Software Engineering Institute

AVSI

System Architecture Virtual Integration: An Industrial Case Study

Peter H. Feiler
Jorgen Hansson
Dionisio de Niz
Lutz Wrage

November 2009

TECHNICAL REPORT
CMU/SEI-2009-TR-017
ESC-TR-2009-017

Research, Technology, and System Solutions (RTSS) Program

http://www.sei.cmu.edu

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

Table of Contents

List of Figures

List of Tables

Acknowledgments

System Architecture Virtual Integration (SAVI) is an industry initiative by a number of aerospace companies and government organizations to improve the engineering practice for software-reliant aircraft systems under the umbrella of the Aerospace Vehicle Systems Institute (AVSI). The proof-of-concept phase of this initiative was carried out by representatives of the following member companies and organizations: AVSI, Boeing, Airbus, Lockheed Martin, Rockwell Collins, BAE Systems, GE Aviation, U.S. Army, and Federal Aviation Administration (FAA). The Software Engineering Institute (SEI) contributions were funded by SAVI members through AVSI. In particular the authors would like to acknowledge the contributions by the other members of the proof-of-concept (POC) demonstration project, which is the focus of this case study report, to the definition of a to-be process and the development of a return on investment (ROI) model. Keith Appleby from BAE Systems, John Glenski from Rockwell Collins, Jean-Jacques Toumazet from Airbus, Joe Shultz from GE Research, and Dave Redman from AVSI actively participated in creating the aircraft model for the demonstration and the Microsoft Excel spreadsheet version of the mass analysis. We would also like to thank the other members of the full SAVI team who participated in shaping the requirements for the POC demonstration and use cases.

Executive Summary

The aerospace industry is experiencing exponential growth in the size and complexity of onboard software. It is also seeing a significant increase in errors and rework of that software. All of those factors contribute to greater cost; the current development process is reaching the limit of affordability of building safe aircraft. The size with respect to source lines of code (SLOC) has doubled every four years since the mid-1990s; the 27M SLOC projected for 2010-2020 is estimated to exceed $10B.

The Aerospace Vehicle Systems Institute (AVSI) has launched an international, industry-wide initiative called System Architecture Virtual Integration (SAVI) to reduce cost/cycle-time and risk (i.e., rework) by using early, and frequent, virtual integrations. In this context, the SAVI paradigm necessitates

- an architecture-centric, multi-aspect model repository as the *single source of truth*
- a component-based framework in support of model-based and proof-based engineering
- a model bus concept for consistent model interchange between repositories and tools
- an architecture-centric acquisition process throughout the system life cycle that is supported by industrial standards and tool infrastructure

The objective of the SAVI initiative is to put in place a model-based, architecture-centric, virtual integration practice for the next generation of aircraft. Early and continuous virtual model integration based on standardized representations ensures that

- Errors are detected as early as possible with minimal leakage to later phases.
- Models with well-defined semantics facilitate auto-analysis and generation to identify and eliminate inconsistencies.
- Automated compatibility analyses at the architecture level scale easily.
- Industrial investment in tools is leveraged through well-defined interchange formats.

To establish cost-effective management and limit risks, the proof-of-concept (POC) phase of this initiative has been executed to

- document the main differences between a conventional acquisition process and the projected SAVI acquisition process
- evaluate the feasibility and scalability of the multi-aspect model repository and model bus concepts central to the SAVI project
- assess the cost, risk, and benefits of the SAVI approach through an return-on-investment (ROI) study[1] and development of a SAVI initiative roadmap

AVSI members in the POC phase included Boeing, Lockheed Martin, Airbus, Rockwell Collins, Honeywell, BAE Systems, GE Aviation, FAA, and the U. S. Department of Defense. In a separate study, the SAE International Architecture Analysis and Design Language (AADL) industry standard, and its supporting tools, was chosen as the key technology for the POC phase. The SEI par-

[1] The ROI study is the subject of a forthcoming report (CMU/SEI-2010-TR-001).

ticipated in the POC phase, bringing expertise in AADL and the Open Source AADL Tool Environment (OSATE) with its modeling, model management, and an extensible set of analysis capabilities.

In the POC phase, a demonstration was performed to evaluate the feasibility of the SAVI approach to address the *multiple truth* problem of analyses. This demonstration used a model repository and model bus centered on an extensible architecture modeling notation with strong semantics and annotations. It included a number of virtual integration activities that replace traditional design reviews by

- recording subsystem requirements in an initial system model during request for proposals
- verifying
  - supplier model compatibility and initial resource allocations, during proposal evaluation
  - interfaces and functionality, during preliminary design integration
  - performance, during critical design integration.

