Joint Common Architecture Demonstration
Shadow Architecture Centric Virtual Integration Process

Final Technical Report

October 9, 2015

Prepared for
Army AMRDEC

Prepared by
Adventium Labs
111 Third Avenue South, Suite 100
Minneapolis, MN 55401

Under Contract
BPA W31P4Q-05-A-0031, Task Order 37

Contents

1 Executive Summary **1**
1.1 Problem Addressed . . . 1
1.2 ACVIP Approach and Goals . . . 2
1.3 Summary of Results . . . 4
1.4 Next Steps . . . 5

2 JCA Demo AADL Timing Model **6**
2.1 Input Data . . . 6
2.2 Model Development . . . 6
2.3 Model Structure . . . 8
2.4 Size and Effort . . . 9
2.5 What If? Analysis . . . 11

3 AADL Timing Analysis Tools **12**
3.1 Requirements for JCA Demo Analysis . . . 12
3.2 Survey of Timing Analysis Tools . . . 13
3.3 AADL Framework for Timing Analysis . . . 13

4 Timing Issue Reports and Evaluations **16**

5 Observations and Recommendations **18**
5.1 AADL Tool Suitability . . . 18
5.2 ACVIP AADL Modeling Handbook . . . 20
5.3 Lessons Learned for Future ACVIP Dem/Eval Projects . . . 21

References **24**

1 Executive Summary

1.1 Problem Addressed

Development cost and schedule overruns have been all too common on large system projects, e.g. F-22 fighter (10 year delay, \$13B overrun[23]), A400M airbus (4 year delay, 6B euro overrun[1]), F-35 fighter (50% cost overrun[28]), A380 airbus, and Boeing 787. Unexpectedly high operation & sustainment costs have limited many systems, e.g. the Space Shuttle goal was 25 flights per year but the achieved was about 5 per year[29]. Project cancellations or field failures have occurred because systems could not meet end-user requirements, e.g. Comanche, London Ambulance Service. Between 2007 and 2011 there were 74 Nunn-McCurdy breaches involving 47 major acquisition programs[11]. \$46B worth of DoD development programs were canceled in the first decade of this century[20].

Studies repeatedly identify inadequacies during the early phases of development as root causes for these problems. Engineering and design issues are a major cause of Nunn-McCurdy breaches[11]. Cost and schedule overrun is often due to unplanned rework during system integration testing, yet most of the originating design mistakes are made during the requirements and specification phases. For example, about 70% of defects in software development projects are introduced during requirements engineering and system architecture development but 80% of these are not found until integration testing or later[9]. Since most life cycle cost has already been determined by the end of requirements engineering (illustrated in Figure 1), early defect detection also has the potential to reduce life cycle costs (about 70% of life cycle cost is operations & sustainment, and operations & maintenance consumes about 30% of the overall DoD budget[13]). Ongoing changes to requirements and design throughout development, illustrated in Figure 2, are indicative of both latent defects and non-resilient requirements and designs.

In anticipation of Future Vertical Lift (FVL) acquisitions, the Army Joint Multi-Role (JMR) Science & Technology (S&T) program is looking at methods and technologies to mitigate these problems for future mission systems. The work described in this report was conducted as part of the first of a planned series of JMR Mission System Architecture Demonstrations (JMR MSAD). The methods and results are also of use and interest to other future DoD acquisitions and to civil science & technology initiatives such as the System Architecture Virtual Integration (SAVI) program[21].

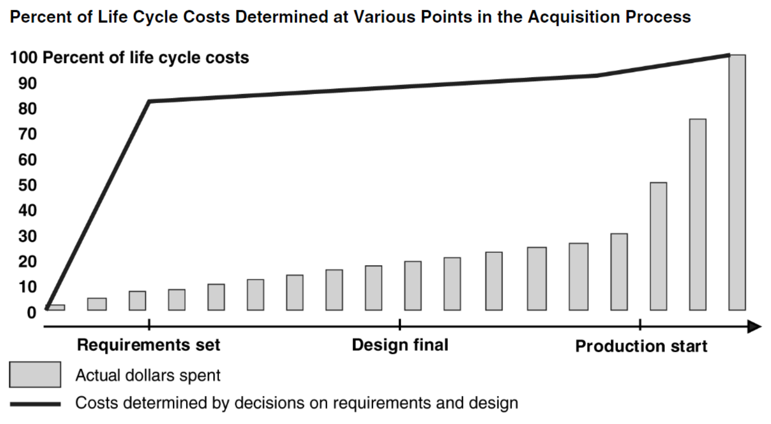

Figure 1: Most life cycle cost is fixed during requirements engineering[10].

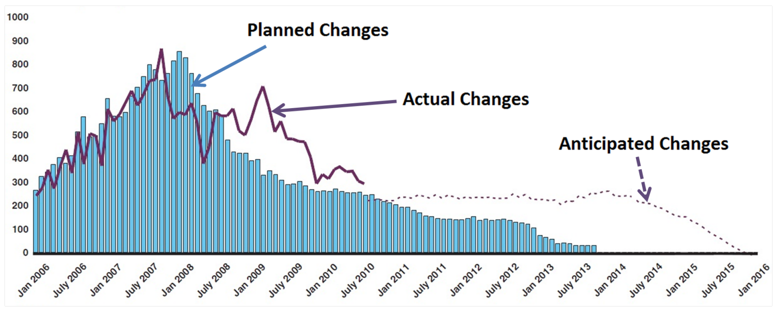

Figure 2: Design changes occur at a significant rate throughout the F-35 program.[12].

1.2 ACVIP Approach and Goals

To address the above problems for mission systems, the US Army is developing a model-based Architecture-Centric Virtual Integration Process (ACVIP). Starting during the requirements engineering phase, the mission system architect develops models written in the SAE standard Architecture Analysis and Design Language (AADL)[18]. The AADL models evolve and are subjected to a variety of analyses at key project milestones to detect defects early in the process.

AADL models will be provided as part of the requirements and technical specifications issued to suppliers. Suppliers will deliver detailed AADL models of their equipment at milestones such as preliminary and final design reviews. The government will perform or oversee the virtual integration and analysis of AADL models from the suppliers at these milestones to detect potential defects in the early rather than the late phases of a project. This will allow corrective and preventative action to occur in early development when the cost of such action is still low.

One goal of this work is to identify methods and technologies that maximize early defect detection effectiveness without requiring excessive amounts of model development and analysis effort. This involves determining what information at what level of detail should be captured in a mission system architecture model at what development milestones, and what analyses should be applied to those models at those milestones. Concrete outputs of this work are technical and acquisition management guidelines documents, which are currently in development[16].

Another goal is to assess feasibility and maturity of ACVIP technologies (methods and supporting model-based engineering tools) that could be available in time for the first FVL acquisition. Concrete outputs of this work are a roadmap and support activities to mature and transition high-value ACVIP methods and tools into the supplier base. The most recent ACVIP Roadmap was presented to JMR management in January 2015[14].

The first in the planned series of JMR Mission System Architecture Demonstrations was the Joint Common Architecture (JCA) Demonstration, initiated in 2014. This demonstration was designed (among other things) to assess a variety of technologies for improved interoperability, portability, commonality and reuse. A Broad Agency Announcement (BAA) was issued to procure a Data Correlation and Fusion Manager (DCFM) software application from two suppliers. The government team and support contractors then integrated those DCFM software components into multiple configurations of a prototype Modular Integrated Survivability (MIS) system hosted in the Aviation Systems Integration Facility (ASIF) at the AMRDEC Software Engineering Directorate. The integrated system accepts inputs from a number of simulated sensor systems (such as Weapons Watch and Tactical Network), uses the DCFM to correlate and fuse track data from multiple sensors, and then displays that data on a moving map. Figure 3 illustrates the high-level architecture of the integrated system.

