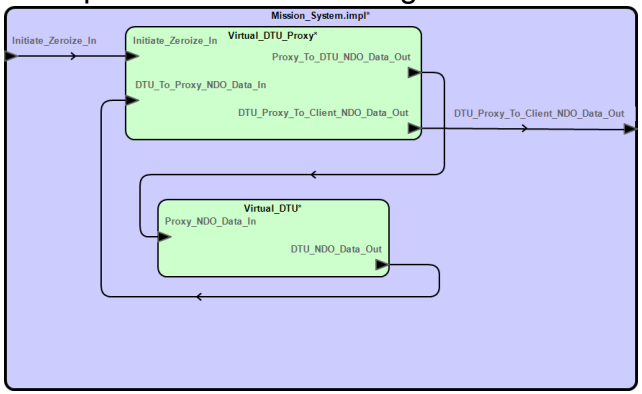
Purpose	Tool Used
Analyze Compatibility between Interface Types	OSATE
Analyze Power Requirements Calculate Weight Totals	OSATE
End-to-End Flow Latency	OSATE, FASTAR™ Schedulability
Analyze Software Resource Budget and Utilization	OSATE, FASTAR Utilization
Security Risk Management	RMF
Fault Impact Analysis	EMV2 Fault Tree Analysis
Behavior	AGREE, CAMET™ SLICED

ASSUME GUARANTEE REASONING ENVIRONMENT (AGREE)

HTTPS://GITHUB.COM/LOONWERKS/AGREE

- AGREE statements resolve behavior ambiguity using formal logic

Example interaction zeroizing the virtual DTU



«interfaceRequirement»
REQ00013

notes

The Virtual DTU Shall output Zeroize_Status as:
- "WAITING", when Remove_DTU_Keys_Trigger is FALSE
- "IN_PROCESS", when Remove_DTU_Keys_Trigger is TRUE and all passcode and file cache have NOT successfully been zeroized
- "ZEROIZED", when Remove_DTU_Keys_Trigger is TRUE and all passcode and file cache have successfully been zeroized

Requirements translated into AGREE statements

```
guarantee "The Virtual DTU Zeroize_Status shall output the same state,
    transition from 'WAITING' to 'IN_PROCESS',
    transition from 'IN_PROCESS' to 'ZEROIZED',
    or transition from 'ZEROIZED' to 'WAITING'" :
    true -> no_transition or transition_1 or transition_2 or transition_3;
```

Issue found: if a falling edge immediately follows a rising edge, the system goes from IN_PROCESS to WAITING based on the trigger rather than completion of zeroize

Ensure elements zeroize only when a trigger occurs

```
guarantee "The Mission System shall output Proxy_Zeroize_Status as 'IN_PROCESS' or
    'ZEROIZED' if and only if Initiate_Zeroize_In is TRUE" :
    (zeroize_state = IN_PROCESS or zeroize_state = ZEROIZED) <=>
    Initiate_Zeroize_In = true;
```

Resolved by eliminating potential trigger behavior in system

```
assume "A falling_edge shall occur in the input stream for Remove_DTU_Keys_Trigger if
    and only if the previous state of Zeroize_Status was ZEROIZED" :
    true -> falling_edge <=> pre(DTU_NDO_Data_Out.Zeroize_Status) = ZEROIZED;
```

WHAT WE LEARNED

WHAT WORKED WELL

- Use of a “harness” to extend system model to each specific analysis type
  - Smaller individual model files
  - Easier to maintain separate analyses

- RSC SW Supplier’s AADL models were detailed which allowed mining data flows (inputs to outputs) present in the implementation

- 4th party SW integration found errors early during specification due to modeling

- Using OSATE’s static analysis of interface types to indicate incompatibility of interfaces defined