For this demonstration, an aircraft model was created and analyzed over two months. First, the demonstration illustrated analyses at the Tier 1 and Tier 2 levels. At the Tier 1 level, the model was analyzed for system properties such as mass and electrical power consumption; at the Tier 2 level, it was analyzed for the Integrated Modular Avionics (IMA) architecture, by revisiting the previous analyses and adding computer resource analyses and end-to-end latency analysis across subsystems. The demonstration then explored AADL support for managing supplier subcontracting through a model repository. Negotiated subsystem specification could be virtually integrated and inconsistencies between supplier specifications such as inconsistent data representation, mismatched measurement units, and mapping into a communication protocol could be detected.

Three of the suppliers then developed a task-level specification of their subsystem and performed local validation through analysis. In one case, the subsystem was elaborated into behavioral specifications via UML StateCharts and an implementation in Ada. During this process, the suppliers routinely delivered AADL models back to the airframer (i.e., aerospace company) for repeated revalidation through virtual integration with increasing model fidelity. The demonstration model itself was developed by team members in Iowa and Pennsylvania in the U. S., France, and the UK who utilized a model repository located in Texas (USA).

After the POC demonstration, the SAVI team concluded that "the SAVI concept is sound and should be implemented with further development." In addition, ROI study provided sufficient evidence for management to authorize that the SAVI initiative proceed with three more phases. These stages are intended to take the SAVI technology from Technical Readiness Level (TRL) 3 to TRL 9.[2] Implementing these phases will involve putting a technology infrastructure in place for industrial use, adapting the methodologies to incorporate this architecture-centric approach to virtual system integration and assurance, and affecting the certification processes used by the regulators.

[2] TRL is an assessment of a technology's maturity; U. S. government agencies and many large companies use this assessment before adding a technology. TRL 3 is characterized by a proof of concept; TRL 9 indicates a technology is proven by application. For more, see http://en.wikipedia.org/wiki/Technology_readiness_level.

Abstract

The aerospace industry is experiencing exponential growth in the size and complexity of onboard software. It is also seeing a significant increase in errors and rework of that software. All of those factors contribute to greater cost; the current development process is reaching the limit of affordability of building safe aircraft. An international consortium of aerospace companies with government participation has initiated the System Architecture Virtual Integration (SAVI) program, whose goal is to achieve an affordable solution through a paradigm shift of "integrate then build." A key concept of this paradigm shift is an architecture-centric model repository to support analysis of virtually integrated system models with respect to multiple operational quality attributes such as performance, safety, and reliability, and to do so early and throughout the life cycle at different levels of fidelity. The result is discovery of system-level faults earlier in the life cycle—reducing risk, cost, and development time. The first phase of this program demonstrated the feasibility of this new development process through a proof of concept demonstration which is the topic of this report.

1 Introduction

For the international aerospace industry, the cost of system and software development and integration has become a major concern. Aerospace software systems have experienced exponential growth in size and complexity, and also unfortunately in errors, rework, and cost. New development of safe aircraft is reaching the limit of affordability. Figure 1 shows that the size (measured in source lines of code or SLOC) has doubled every four years since the mid-1990s and that 27M SLOC of software are estimated by 2014. Other analyses show that the cost for 27M SLOC would exceed $10B.