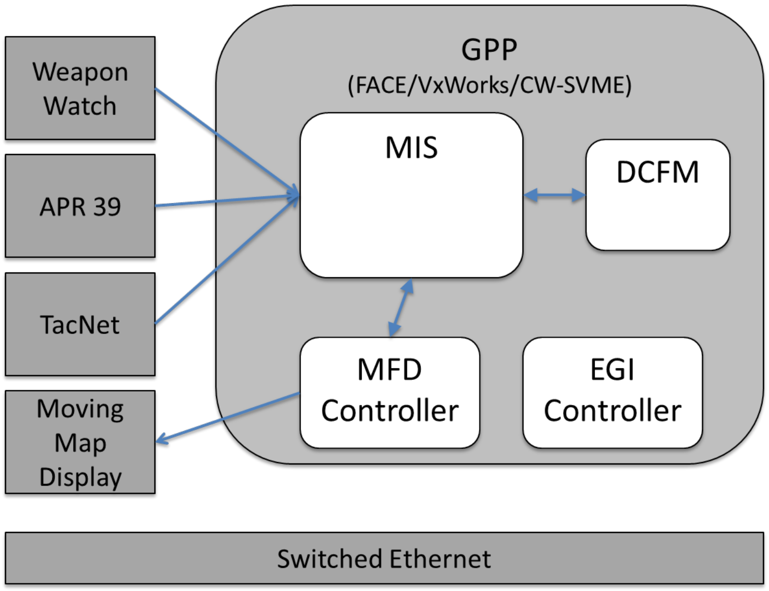

Figure 3: JCA Demo Integrated System Architecture

In parallel with this JCA Demo and under the direction of the government, the Software Engineering Institute (SEI) and Adventium Labs performed a shadow ACVIP. Mission system architecture models were written in the SAE standard Architecture Analysis and Design Language (AADL) by the SEI and Adventium Labs using technical data provided by the baseline JCA Demo participants. An AADL model for the DCFM component was then virtually integrated with MIS and Operating Environment (host computing platform) models to form an AADL model of the overall system. These models were analyzed to detect defects in requirements, safety, and timing properties.

This ACVIP was a shadow performed by the SEI and Adventium Labs in collaboration with the government. The ACVIP activities were not performed at the same time as they would have been in an actual ACVIP, they were delayed relative to the JCA Demo baseline activities. During the ACVIP Shadow the government maintained some isolation between the ACVIP Shadow team and the JCA Demo baseline teams and mediated the exchange of technical information. Issues detected by the ACVIP Shadow performers were reported to the government team, who also received reports of issues found by the various baseline JCA Demo performers during actual component development and system integration. At the time of this writing, the baseline JCA Demo and ACVIP evaluations are still ongoing. The final government evaluations will allow an assessment of the ACVIP methods and tools applied and produce lessons learned to improve these methods, tools, and future JMR MSAD demonstrations.

The SEI team focused on the development of AADL models to capture and analyze requirements and safety properties. The Adventium Labs team focused on the development of conceptual and design AADL models and the capture and analysis of timing and resource demands. Two reasons for making this selection for the Adventium Labs team were as follows.

- This performed ACVIP activites at multiple phases of development. Between the SEI and Adventium Labs, the shadow investigated ACVIP methods that would be applied during system and subsystem requirements, functional architecture development, conceptual architecture development, and architecture and component high-level design.

- This focuses on a category of defects that are expected to fit the profile of high-cost defects: introduced early in development but not detected until system integration time. Suppliers develop individual software applications independently of each other using lightly-loaded lab systems, but timing defects often only show up when multiple applications are integrated and interface and contend with each other on heavily-loaded target compute platforms.

The rest of this technical report is about the Adventium Labs timing modeling and analysis ACVIP activities.

1.3 Summary of Results

The baseline JCA Demo activity was extended from an original February 2015 completion to a May 2015 completion, which is beyond the period of performance of the Adventium Labs ACVIP Shadow task. At the time of this writing, the government ACVIP evaluation activities are still in progress[4]. This technical report documents Adventium Lab's results at the end of our JCA ACVIP Shadow period of performance.

Using technical data provided by the MIS system integration team and the DCFM component suppliers, Adventium Labs developed conceptual architecture and design architecture models following the draft ACVIP AADL Modeling Guidelines document[16]. The total model consisted of about 2400 text declarations in 48 files, of which about 400 lines in 2 files were automatically generated from a Future Airborne Capability Environment (FACE) data model provided by the government as part of the DCFM solicitation. About 170 hours were spent writing models and performing analysis (not including program management, roadmapping, reviews, etc.)

Adventium Labs reported 16 timing-related issues (potential defects) detected during model development and analysis. Most of these were detected during model development rather than as a result of running an analysis tool – simply creating a model that has rigorous syntactic and semantic rules revealed ambiguities and inconsistencies in the original natural language requirements and design specifications. At government request after 15 of these had been reported, we selected 5 to undergo a preliminary assessment by the system integration team. The system integration team judged 3 of these to be correct findings, 1 To Be Determined(TBD), and 1 an ambiguous issue report where they were not sure. Of the 3 judged to be correct findings, 2 were estimated to have a low cost-to-repair and one a medium-to-high cost to repair. We note that the scope of the MIS demo changed during the project, and this affected whether some of the reported issues were still relevant or not.

A question to be answered by JCA MSAD ACVIP studies is, What modeling and analysis guidelines result in a cost-effective ACVIP? What level of detail should be captured in what models at what phases of development? What analyses should be performed at what milestones? Although the AVSI SAVI project did earlier return-on-investment studies that estimated high returns[26], those studies were based on literature surveys and rough estimates of detection effectiveness. The JCA MSAD ACVIP studies are the only ones of which we are aware that are doing actual experiments to determine answers to these questions at a level of detail that will provide useful guidelines for ACVIP practitioners. The preceding paragraph shows the kind of cost/benefit data that is being obtained during this first study. Analysis, evaluation, and lessons learned are still pending from the government team.

The first version of the AADL standard was issued in 2004 and the most recent in 2009. The AADL wiki tools page lists about 20 tools available from a variety of sources[2]. We are aware of six that support some form of timing or resource loading analysis. However, the maturity of the tools varies significantly. Most are provided as-is by universities rather than offered as supported products by commercial tool vendors. Some are still proof-of-feasibility research tools. Existing tools often support only a limited subset of the AADL language. For example, many timing tools support single-processor schedulability analysis but cannot analyze distributed systems. When we started our work, we could find no AADL tool that would analyze Ethernet networks or ARINC 653 compute modules (a widely-used standard for safety-critical systems). The level of testing, usability, documentation and training, and available tool support varies widely. Prioritization and maturation of AADL tooling is needed to support full-scale development of an entire FVL mission system. This is reflected in the most recent ACVIP roadmap[14].

1.4 Next Steps

The current JMR MSAD plan calls for an Architecture Implementation Process Demonstrations (AIPD) followed by a capstone mission system architecture demonstration ending in FY19. These are expected to include ACVIP activities. The next step for our JMR MSAD ACVIP support is to support government evaluation of JCA Demo ACVIP Shadow results. We will also support the use of these results and lessons to improve the guidelines and roadmaps and support planning and preparation for the FY16 JMR MSAD AIPD.

The final section of this report provides observations and recommendations for assessing AADL tool suitability for future projects, lessons learned for the ACVIP AADL Modeling Handbook, and input for planning future JMR MSAD demonstrations.

2 JCA Demo AADL Timing Model

This section describes the development process and the structure and general contents of the AADL models developed for our JCA Demo ACVIP Shadow task.

2.1 Input Data

We developed an AADL model for one configuration of the JCA Demo system using a series of Word® documents and UML and FACE models that we received over a period of time from the government, the JCA Demo system integration team, and a supplier of a Data Correlation & Fusion Manager (DCFM) software component that was integrated into the system. Several deliveries of technical data were revisions of previous deliveries. In some cases we received data as answers to questions that we asked (where these exchanges were mediated by the government as described in a subsequent subsection).