WHAT WE LEARNED

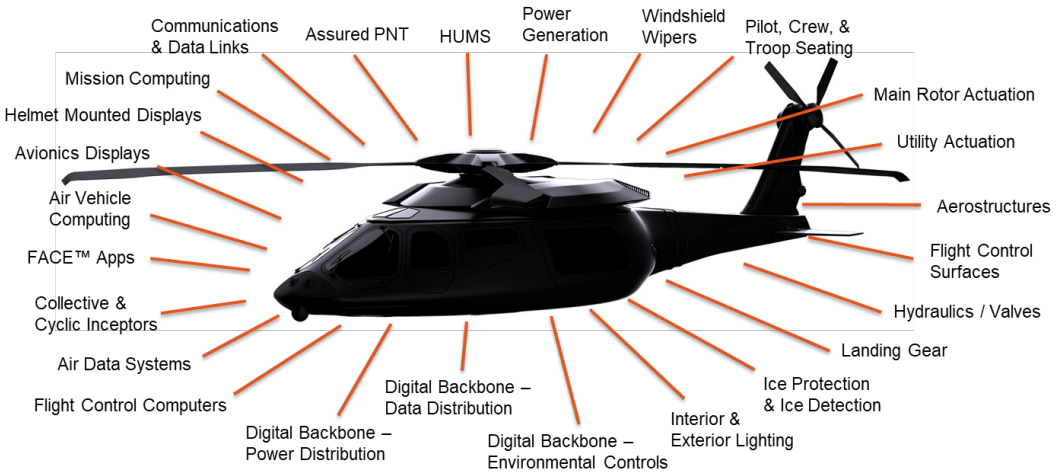
WHAT NEEDED IMPROVEMENT

- Engineering discipline relative to AADL model issues/problems found
  - Treat issues found as problem reports on system design
- Reusable components' SysML and AADL models were out of sync
  - SysML model represented future / complete state
  - AADL model reflected increment of functionality being delivered
- AADL models should precede software deliverables for analysis to impact implementation
  - Maintain strict versioning between model and software deliverable (I.e., the model is what the software build implements)
- To support integration, AADL models about software should focus on its execution unless targeting auto-generation from the model
  - Threads, thread rates, what's invoked by a thread
  - Supplemental to SysML sequence diagrams
- Each tool had unique syntax even if describing the same property (e.g. CPU utilization)