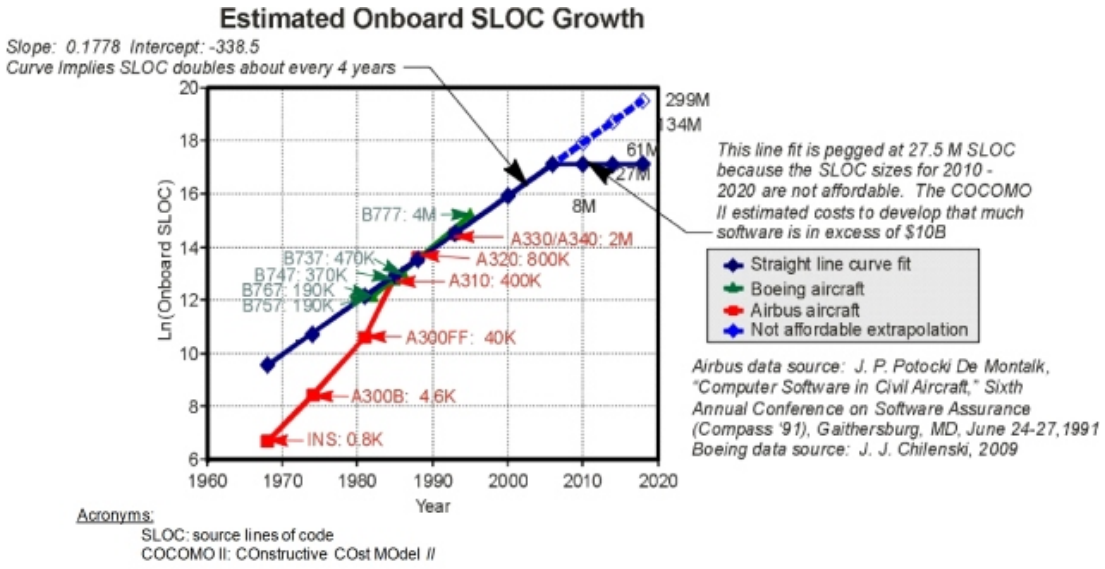

Figure 1: Estimated Onboard SLOC Growth

This problem requires an industry-wide adoption of new development practices, since many suppliers work with multiple airframers (i.e., aerospace companies) and a number of subsystems are common to multiple aircraft. One technical cause for this concern is that embedded software supports greater functionality, leading to a new class of problems not addressed by traditional system modeling and analysis.

Industrial practice has moved from federated systems to integrated modular avionics (IMA) to leverage a common distributed computing platform. The partition concept found in the ARINC653 standard, for example, provides flexibility to accommodate growth through space and time partitioning.[3] However, decisions about the runtime architecture affect system operations in unexpected ways due to mismatched assumptions. For example, Rate-Monotonic Analysis can determine whether deadlines are met for any set of preemptively scheduled fixed priority tasks. Despite this deadline assurance, flight control systems may experience increased instability due to unexpected frame-level jitter introduced by non-deterministic sampling. Similarly, an application

[3] ARINC653 or the Avionics Application Standard Software Interface allows applications to host different levels of software on the same hardware in the context of IMA.

partition may experience slower than expected execution speed due to memory and bus contention by direct memory access (DMA) transfers.

It is clear there is a need to predict the dynamics of the embedded systems. Both the system engineering and the software engineering communities are practicing model-based engineering toward this end. Models of different aspects of a system have been developed and analyzed (e.g., simulation of Simulink control models, Rate-Monotonic Analysis of timing models, and fault impact analysis performed on fault trees). However, industrial experience has shown that such models developed independently by different teams over the life cycle result in *multiple versions of the truth*—they do not remain consistent with the evolving architecture.

As the importance of architecture has been recognized, industrial standards for architecture modeling technology have emerged: OMG SysML [1] for system engineering and SAE Architecture Analysis and Design Language (AADL) [2] for embedded software systems.[4] SysML is a graphical modeling language in the form of an extensible Unified Modeling Language (UML) profile used early in system development to represent the requirements, component structure, discrete state behavior, and parametrics (i.e, physical behavior as equations) of a system. AADL is an international extensible architecture modeling language standard for embedded software system with graphical and textual representations, well-defined execution semantics and an XML interchange format. It focuses on capturing the architecture of the physical system, the computer platform, and the embedded software runtime architecture and their interactions in order to drive the analysis of multiple operational quality attribute dimensions, such as safety, reliability, and performance. The TOPCASED industry initiative has advanced the concepts of an architecture-centric model repository and a model bus that interfaces to different analysis tools as central elements of an open source software system tool infrastructure for embedded systems [3].

The Aerospace Vehicle Systems Institute (AVSI), a global cooperative of aerospace companies, government organizations, and academic institutions, has launched an international, industry-wide program called System Architecture Virtual Integration (SAVI) to reduce cost/cycle time and risk (i.e., rework) by using early (and frequent) virtual integrations. Major players of the SAVI project include Boeing, Airbus, Lockheed Martin, BAE Systems, Rockwell Collins, GE Aviation, FAA, U. S. Department of Defense (DoD), Carnegie Mellon® Software Engineering Institute (SEI), Honeywell, Goodrich, United Technologies, and NASA. The SAVI paradigm necessitates

- an architecture-centric, multi-aspect model repository as the *single source of truth*
- a component-based framework in support of model-based and proof-based engineering
- a model bus concept for consistent interchange of models between repositories and tools
- an architecture-centric acquisition process throughout the system life cycle that is supported by industrial standards and tool infrastructure

To establish cost-effective management and limit its risks, the management of the SAVI initiative ordered that a proof of concept (POC) project be carried out.

---

[4] For a list of the acronyms used in this report, see the Appendix.

® Carnegie Mellon is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

In this report, we discuss key concepts of the SAVI paradigm, describe the POC scope, and discuss the series of development scenarios used in the POC demonstration to illustrate the feasibility of improving the quality of software-intensive aircraft systems.