Table 1 summarizes the key data used to build the AADL model. This table omits Modular Integrated Survivability (MIS) briefings seen prior to the JCA Demo ACVIP Shadow project; presentations given as part of a technical exchange meeting in August 2014; questions and answers mediated by the government team (most recently 8 Apr 2015); and test documentation received from the DCFM suppliers.

2.2 Model Development

Model development was performed incrementally and was affected by a number of factors.

As discussed in the previous subsection, technical data was provided incrementally and was subject to ongoing change. Initial models were created as data became available and then were upgraded at various points as additional or changed technical data became available.

The ACVIP Shadow was not performed in-process, meaning that the ACVIP activities were not performed at the same time as they would have been in a real project. The shadow team sometimes received delayed data, and the baseline JCA Demo teams proceeded without waiting for shadow ACVIP model development and defect detection and resolution. This avoided interference with the baseline JCA Demo, but it complicates ACVIP evaluation by the government team. For example, the government evaluators want to determine if an early-phase ACVIP action would have detected and removed an error that would otherwise have escaped to system integration, but unrestricted questions from the ACVIP team to the JCA Demo team in later phases might have led to the correction of that defect in some intermediate phase without government evaluator knowledge. One step taken to simplify post-shadow analysis and evaluation was to limit interactions between the ACVIP shadow team and the other JCA Demo teams. All such interactions were mediated by the government team. A side effect of this is that the AADL models usually lagged behind and were not completely consistent with the actual JCA Demo configuration, which changed during the course of the JCA Demo project.

Model development overlapped development of the timing analysis tooling. Initial checks on the model were limited to standard AADL syntax and semantic checks defined in the standard. As tool capabilities became available, the model could be checked to see that at least parts of it met the requirements for schedulability analysis. Model changes due to applying the timing analysis tools were thus further delayed in some cases.

Several factors affected the structure of the AADL model and the order in which AADL projects were developed.

Date Received	**Description of Technical Data**
29 Jan 2014	**Joint Multi-Role Technology Demonstrator (JMR TD) Joint Common Architecture Demonstration (JCA Demo) Broad Agency Announcement (BAA)** issued 27 Jan 2014, the solicitation for proposals from potential suppliers for DCFM software components.
20 Feb 2014	**Supplemental Information for the Joint Common Architecture Demonstration (JCA Demo) Broad Agency Announcement (BAA)** dated 19 Dec 2013, supplemental data provided on request from potential responders to DCFM solicitation that includes a Word® document of requirements and a FACE data model of the DCFM interface.
5 Jul 2014	**MIS Stakeholder Requirements** dated 15 Apr 2014, a Word® document that contains high-level requirements from the end-user (e.g. pilot) perspective.
5 Jul 2014	**MIS Build 2 Demo Plan** dated 1 Jul 2014, a Word® document that presents the overall architecture and set of configurations for the JCA Demo system.
5 Jul 2014	**System Specification Modular Integrated Survivability** dated 3 Jul 2014, a Word® document that contains technical requirements for the MIS software (a subsystem within the overall JCA Demo system).
16 Jul 2014	**System_Model**, a Rhapsody UML model that contains high-level diagrams of the MIS major functions and interfaces.
18 Sep 2014	**Configuration_Model**, a Rhapsody UML model that contains diagrams of the JCA Demo software components and interfaces.
25 Sep 2014	**Data Correlation and Fusion Manager Software Requirements Specification** dated 12 Sep 2014, PDF requirements document for the DCFM software component supplied by Honeywell.
25 Sep 2014	**Data Correlation and Fusion Manager Software Architecture Description** dated 12 Sep 2014, PDF software architecture document for the DCFM software component supplied by Honeywell.
25 Sep 2014	**Data Correlation and Fusion Manager Software Design Description** dated 12 Sep 2014, PDF design document for the DCFM software component supplied by Honeywell.
25 Sep 2014	**Data Correlation and Fusion Manager Interface Design Description** dated 12 Sep 2014, PDF IDD for the DCFM software component supplied by Honeywell.
29 Sep 2014	**Supplemental Information for the Joint Common Architecture Demonstration (JCA Demo) Broad Agency Announcement (BAA) R1** dated 25 Aug 2014, revision of previous version listed above.
03 Oct 2014	Received revisions of Honeywell DCFM documents listed above.
23 Oct 2014	**MIS Build 2 Demo Plan** dated 18 Sep 2014, revision of previous version listed above.
11 Nov 2014	Received revisions of Honeywell DCFM documents listed above.
11 Nov 2014	Received Boeing-Sikorsky DCFM documents.

Table 1: Technical Data Consulted to Create the AADL JCA Demo Timing Model

- We followed the draft ACVIP AADL Modeling Guidelines[16]. We included comments in the models and some example traceability links to received technical data. These guidelines also identify four levels of model abstraction or detail to support evolutionary refinement of modeling detail across project development phases.

  Functional Architecture models identify functions to be performed but say nothing about the mapping of functions to specific pieces of software or hardware. These are used primarily during requirements engineering phases.

  Conceptual Architecture models identify acquirable software and hardware components. These are primarily used during architecture trade-off studies and acquisition planning.

  Design Architecture models identify increasingly detailed design decisions. These may be subjected to increasingly detailed analysis.

  Integration Architecture models include data that is used to carry out software and system integration activities. These may be used for activities such as automatic generation of configuration files, model-based verification, and model-based evidence for airworthiness and security certification.

- Technical data was received incrementally in a generally top-down fashion (e.g. stakeholder requirements before component designs). Throughout the ACVIP shadow, changes had to be made to some AADL models as we received revised versions of technical data or answers to questions we asked. The order in which model development and analysis phases are presented in the modeling guidelines is generally consistent with the incremental technical data deliveries that occurred on this shadow.

- The Open Source AADL Tool Environment (OSATE) is an Eclipse-based IDE that follows Eclipse conventions, including structuring large models developed by multiple groups as a collection of Eclipse AADL projects. We assumed an Eclipse AADL project would be a basic unit of change and configuration management and model exchange as discussed in the modeling guidelines. We assumed AADL projects should have acyclic inter-project dependencies, and that a supplier should not have to develop an AADL project before other projects on which it depends are available.

- The JCA Demo system is composed of multiple subsystems. We assumed these different subsystems have potentially different suppliers for both the actual implementations and the AADL models. Only cursory technical data about the interface was available for some subsystems. Consequently, a mixed-fidelity AADL model was developed in which some peripheral subsystems were modeled only at the conceptual architecture level.

2.3 Model Structure

Figure 4 illustrates the JCA Demo system configuration as shown in the initial demo plan document we received. We decomposed the overall JCA Demo system architecture into the following major subsystems.

Modular Integrated Survivability (MIS) is a software subsystem containing multiple software components (an MIS software component is roughly comparable to a FACE software Unit of Portability, UoP). MIS is itself intended to be a procurable software item that can be ported to multiple air vehicles and configured to integrate different combinations of survivability equipment for display in different cockpit configurations.

Operating Environment (OE) provides a FACE-compliant operating environment. It is a combination of computer processor hardware, Ethernet and MIL-STD-1553 network interfaces, an ARINC 653 compliant Real-Time Operating System (RTOS), and FACE Transport Services and other FACE software segments.

Data Correlation and Fusion Manager (DCFM) is an MIS correlation client that interfaces with the MIS to access sensor data, perform track correlations and fusions, and store correlated and fused track data back into MIS.

Equipment is the set of available equipment that provides survivability and other sensor and aircraft state data to MIS and reads survivability data from MIS for cockpit display or other purposes.