WHERE WE ARE HEADED

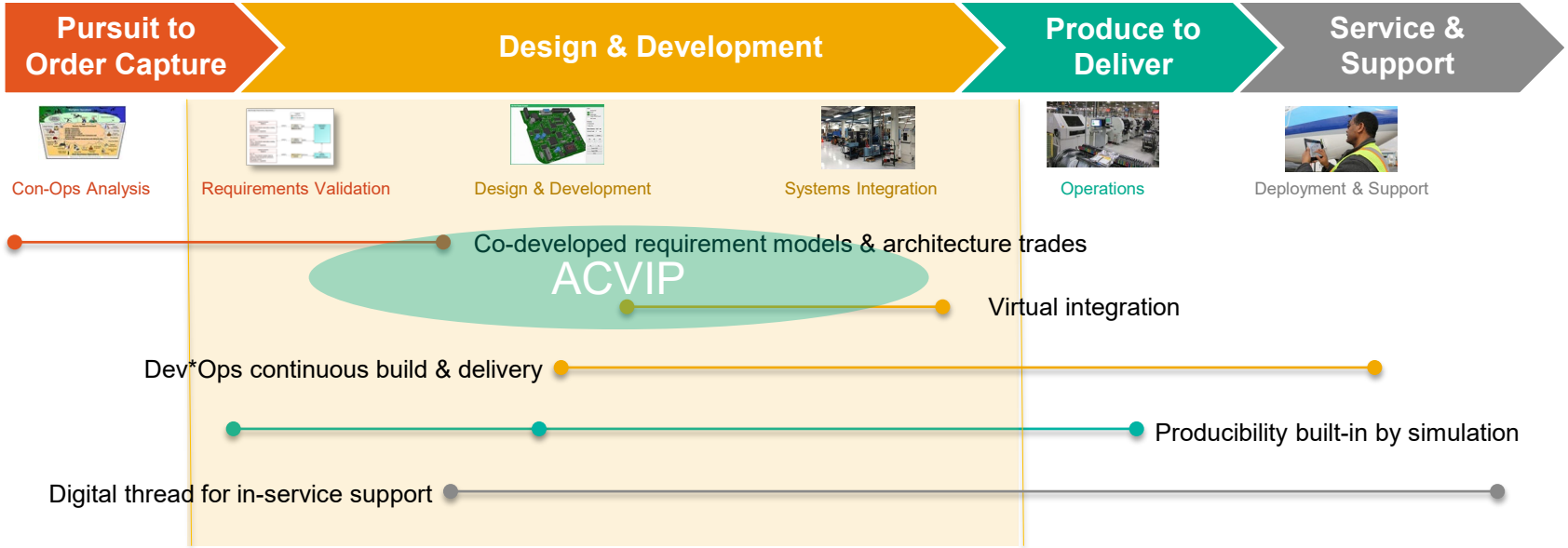
A DIVERSE DIGITAL ENGINEERING ECOSYSTEM

- ACVIP is one piece of the overall digital engineering strategy
- Use AADL models to help inform integration
- Use AGREE to build arguments of correctness as a supplement to test
- Expect “virtual integration” to align with diverse businesses, products, and domains with unique tools and varied digital engineering capabilities



DIGITAL PRODUCT LIFE-CYCLE (DPLC)

COLLINS DIGITAL TRANSFORMATION INITIATIVE, WITH THE OBJECTIVE OF INTEGRATING AND DEPLOYING MODEL-BASED AND DIGITAL CAPABILITIES IN HOW PRODUCT IS DEVELOPED, PRODUCED, AND SUPPORTED



Priority: Connect full product life cycle phases through models to deliver speed & efficiencies

QUESTIONS?

BACKUP

ACRONYM LIST

- AADL – Architecture Centric Design Language
- ACVIP – Architecture Centric Virtual Integration Process
- AGREE - Assume Guarantee Reasoning Environment
- ASLT - Assault
- ATK - Attack
- AVN - Aviation
- CAMET – Curated Access to Model-based Engineering Tools
- CPU – Computer Processing Unit
- DG – Design Group
- DPLC – Digital Product Life Cycle
- EMV2 – Error Model Annex Version 2
- FACE – Future Airborne Capability Environment
- FASTAR – Framework for Analysis of Schedulability, Timing and Resources
- GFI – Government Furnished Information
- HUMS – Health Usage and Monitoring System
- JMR – Joint Multi-Role
- MSAD – Mission System Architecture Definition
- MSI – Mission System Integrator
- ObjArch – Objective Architecture
- OSATE – Open Source AADL Tool Environment
- PEO – Program Element Office
- PNT – Position, Navigation, and Timing
- RefArch – Reference Architecture
- RMF – Risk Management Framework
- RSC – Reusable Software Component
- SLICED – State Linked Interface Compliance Engine for Data
- SW - Software
- SysArch – System Architecture
- SysML – Systems Modeling Language
- UAH – University of Alabama, Huntsville

ABSTRACT

- Collins use of Architecture Analysis Design Language (AADL)-based architecture analyses during Joint Multi-Role Mission System Architecture Definition (JMR MSAD) Capstone provided us an opportunity to analyze the implementation before we integrated reusable software components. Our brief provides an overview of our JMR MSAD Capstone effort, the Architecture Centric Virtual Integration Process (ACVIP) tools used, and how AADL analysis facilitated integration. We have since incorporated those lessons learned into our future digital engineering initiatives

ABOUT THE PRESENTERS

Stephanie Burns in a Technical Fellow with Collins Aerospace. She has been involved in several Army Science and Technology programs associated with Joint Multi-Role Mission System Architecture Definition (JMR MSAD) defining requirements for open systems approaches and served as Principal Investigator on DARPA Cyber Assured Systems Engineering (CASE) TA6 applying AADL and AGREE to demonstrate a cyber hardened system

Andrew Muxen is a principal systems engineer with Collins Aerospace. He has been involved in the Capstone, A-TEAM, and FVL programs. He also leads adoption of MBSE at Collins through the Collins MBSE WG and is OCSMP Model Builder – Intermediate certified.