2 Key Concepts of SAVI

Current development processes allow 70% of faults to be introduced early in the life cycle, while 80% of them are not caught until integration test or later with a repair cost of 16x or higher. Figure 2 shows percentages for fault introduction, discovery, and cost factor of repair [4, 5, 6]. If we can use the SAVI approach of architecture-centric virtual integration and analysis to discover a reasonable percentage of system-level faults earlier in the process, we can expect cost savings larger than the additional investment in modeling and analysis.

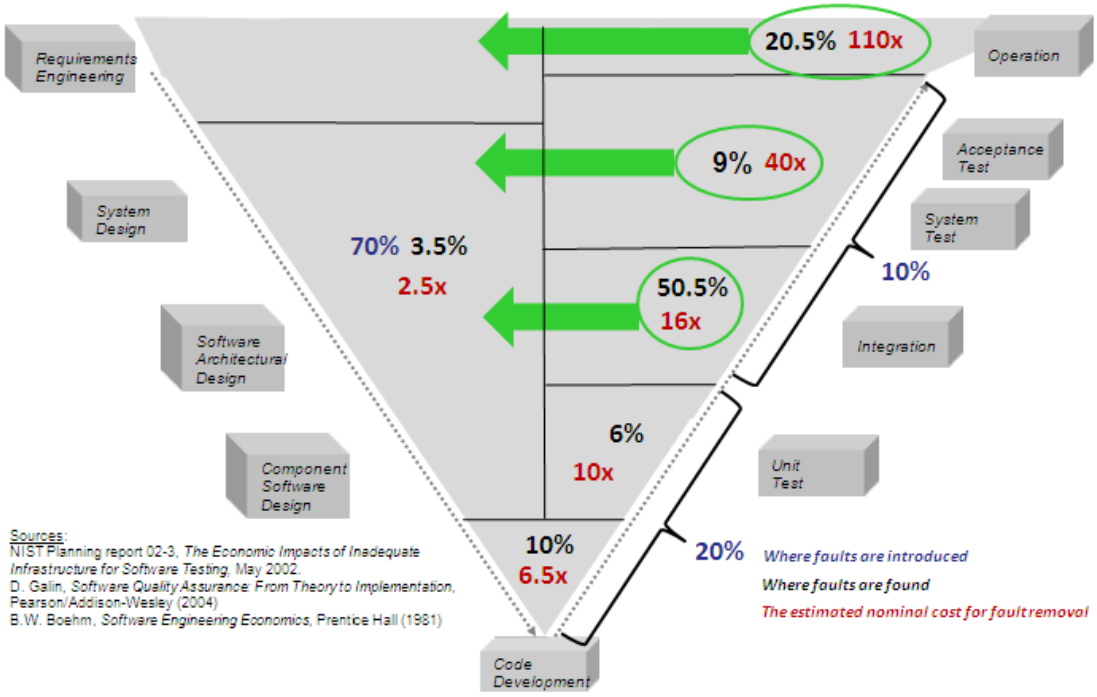

Figure 2: Benefits of Early Fault Discovery

A key concept of virtual integration is the use of an annotated architecture model as the single source for architecture analysis, as illustrated in Figure 3. In other words, instead of independently maintained analytical models, they are auto-generated from an architecture model with well-defined semantics and annotated with relevant analysis-specific information (e.g., fault rates or security properties). Any changes to the architecture throughout the life cycle are reflected in all dimension of analysis (e.g., substitution of a faster processor to accommodate a high workload not only is reflected in schedulability analysis, but also may impact end-to-end response time, and requires revalidation of increased power consumption against capacity as well as possible change in mass).

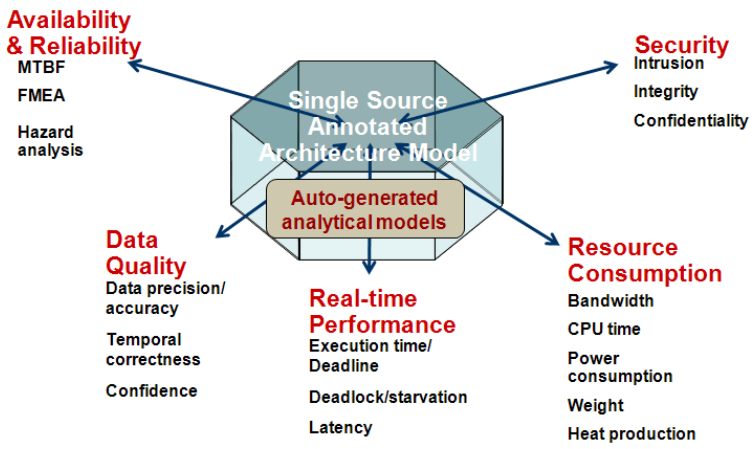

Figure 3: Single Source Annotated Architecture Model

A second key concept is the use of a model repository and model bus, illustrated in Figure 4. The model repository contains the architecture model with annotations, as well as detailed models that are refinements of architecture components. For example, details of physical system components are modeled with Modelica, computer hardware components with VHDL,[5] control system components with Simulink, and discrete application behavior with UML Statecharts, MathWorks State Flow, Scade, or programming languages (see Figure 5).