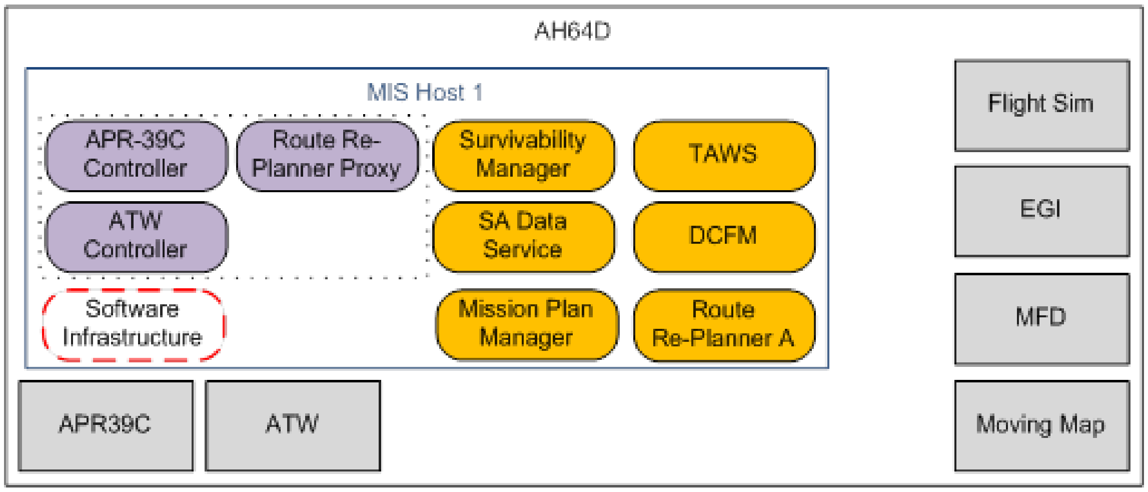

Figure 4: JCA Demo AH64D Configuration from **MIS Build 2 Demo Plan** dated 1 Jul 2014

The combination of architecture model level (functional, conceptual, design, integration) and subsystem (MIS, OE, DCFM, Equipment) resulted in 16 potential AADL projects. Only 8 of these possible AADL projects were created in this shadow. In a few cases projects contained AADL packages that might come from multiple suppliers in practice, which might have resulted in decomposing one of our AADL projects into multiple AADL projects. Table 2 lists the AADL projects created and summarizes their contents.

The amount of information and level of detail in the final models was driven by a combination of the technical data we obtained and the data required by the schedulability analysis tools we used. In some cases the latter was more than the former (e.g. worst-case execution times, operating system overheads), and we had to make our best guesses at reasonable numbers.

2.4 Size and Effort

The total model consisted of about 2400 semicolons in 48 files, where semicolon count is used as an estimate of the number of AADL text declarations. About 400 declarations in 2 files were automatically generated from the FACE DCFM data model provided by the government in the BAA supplement package. Adventium Labs created a separate tally number to record hours spent creating and updating the AADL model, which was about 170 hours. This underestimates the actual time that would be spent, as it excludes program management and technical exchanges and reviews that would be necessary in practice.

AADL Project	Summary Description
MIS_System_Function	The first project created, it captures requirements from the initial **MIS Stakeholder Requirements** and **System Specification for Modular Integrated Survivability** document. It was the only functional model created because the focus of the ACVIP timing was design modeling to support schedulability analysis.
JCA_Demo_Concept	Captures the high-level structure common to all JCA Demo configurations plus the AH64D and OH58 configurations. This was based on the initial **MIS Build 2 Demo Plan** document.
JCA_Equipment_Concept	Models the interfaces to equipment such as WeaponWatch, TacNet, and MovingMap display. The Ethernet conceptual model is included in this project because it is referenced by both Equipment and OE models. No design models were developed for equipment, they were treated as black box models during analysis. Analysis simply assumed this equipment complied with its declared interface timing properties.
MIS_System_Concept	Models the internal decomposition of the MIS software subsystem into software components based on the initial UML **System_Model** and **Configuration_Model**.
DCFM_Concept	Models the interface to the DCFM component based on data provided in the **Supplemental Information for the Joint Common Architecture Demonstration BAA**. The FACE data model was first input to a prototype tool obtained from Vanderbilt Institute for Software Integrated Systems that would automatically translate it into AADL[7]. The generated AADL was subsequently manually edited to work-around some translator bugs and to make it easier to virtually integrate with the other AADL models.
JCA_Demo_Design	Models the AH64D configuration of the JCA Demo in sufficient detail to perform a schedulability analysis. Details such as threads and their periods and estimated worst-case execution times were added. The Ethernet design models were included in this project because the selected Ethernet equipment was assumed to be part of a common Integrated Modular Avionics (IMA) computing platform.
MIS_System_Design	Models the MIS software subsystem design in sufficient detail to perform a schedulability analysis.
DCFM_Design	Models the DCFM software component design in sufficient detail to perform a schedulability analysis based on the DCFM documentation provided by the Honeywell DCFM supplier.

Table 2: AADL Projects in the JCA Demo Shadow ACVIP Timing Model

2.5 What If? Analysis

The model is structured so that the partition schedule, partition and thread context swap times, and thread periods and worst-case execution times are declared in extensions of the overall AH64 configuration. This makes it easy to play "what-if" by going to a single declaration and modifying parameters of interest.

The root system implementations to instantiate and analyze are in project JCA_Demo_Design, package JCA_Demo_AH64D_Design. The implementations to instantiate are OE1_Sep14Schedule (for the partition set and schedule provided in September of 2014) and OE1_Apr15Schedule (for the partition set and schedule provided in April of 2015).

3 AADL Timing Analysis Tools

The AADL tools used on this project were the Open Source AADL Tool Environment (OSATE, an Eclipse-based AADL integrated development environment[19]); a FACE-2-AADL translator (a prototype capability for the ISIS FACE Tools[8]); and the Framework for Analysis of Schedulability, Timing And Resources (FASTAR, a prototype AADL timing analysis framework with integrated MAST and SPICA tools[17]).

This section is an overview of material found in the FASTAR technical report[17] plus some additional observations about analysis of the JCA Demo Shadow model in particular.

3.1 Requirements for JCA Demo Analysis

The JCA Demo ACVIP timing model declares threads that are dispatched and execute on processors, messages that are sent and received by threads, which processors host which threads, which networks carry which messages, etc. The model declares which scheduling algorithms and parameters are used to determine in what order threads and messages are serviced by the hardware resources. The model also declares timing requirements, such as the maximum allowed latency from the arrival of sensor data to the MIS subsystem to the departure of correlated tracks from MIS to the display (which was 1600ms). A timing analysis tool will input the AADL model, analyze the possible timing of thread executions and message transmissions, and determine if these will satisfy all the timing requirements declared in the model.

The JCA Demo Shadow timing model contains a General Purpose Processor (GPP) that provides a FACE ARINC 653 operating environment and a switched Ethernet that connects the GPP to a set of external equipment. The stakeholder requirements called for timely notification of threats to the crew, which if quantified would be a maximum latency from sensor detect through sensor equipment then network then GPP then network then display equipment. Many such end-to-end latency requirements apply to flows of information that pass through multiple pieces of equipment in a distributed system.

The FACE ARINC 653 GPP uses a scheduling algorithm specified in the ARINC 653 standard. This is a layered scheduling algorithm. A static list of fixed-sized time windows is cyclically scheduled on the processor. Each hosted software application is assigned a subset of these time windows. These time windows determine which application is being executed at each instant. An application may have multiple threads that are dispatched at different rates or due to different events. While in a time window, the thread with the highest priority for that application is executing.

Ethernet networks are scheduled using a non-preemptive fixed priority algorithm. Every message channel from one piece of equipment or application to another is assigned a priority. When messages arrive at an Ethernet controller or Ethernet switch for transmission, they are queued according to their priority. Once a message begins transmission over a cable, transmission will continue until that message is complete even if a message having higher priority arrives before then.

The architecture of the system is structured as a series of layers. There are layers of software that provide the FACE ARINC 653 operating environment: hardware processor, real-time operating system, FACE transport services segment, application partition, FACE transport services segment interface, application software. There are layers of protocols for data flowing between software applications over networks. The AADL model must also be structured in a layered manner. This concept is illustrated by Figure 5.

In our project sufficient detail to perform schedulability analysis was not available for every piece of equipment. We would not have had sufficient resources to model all of them in detail even if we

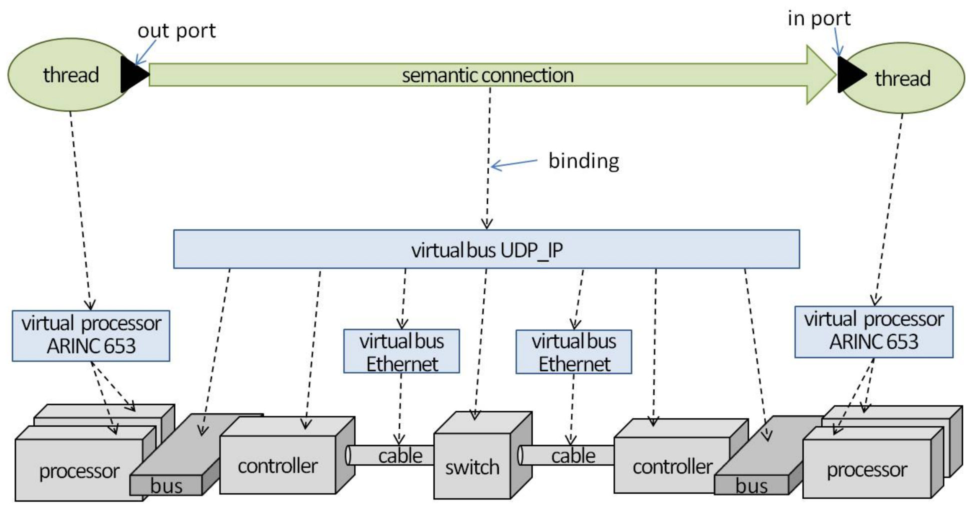

Figure 5: Architectures are Layered

had the information. This will occur in practice, also. Some portions of the system were modeled and analyzed as “black box” components, where analysis simply assumes those components comply with the timing behaviors specified in their interface models. In the JCA Demo, external equipment such as sensors and displays fell into this category.

3.2 Survey of Timing Analysis Tools

The two broad methods used to assess timing characteristics of a model are simulation and schedulability analysis. Simulation is basically execution of a model for a set of test inputs. Schedulability analysis applies mathematical rules to determine upper and lower bounds for all possible timing behaviors permitted by a model. The two have different strengths and weaknesses and can be used in a complementary way. The JCA Demo ACVIP Shadow used schedulability analysis methods.

There are a variety of schedulability analysis methods and tools. Different ones are suited for different kinds of equipment – different thread dispatching and message passing patterns and protocols, different scheduling algorithms, different kinds of timing requirements. Table 3 shows a survey of schedulability analysis tools. Many of these use their own proprietary input formats, only some of them can operate on AADL models.

There is no single tool in Table 3 that will schedule the complete system, for example to determine an end-to-end latency between a message departing a sensor and a message containing the processed results arriving at a display. Even where a tool listed can accept AADL input, its full capabilities may not be available to analyze AADL models. For example, the previously existing bridge from OSATE to the MAST tool only supported uni-processor schedulability analysis. SPICA was a schedule generation tool but was not initially formulated to accept a user-defined schedule as input and analyze its correctness w.r.t. declared timing requirements.

3.3 AADL Framework for Timing Analysis

At the time the JCA Demo ACVIP Shadow project started, Adventium Labs was working on another project to address the problem of analyzing multi-layered heterogeneous distributed systems. This project was creating a framework that would allow multiple existing tools to be used collaboratively to analyze an overall system. The user can direct that different portions of the AADL

Tool	**Supplier**	**Comments**
OSATE CPU Util Budgeting	SEI	Computes processor utilizations from user-specified MIPS loads and capacities
OSATE Latency Summing	SEI	Sums user-specified latencies along flows
OSATE CPU RMA	SEI	Processor preemptive fixed priority analysis
MAST	U Cantabria	Preemptive fixed priority and earliest deadline analysis, Earliest Deadline First (EDF) within Preemptive Fixed Priority (PFP), linear sequences of dependent tasks
Cheddar	U Brest	Preemptive fixed priority analysis
AADL Inspector	Ellidiss	Preemptive fixed priority analysis, ARINC 653 simulation
RTaW-Pegase	Realtime-at-Work	Network calculus tool for switched networks
SPICA	Adventium Labs	Supports ARINC 653 style schedule generation and analysis
ASIIST	UIUC	Network calculus tool for Peripheral Component Interconnect (PCI) networks
RAPID	Tri-Pacific Software	Preemptive fixed priority analysis
ARINC 664 AFDX	U Toulouse	Network calculus tool for ARINC 664 networks
ARINC 664 AFDX	Rockwell Collins	Network calculus tool for Rockwell Collins ARINC 664 product
TTEthernet	TTTech	Tool for TTTech Time-Triggered Ethernet product
SymTA/S	SymtaVision	Primarily targets OSEK compliant automotive systems
RTC	ETH Zurich	Open source real-time calculus toolkit
UPPAAL	U Uppsala	Timed automata model checker

Table 3: Schedulability Analysis Tools

model are to be analyzed by different tools. The framework converts the outputs of one into the inputs of another as needed and iterates until a global solution is reached. This is the Framework for Analysis of Schedulability, Timing And Resources (FASTAR)[17]. Figure 6 illustrates how the FASTAR Timing plugin is installed into the OSATE AADL development environment and, in turn, various specialized analysis tools are installed as plugins to FASTAR.

For the initial configuration of FASTAR, we selected the MAST and the SPICA tools to integrate. This is because MAST supports Ethernet analysis and SPICA supports ARINC 653 analysis. Long-term, FASTAR will permit the developers of analysis tools and/or the developers of mission system architectures to select and integrate desired tools. The Eclipse license terms allows IP rights to be preserved for individual integrated plug-ins, so an analysis tool vendor could retain commercial rights to their tool or a system architect could use an internally developed proprietary analysis tool.

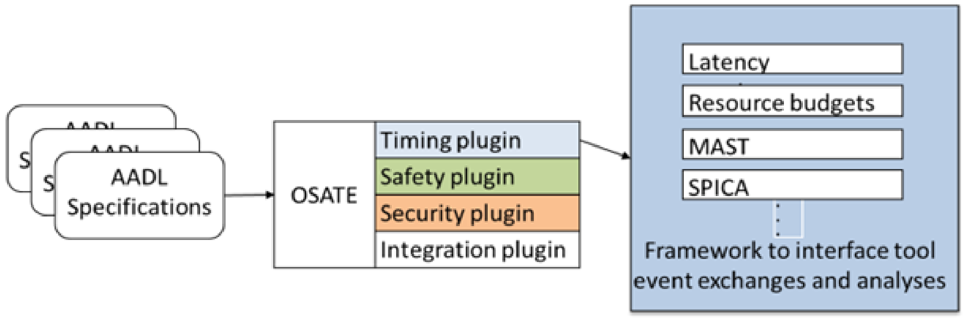

Figure 6: Framework for Integrating Multiple Selected Timing Analysis Tools

Because the FASTAR project overlapped with the JCA Demo ACVIP Shadow, we were able to feed requirements from the shadow project into the FASTAR project. For example, we changed the initial set of third-party tools to be integrated into the framework (we switched from the Real-Time Calculus toolkit to SPICA because the latter could be modified to analyze ARINC 653 schedules). The use of the JCA Demo Shadow model as a requirements study and test case also prioritized and guided development of analysis capabilities for layered architectures and architectures with some black-box components.

4 Timing Issue Reports and Evaluations

We filed a total of 18 issue reports with the government ACVIP team to date. Some of the reports overlapped, and Table 4 compresses these into 15 issues.