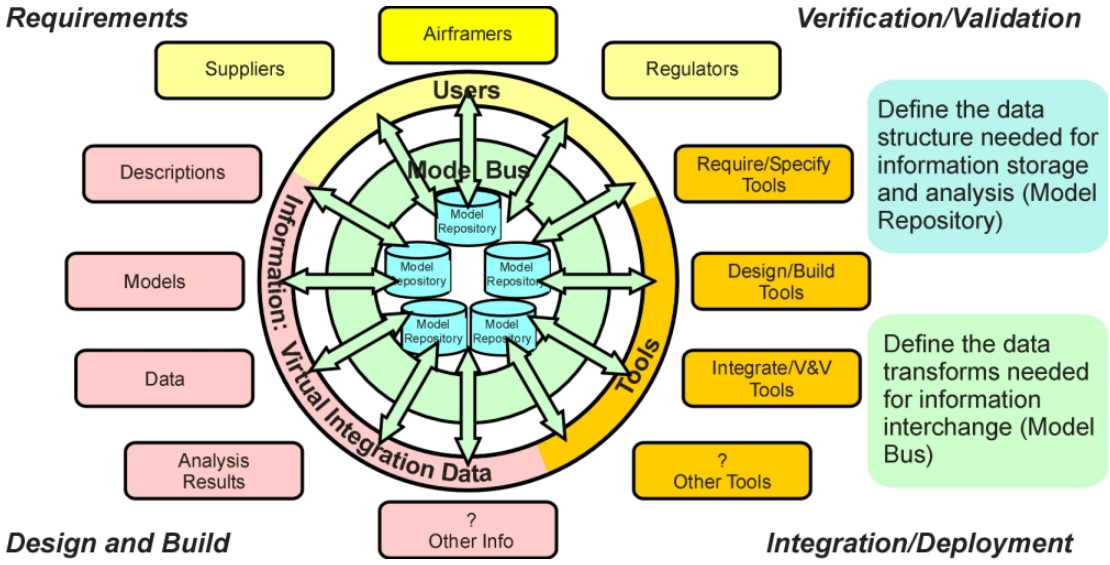

Figure 4: Model Repository and Model Bus

The model bus concept provides a data interchange mechanism between model repositories and representations acceptable to analysis and generation tools. For example, it supports the interchange of annotated architecture models in a standardized XML format. Similarly, data transform specifications provide the translation from an annotated architecture model to specific analytical model formats, such as those for timing models, fault trees, or security models.

[5] VHDL is VHSIC (Very High Speed Integrated Circuits) Hardware Description Language.

The SAVI development process must support collaboration between

- system engineers—whose primary focus is the architecture of the physical system
- other engineering roles such as the embedded software system engineers—whose focus is the interaction between the physical system architecture, the computer platform architecture, and the task and communication architecture of the embedded application
- engineers—whose focus is the architecture and detailed design of physical components, computer hardware components, and software components

Thus, the system is expressed in a combination of modeling notations whose models are mapped into the model repository without requiring manual replication of information. Figure 5 illustrates the interaction between such models.