In January 2015 the government evaluation team performed a preliminary assessment of shadow ACVIP issues reported up to that time. They asked each ACVIP shadow performer (SEI and Adventium Labs) to identify five reported issues that we recommended be assessed by the MIS integration team. They then asked the MIS integration team to assess these, including whether they thought the issue report correctly identified a defect and their estimated cost to repair that defect (Low, Medium, High). The last two columns of Table 4 indicate the 5 issues evaluated and the responses. The scope of the MIS project changed during the program, which changed whether issues were relevant or not for the final demonstration. The MIS team only evaluated the five issues we identified.

All but the last two issues were submitted while the AADL model was being prepared. They were identified because the need to capture information in a modeling language with a rigorous syntax and semantics forced certain questions to be raised. This is consistent with findings reported in the literature for formal methods that many defects are detected or avoided simply through the discipline of being captured in a rigorous modeling language[5].

All but the last two issues were submitted before the FASTAR schedulability analysis capability was sufficiently mature to attempt an analysis of the model. It is reasonable to expect that further use of the analysis tool will uncover additional issues.

At the time of this writing, the model indicates a defect due to inconsistency between the most recent partition schedule and the application thread periods we have from earlier design documentation. (This is the OE1_Apr15Schedule model instance.)

File Date	Summary	Bug?	Cost
18-Jul-14	The relationship between component states and MIS system state is not fully specified. It is unclear what the MIS system state is for each combination of the possible states of all the components within that system.	Y	L
18-Jul-14	The set of transitions between component states are not completely defined. Assumptions had to be made when constructing the AADL mode transition diagrams.		
18-Jul-14	It is not stated whether each component must manage its own configuration data or if a common platform configuration management service is provided and to be used.		
18-Jul-14	The timing of a request/response exchange is specified as a periodic send of the request. However, this is not the same as the time between arrivals of that request or the time between returns of the response or the request-response latency due to jitter and latency in the transmission and processing of requests. Requirement is not clear.		
18-Jul-14	For components that service requests, timing is specified as the arrival rate of requests. Some of these components may response to multiple clients. Do the arrival rates apply to each individual client or to the set of all clients in a particular system configuration? Request arrivals may be bursty due to jitter, are the rates assumed to be average rates or does this matter? Requirement is not clear.	NR	NR
18-Jul-14	The relationship between the terms ”capability,” ”function,” and ”capability interface” is not well defined. This may lead to some confusion about what requirements apply to what.		
25-Jul-14	The MIS System is a subsystem within the overall JCA Demo system. What are the system requirements for the overall JCA Demo system? Is there traceability between the JCA work products? The flow-down of requirements is not clear.		
25-Jul-14	The MIS System provides two types of interfaces to Aircraft Survivability Technology (AST) equipment, a native interface to specific legacy equipment and a portable MIS_AST interface for upgraded and new equipment. Which interface is used by which equipment in the JCA Demo? This is not clear.		
25-Jul-14	The Stakeholder Requirements state that hazard warnings will be provided to the crew in a timely manner. The MIS Build 2 Plan (assumed by ACVIP modeler to be the JCA Demo System requirements) derives no JCA demo system timing requirements for this.	Y	M/H
8-Aug-14	The MIS System Specification shows a single interface to support multiple clients, MIS_AST equipment, etc. It is unclear if the protocol will use a single incoming port to service multiple clients. Some platforms may not support having multiple clients make requests to a single server port, each individual client may require its own pair of request and reply ports. The requirements are not clear.		
14-Nov-14	Stakeholder requirement that crew be alerted in a timely manner is not fully quantified in derived requirements and specifications.		
14-Nov-14	Specified MIS subsystem latency requirement of 1600ms may not be feasible.	TBD	NR
14-Nov-14	Where request/response protocols are used from clients to MIS, there is a risk of overflowing request message input ports.		
23-Dec-14	The specified module partition schedule does not satisfy ARINC 653 scheduling rules.	Y	L
3-Mar-15	Different patterns were selected for different software applications and message connections: periodic publish-subscribe for some and event-driven request-response for others. This awkward mix of patterns is difficult to schedule and verify and increases the likelihood of scheduling defects.		

Table 4: Summary of Timing Issues Reported by ACVIP Timing Analysis

5 Observations and Recommendations

In this final section we offer some observations and recommendations in three areas.

- How do we assess when an AADL tool is suitable for use on a project?
- What lessons learned can be incorporated into the ACVIP AADL Modeling Handbook?
- What lessons learned can help plan future ACVIP dem/eval projects?

5.1 AADL Tool Suitability

The suitability of an AADL tool must be assessed with respect to a planned purpose. In this report the assumed purpose is to support the JMR MSAD AIPD and Capstone Demo S&T projects. This is a nearer-term intermediate purpose than the long-term goal of insuring the availability of AADL tools fit for use on Future Vertical Lift (FVL) acquisitions. For JMR MSAD we want tools that are suitable for use by engineers doing advanced Research & Development projects but who are users rather than developers of AADL tools. They will be applying these tools to support the development of demonstration mission systems that are of significant size and complexity. In this report we assume "significant size and complexity" means not less than the JMR Demo shadow models.

Here are some criteria that could be used to assess AADL tool suitability. We apply them to the tools used in this project, based on our experience with those tools on this project.

The functionality must be adequate for the intended purposes. For example, although a number of AADL schedulability analysis tools existed at the start of this project, none could analyze Ethernet networks or ARINC 653 compute modules. AADL is a sufficiently rich language that no tool supports every possible combination of features. Many tools support only a small subset (for example tools produced by university research projects). AADL declarations such as feature groups and feature group connections, flows, modes, virtual categories and layered bindings, etc. are often not necessary to demonstrate and evaluate technical feasibility of a particular tool but may be necessary for substantial models subjected to multiple tools.

The JMR Demo model made substantial use of extensions (inheritance), feature groups, feature group connections, flows, and virtual categories and layered bindings. Our survey of existing tools showed that support for layering and flows is limited. Several of the issue reports we filed for OSATE related to feature groups and their use in connections and flows, suggesting this aspect of the language may not have been well-exercised by plug-in developers. Our model made very little use of modes in part because support for modes was not expected to be available in the schedulability analysis tools and framework we used.

Scalability must be adequate for the intended purpose. Both the size of the model and the size of the development team should be considered, as tool features that support collaborative development may be important. Many analysis tools can be demonstrated on small models but will not scale to models of the size anticipated for JMR MSAD projects. The JCA Demo model grew sufficiently large that we did begin to see some slow-down in the OSATE IDE (edit gestures taking up to 3 seconds rather than the near-instantaneous editing experienced with the Eclipse-based Java IDE). Analysis of the Ethernet network and workload by the Modeling and Analysis Suite for real-Time systems (MAST) tool and analysis of the preliminary partition schedule by the Separation Platform for Integrating Complex Avionics (SPICA) tool were surprisingly efficient and required only a few seconds. However, the current SPICA analysis does not yet take into account multi-partition flow latency constraints. We would like to test scalability to larger and more complex models.

The tool must be sufficiently dependable. It must have sufficiently few defects and a sufficiently low error rate. Tool developers have high tolerance for bugs in their own tools, external R&D users on prototype projects less so, production engineers even less so. There are qualitative questions that we can ask to assess tool maturity. How many projects has the tool been used on? How large and complex were those models? Does the tool undergo regular regression testing? How large is the regression test suite? How large is the active user community?

On this project, Adventium Labs was an R&D user of OSATE (an Eclipse-based AADL IDE obtained from the SEI); FACE-2-AADL translator (a translation tool obtained from Vanderbilt University); and MAST (a schedulability analysis tool obtained from University of Cantabria). We filed perhaps a dozen bug reports for OSATE in the last year, but all were reasonably easily dealt with using a combination of work-arounds and timely bug-fixes by the OSATE supplier (SEI). We had some email exchanges with Vanderbilt and filed 7 issues on the FACE-2-AADL translator, but we were able to manually work-around all of the issues. We had perhaps a half-dozen email exchanges with the MAST developers but were able to work-around the issues we encountered. On this project, Adventium Labs was a tool developer user of FASTAR and SPICA. FASTAR was still being developed and SPICA significantly modified during this ACVIP shadow project, and the error rate would probably not have been acceptable to most third-party users.