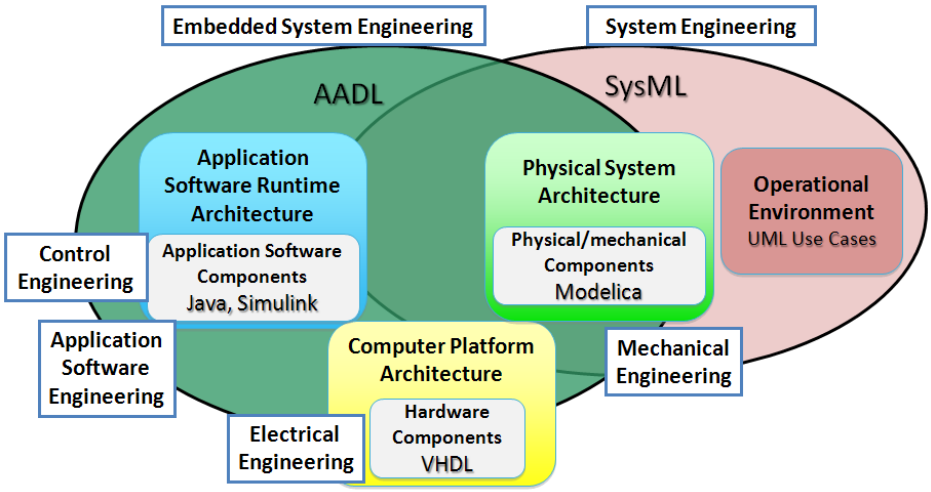

Figure 5: Collaborative Engineering

In support of the development process, the repository also includes requirements, test data, and analysis results. Standardized interchange formats such as the Requirements Interchange Format (RIF) and AP233 for system engineering artifacts are emerging. Therefore, SAVI will define the data structures needed in the model repository for information storage and analysis, and data transformations needed for data interchange and to leverage ongoing efforts in standard organizations.

As Figure 4 on page 6 shows, the airframer, the suppliers, and regulators may operate their own instances of the model repository, each supported by a standard interchange format for its content. This allows them to use different products and existing solution as the repository implementation, while relying on a standardized Meta model combined with XML to facilitate interchange. By utilizing the version and configuration management capabilities of the model repositories they are able to maintain a consistent flow of model releases between their organizations, as illustrated in one of the POC demonstration scenarios.

Standardized interchange representations for these analytical models facilitate tools chains (integration of multiple tools) to enable new and more effective integration checks and analyses by leveraging existing tools and minimizing transformations into tool-specific and proprietary representations. Examples of such emerging interchange formats include ISO/IEC 15909 for Petri nets

to interface with Petri net analysis tools [7], and FIACRE for state-based behavior specification to interface with different model checking tools [8].

SAVI virtual integration activities replace traditional design reviews by

- recording subsystem requirements in an initial system model during request for proposals
- validating
  - supplier model compatibility and initial resource allocations during proposal evaluation,
  - interfaces and functionality during preliminary design integration
  - performance during critical design integration

Early and continuous virtual model integration based on standardized representations insures that

- Errors are detected as early as possible with minimal leakage to later phases.
- Models with well-defined semantics facilitate auto-analysis and generation to identify and eliminate inconsistencies.
- Automated compatibility analyses at the architecture level scale easily.
- Industrial investment in tools is leveraged through well-defined interchange formats.

3 Description of Proof of Concept Demonstration

In order to establish cost-effective management and limit risks of the SAVI program, a POC project has been executed with the following goals:

- document the main differences between a conventional acquisition process and the projected SAVI acquisition process
- identify the potential benefits that accrue with the SAVI acquisition process
- evaluate the feasibility and scalability of the multi-aspect model repository and model bus concepts central to the SAVI project
- assess the cost, risk, and benefits of the SAVI approach while realizing increased performance, safety, and security through a return on investment (ROI) study and development of a SAVI development roadmap

3.1 Proof of Concept Demonstration Requirements

The SAVI POC team established a prioritized set of requirements, which are summarized in Table 1. As expected, they reflect the objectives of SAVI: feasibility of the model repository and model bus concepts, support for cooperative modeling of a realistic system with multiple levels of abstraction; and multiple analyses by different tools at multiple levels of fidelity driven from the same architecture model. Note that emphasis was placed on validation early in the development process. For one subsystem the inclusion of detailed design models and source code was included, but compliance checking of the source code against the model specification was not demonstrated.

Table 1: Prioritized POC Requirements

#	***Requirement***	***Category***
1	Establish Model Bus infrastructure	Process
2	Establish Model Repository infrastructure	Process
3	Inform ROI estimates through POC performance & results	Process
4	Analyses must be conducted across the system	Analysis
5	Two or more analyses must be conducted	Analysis
6	Analyses must be conducted at multiple levels of abstraction	Analysis
7	Analyses must validate system model consistency at multiple levels of abstraction	Analysis
8	Analyses must be conducted at the highest system level abstraction	Analysis
9	Model infrastructure must contain multiple model representations	Model
10	Model infrastructure must contain multiple communicating components	Model

The SAVI POC team decided to model an aircraft system at three tiers:

1. Tier 1, the aircraft from a system engineering perspective
2. Tier 2, the aircraft IMA system as an embedded software system
3. Tier 3, along with elements of the IMA at the subsystem/line replaceable unit (LRU) level

The team then identified the set of analyses for each of those tiers, propagating and validating requirements and constraints across model levels and across multiple operational quality dimensions. In addition, the POC project was to demonstrate the feasibility of an architecture-centric model repository supporting the business process of airframer/supplier interaction.

3.2 The Aircraft System Model

Figure 6 shows the drawing of the aircraft system provided to the POC team. It shows major physical subsystems, some providing aircraft capability, such as navigation or landing gear, and others providing physical resources to the subsystems, such as the electrical power, hydraulics, and fuel.