Availability of support is an important factor that affects tool dependability and tool usability. Support means timely response to bug reports and requests for enhancements; training services; and technical consulting to apply the tool on a specific project. When assessing an AADL tool, consideration should be given to whether that tool has an active and funded development and support project and other users. For some AADL tools, use on JMR MSAD projects may require collaborative use in which support funding flows to the tool developers for both tool enhancement and support and technical advice on its use. For some university-supplied tools, even informal support may not be available after the students working on a tool project have graduated.

At this time, to the best of our knowledge only OSATE (and AADL Inspector, which we didn't use on this project) have established bug reporting and training procedures and offerings. There are current and anticipated projects (AADL Workbench by the SEI, Mature FASTAR by Adventium Labs, FACE 3.0 Tools by Vanderbilt) that are expected to lead to improved training and support for the other tools used on this ACVIP shadow project.

Usability refers to characteristics that affect how easy and effective it is for people to use a tool. Common usability criteria are [27]:

Learnability: How easy is it for users to do basic tasks? How steep is the initial learning curve?

Efficiency: How much more quickly can a user accomplish a task using the tool than using alternative methods and tools?

Memorability: How quickly can a user resume work with a tool when they return to it after a period of not using it?

Error-prone: How often do users make mistakes when using the tool? How easy are mistakes to detect and correct? How serious are the consequences of mistakes?

Satisfaction: How pleasant or frustrating is the tool to use?

Usability criteria are particularly difficult for the tool developers to objectively assess, and we offer none in this report. Use by and feed-back from third-parties (e.g. by JMR MSAD demo performers) is particularly helpful to assess and improve these characteristics.

5.2 ACVIP AADL Modeling Handbook

This is the second modeling project where we used the general division of model phases and guidelines into functional architecture, conceptual architecture, design architecture, and integration architecture. In both cases this division seemed to work fairly well. The guidelines discuss several ways to manage relationships between the phases, including use of AADL extends and refines declarations. There are some limitations on what can currently be done with refines declarations, and we filed errata with the AADL standardization committee with suggested improvements. There are use case scenarios where a conceptual component cannot be declared as an extension of a functional component (for example), which is why the current guidelines permit alternative ways to trace between elements of models at different abstraction levels. For example, it is often the case that a function is implemented by multiple conceptual components and a component supports multiple functions, so that a many-to-many traceability is needed.

The SEI is expected to begin work later this year on an ACVIP Acquisition Management Handbook. Collaboration needs to occur between Adventium Labs and the SEI to assure alignment between the acquisition management and the AADL modeling handbooks. The acquisition management handbook should be scoped so that it fills the gap between the ACVIP AADL Modeling Handbook and higher-level DoD acquisition directives, instructions and handbooks[6, 22].

The handbook should more clearly cite and discuss relationships with other key standards such as FACE, OMG, SAE, etc. A clearer mapping should be defined between the terms used in the handbook and the terms used in other documents in order to minimize confusion. We need to limit the list to a few key standards.

We should add rationale paragraphs in places so readers understand the purpose and intended benefits of specific guidelines.

The JCA Demo ACVIP Shadow illustrated the importance of dealing with change across multiple work products being developed by multiple organizations. Material should be added to the section about configuration management and model exchange that discusses change management.

The guidelines should be generally applicable across a broad range of projects. At the same time, the guidelines need to be specific enough to provide useful operational guidance. Several things can be done to try and balance these competing goals.

The current guidelines include material on the need to tailor and profile for each specific project. The first step in an ACVIP is planning the ACVIP activities in a new development project. A special section should be added to more explicitly highlight and discuss this critical facet of the guidelines.

The current guidelines include recurring “JCA Example” subsubsections. We should incorporate useful examples from all JMR MSAD demos as this document evolves. This will allow informative details to be included that help understand the guidelines even though they are too project-specific to be general guidelines.

We anticipate that the JMR JCA Demo ACVIP and future JMR MSAD projects will produce ACVIP evaluations and lessons learned by the government evaluation team. We recommend that the final reports from these efforts should also target the audience of future ACVIP practitioners. That is, produce documents that can also serve as “practice and experience” records that can be cited in the final ACVIP AADL Modeling Handbook and final ACVIP Acquisition Management Handbook. These would serve as detailed accounts of specific projects that should be studied as part of the technology adoption process by future ACVIP project and technical managers.

We need to be careful about what goes into the AADL guidelines versus what should be found in the documentation for user-selected tools or what should be determined based on specific project needs. For example, the guidelines should not include data specific to a particular vendor’s network product or AADL tool. We delayed filling in these sections in any detail until after we accumulated some experience developing and using FASTAR and its User's Manual.

One of the fundamental purposes of the AADL modeling handbook is to provide useful guidance on the level of detail at which to model and analyze at each phase in order to properly balance added ACVIP effort against reduced rework effort. A basic guideline that is already discussed in the draft handbook is, "sufficient for the analyses that have been selected by project management for a particular project milestone." We believe this can be further refined in both the handbooks and ACVIP evaluation reports. The anticipated results of the ACVIP metrics collection and evaluation by the government will be used to improve the guidelines in this area.

We should consider adding an ACVIP improvement supporting process. This supporting process would provide guidelines to future ACVIP practitioners on metrics collection and assessment so that they can continue to evolve and improve the ACVIP through experience on production programs. These guidelines would include lessons learned from the planning and execution of the government ACVIP evaluation efforts.

5.3 Lessons Learned for Future ACVIP Dem/Eval Projects

In this shadow project the ACVIP activities were carried out separately from the baseline demo. Future JMR MSAD projects should do some ACVIP activities in-process. Some model development and analysis should be carried out by performers. The performers should also participate in metrics collection and other activities needed to support government ACVIP evaluation and improvement. The plans for future JMR MSAD projects should thus include some guidance for conducting in-process ACVIP activities by the performers and for supporting ACVIP data collection and evaluation.

In this shadow project AADL models for requirements and AADL models for design timing analysis were written and analyzed largely independently by the SEI and Adventium Labs. Future JMR MSAD demonstrations should do more integrated model development involving multiple groups. Some sharing and collaborative model development should be done using at least conventional shared repository technologies such as git or svn.

AADL tool suitability should be assessed and taken into consideration both during initial planning of the JMR MSAD demonstration and in the concluding ACVIP evaluations. Tools that are of high interest but relatively low maturity will require either shadow use or additional support for and collaboration with performers, for example.

One aspect of defining and executing an ACVIP evaluation plan is specifying how issues and defects and root causes are to be categorized. Such categorizations would be used both by performers when reporting issues and by evaluators when assessing root causes and evaluating ACVIP results. Defect categorization is a topic that should be considered in the plans and guidelines developed for future JMR MSAD projects. The concluding paragraphs of this subsection provide some thoughts on this subject.

There has been much previous work that proposed specific categorizations of defects[24]. In practice a defect categorization (like any model) should be suited for its purpose. For example, although studies that categorized software defects by the day of the week in which they were introduced yielded the interesting result that defect rates are higher on Fridays and Saturdays[15], this categorization does not inform a decision about which unit test coverage metric and threshold to select. When doing process improvement studies, a defect categorization should be designed to inform the questions being asked and the decisions to be made.

In the case of ACVIP we want defect categorizations that inform decisions about what level of detail to include in a model and what analyses to subject those models to at each phase. For example, when an ACVIP evaluation team looks through collected defect data, it might be useful to categorize each defect according to the phase in which the root cause occurred, the analysis method(s) likely to have detected it, and the kind and level of modeling detail that would have recorded the information necessary to capture that defect.