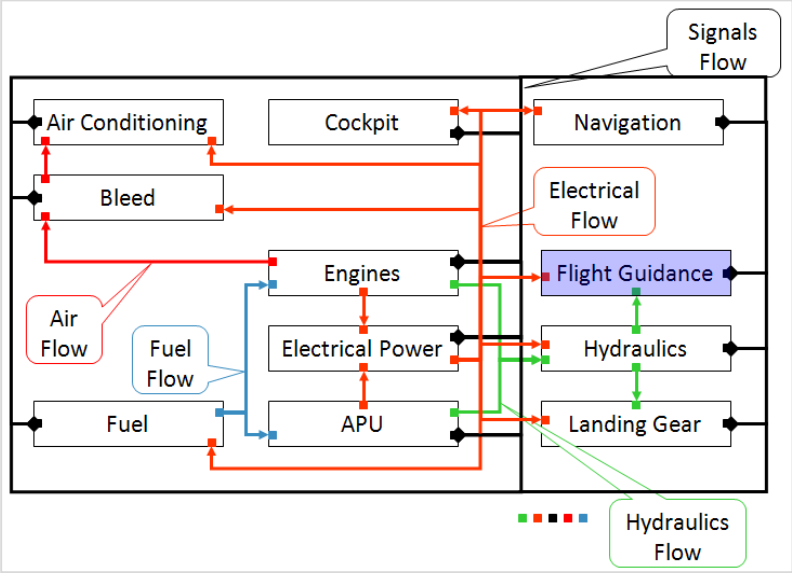

Figure 6: Aircraft System Drawing

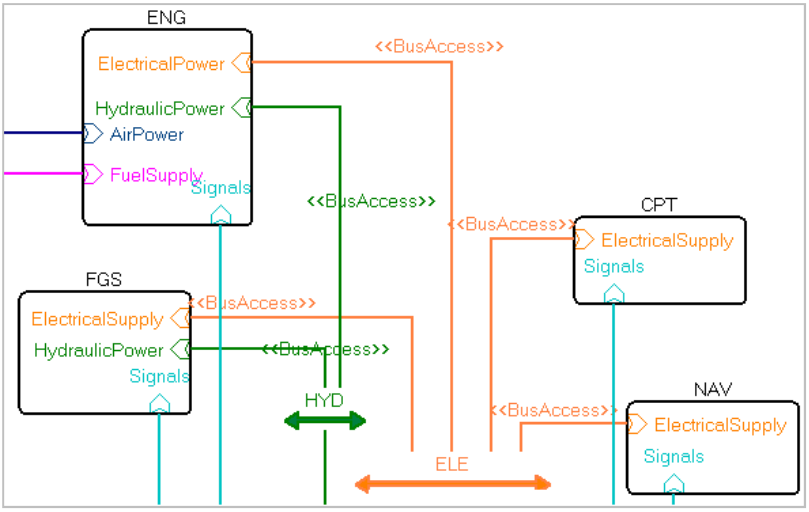

Figure 7: AADL Model of Aircraft System

Figure 7 shows a portion of the corresponding AADL model. In the model, we have represented the physical subsystems as AADL systems that can later be refined and the physical resources as AADL buses. Each aircraft subsystem is represented by a separate AADL system type whose specification includes properties about the physical characteristics (e.g., mass) of the subsystem. A separate bus type has been defined for each type of resource. Bus access connections represent the physical connection between subsystems and their resources. The bus types and access connections also have mass properties. In addition, each bus type has a resource capacity property, and the bus access features (connection points) have resource supply properties, such as the engine contributing electrical power to the electrical power resource, and resource budget properties, such as the cockpit drawing electrical power.

We have elaborated the flight guidance system (FGS) of this Tier 1 model into a Tier 2 model representing the distributed computer platform (physical view) and the embedded application subsystems (logical view) of the IMA subsystem. This elaboration is not a separate model, but a refinement of the FGS system model using the AADL *extends* mechanism. Because of this refinement, we can now specify a Tier 1 variant and a Tier 2 variant of the aircraft model and instantiate both for analysis from a single source.