Recent research on the defect detection effectiveness of various methods shows that effectiveness can vary depending on the severity of the defect. For example, there is some evidence that static code analysis (bug-finder) tools detect many coding defects. However, they tend to be minor defects that are caught in early testing, and the tools have a high false positive rate[3]. ACVIP focuses on high-cost defects that remain latent until software integration or later (including the most expensive of all, defects that escape to fielded systems). ACVIP assessments of effectiveness should take into account estimates of the severity and cost of defects and likelihood of escape to late project phases, not just the number of defects detected.

The traditional notion of a defect is that it exists at a specific point in a specific work product, e.g. a specific line of code, a specific paragraph in a requirements document. It is becoming increasingly apparent that important categories of defects are associated with interactions or inconsistencies between multiple work products rather than one in particular. When studying defect data, it is important to distinguish the change that was made to fix that defect from the root cause of that defect[25]. For example, if two modules make different assumptions about the protocol used to exchange data then either could be changed to fix the defect. However, the root cause does not lie in the module that was selected to be changed but in the ambiguity of the protocol specification or the inconsistency of its implementation.

We filed a 16^{th} issue later after some reflection. This issue is qualitatively different than the first 15. It did not cite a specific paragraph or table in a requirements or design document. Instead, the issue was that different protocols were specified for different interactions in the JCA Demo system. Some applications communicated using a request-response (a.k.a. client-server) protocol, while others used a publish-subscribe protocol. Most definitions of "architecture" include a discussion of so-called patterns or styles that apply to interfaces between components. This mixture of two patterns resulted in a system that is at least difficult (though not yet known to be impossible) to schedule and verify. It significantly complicated the down-stream integration and verification tasks and significantly increased the risk of down-stream defects. It is an architectural issue that spans multiple components rather than being associated with a specific component.

Development process is important, too. A model cannot contain every possible detail. A model is always an abstraction that captures the high-value information that is reasonably available at a particular phase in a project. A defect could result because of inconsistent implementation of an interface model as well as an under-specified interface model. A good model used with a bad development process can produce a failure, just as good software executed on faulty hardware can produce a failure. We will always need to rely on good developmet processes to fill in the details properly. When assessing the likelihood and severity of product defects that could result from something found in a model, the down-stream development process may need to be considered.

The above paragraphs illustrate that there is a difference between a defect in the implemented system that could result in system failure and a defect in an upstream work product. When guidelines are developed for categorizing and reporting defects in requirements and design models the threshold is not a simple, "Will this clearly result in an implementation defect?" but rather, "Does this create significant risk of serious implementation defects?" The guidelines should instruct ACVIP practitioners to look for architectural categories of defects (not just defects that can be associated with a specific requirements paragraph or component). The guidelines should instruct ACVIP practitioners to look for issues that create risk of down-stream defects (not just things that are clearly wrong). ACVIP is, among other things, a project risk mitigation methodology.

References

[1] Airbus Delivers First A400M Transport Plane to France After Years of Delays, COst Overruns. http://rt.com/business/airbus-a400m-france-delays-561/, Sept 2013.

[2] Aadl tools. https://wiki.sei.cmu.edu/aadl/index.php/AADL_tools, 2014.

[3] Nathaniel Ayewah, William Pugh, J.D̃avid Morgenthaler, John Penix, and YuQian Zhou. Evaluating static analysis defect warnings on production software. In *Workshop on Program Analysis for Software Tools and Engineering*, 2007.

[4] Alex Boydston, Peter Feiler, Steve Vestal, and Bruce Lewis. Joint common architecture (jca) demonstration architecture centric virtual integration process (acvip) shadow effort. In *American Helicopter Society Annual Forum*, May 2015. to appear.

[5] Edmund Clarke and Jeannette Wing. Formal methods: State of the art and future directions. In *ACM Computing Surveys*, 1996.

[6] Operation of the defense acquisition system. Technical report, US DoD OSD AT&L, January 2015. http://www.dtic.mil/whs/directives/corres/pdf/500002p.pdf.

[7] Future airborne capability environment (face) academia fast sheet. Technical report, US Army RDECOM, 2015. AMRDEC SED, PoC Scott Dennis.

[8] Isis face project. https://face.isis.vanderbilt.edu/redmine, 2015.

[9] Peter H. Feiler, Jorgen Hansson, Dionisio de Niz, and Lutz Wrage. The Economic Impacts of Inadequate Infrastructure for Software Testing. Technical Report NIST O2-3, National Institute of Standards and Technology, May 2002.

[10] GAO. Best Practices: Setting Requirements Differently Could Reduce Weapon Systems Total Ownership Costs. Technical Report GAO-03-57, U.S. General Accounting Office, 2003.

[11] GAO. Trends in nunn-mccurdy breaches. Technical Report GAO-11-295R, U.S. General Accounting Office, 2003.

[12] GAO. JOINT STRIKE FIGHTER Restructuring Places Program on Firmer Footing, but Progress Still Lags. Technical Report GAO-11-325, General Accounting Office, 2011.

[13] John C. Hawkins. Analysis and Forecasting of Army Operating and Support Cost for Rotary Aircraft. Technical Report AFIT/GCA/ENV/04M-03, Air Force Institute of Technology, March 2004.

[14] Architecture centric virtual integration process workplan and rom proposal for fy15-17. Technical report, US Army RDECOM, January 2015. presented FOUO by JMR MSAD ACVIP Lead to Program Director JMR/FVL, PoC Bruce Lewis.

[15] Huzefa Kagdi, Michael Collard, and Jonathan Maletic. A survey and taxonomy of approaches for mining software repositories in the context of software evolution. In *Journal of Software Maintenance and Evolution: Research and Practice*, 2007.

[16] Adventium Labs. Acvip aadl modeling guidelines, November 2014. Work In Progress.

[17] Adventium Labs. Framework for the Analysis of Schedulability, Timing And Resources (FASTAR) Final Technical Report. Technical report, Prepared for US Army AMRDEC Under Contract BPA W31P4Q-05-A-0031 TO 37, December 2014.

[18] Society of Automotive Engineers. Architecture analysis & design language (aadl). Aerospace Standard AS5506, 2004.

[19] Osate 2. `https://wiki.sei.cmu.edu/aadl/index.php/Osate_2`, 2013.

[20] John Reed. $46B Worth of Cancelled Programs. `http://defensetech.org/2011/07/19/46-billion-worth-of-cancelled-programs/`, 2011.

[21] The system architecture virtual integration program. `http://savi.avsi.aero/`, 2013.

[22] Moshe Schwartz. Defense acquisition: How dod acquires weapon systems and recent efforts to reform the process. Technical report, Congressional Research Service, January 2013.

[23] Ralph Vartabedian and W. J. Hennigan. F-22 Program Produces Few Planes, Soaring Costs. `http://www.latimes.com/business/la-fi-advanced-fighter-woes-20130616-dto,0,7588480.htmlstory#axzz2qCnZQFo2`, June 2013.

[24] Stefan Wagner. Defect classification and defect types revisited. In *Workshop on Defects in Large Software Systems*, July 2008.

[25] Gursimran Singh Walia and Jeffrey Carver. Technical report.

[26] Don Ward, Steve Helton, Greg Polari, and Dave Redman. RoI Estimates from SAVI's Feasibility Demonstration. In *National Defense Industrial Association Systems Engineering Symposium*, 2011.

[27] wikipedia. Usability, 2015. `http://en.wikipedia.org/wiki/Usability`.

[28] Jim Wolf. Price of Lockheed's F-35 Fighter Soars. `http://www.reuters.com/article/2010/03/12/lockheed-fighter-idUSN1123180820100312`, March 2010.

[29] Edgar Zapata. A Review of the NASA Space Shuttle and Human Space Flight Fixed and Variable Space Transportation System Costs. `http://science.ksc.nasa.gov/shuttle/nexgen/Shuttle_FixVar.htm`, July 2009.