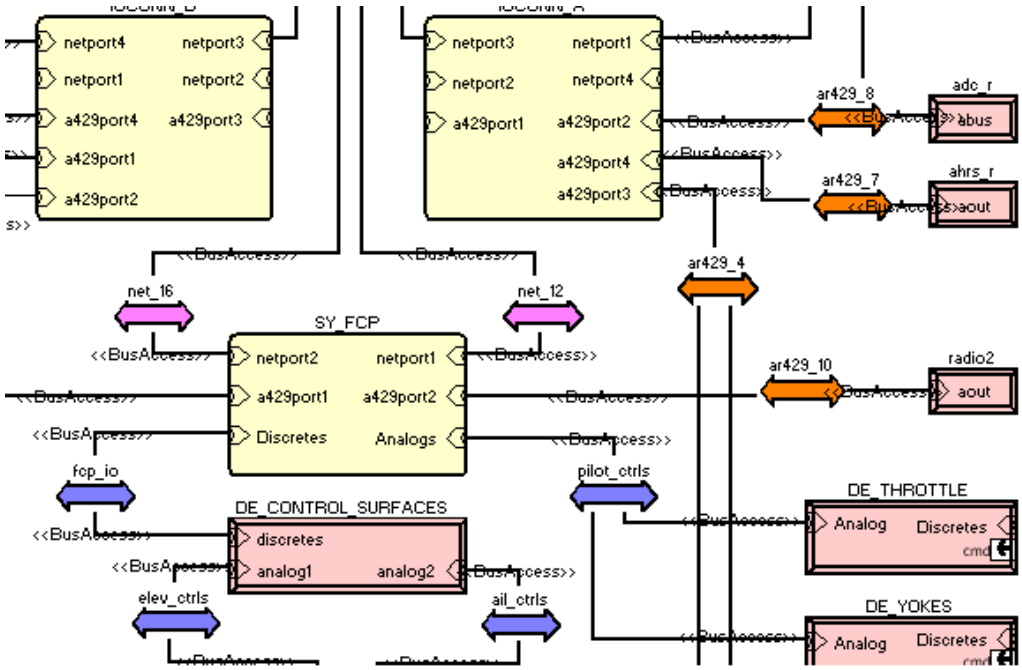

Figure 8: IMA Computer Platform

Figure 8 shows a portion of the physical view—that is, devices to represent sensors and actuators to the physical system, buses to represent networks such as ARINC429, and systems to represent processing units and communication units. The symmetry reflects the dual redundant nature of the IMA platform.

The Tier 2 model also elaborates the electrical power distribution by a power subsystem that receives its supply from the main power system and provides it to the various computer hardware components, which is captured in a graphical view separate from the physical view in Figure 8.

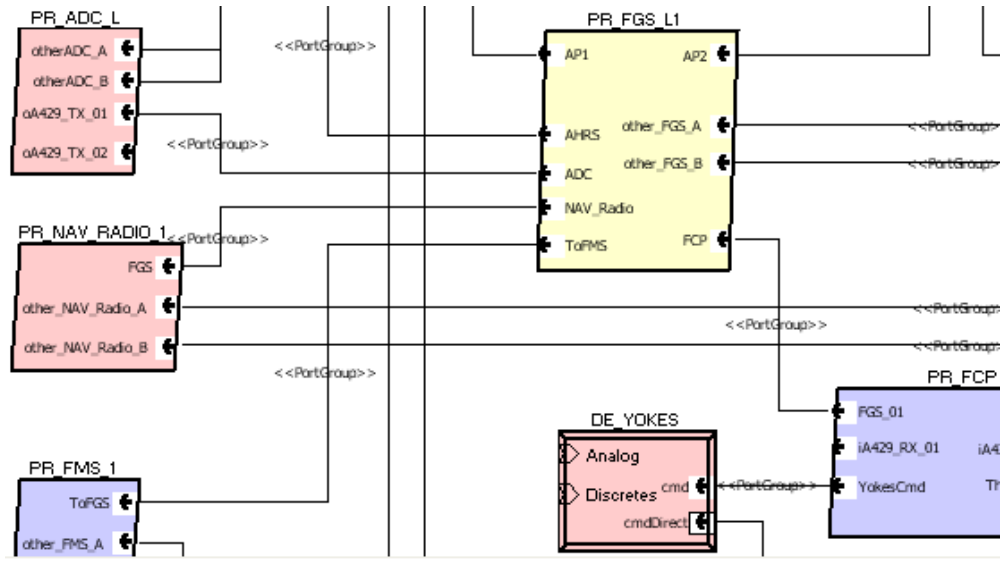

Figure 9: IMA Embedded Application Subsystems

Figure 9 shows a portion of the logical view as a collection of embedded application subsystems. We have used AADL port groups and connections to model interaction between subsystems. Port groups represent a collection of individual port connections, which suppliers later elaborate through port group types.

We have also included two end-to-end flows in order to analyze the stick–to-surface response time when operating in direct mode and with flight guidance and autopilot involved. Their textual representation is shown in Figure 10 indicating an expected latency not to exceed 150 microseconds and 25 milliseconds.

```
flows
  fStickToSurface_DirectMode: end to end flow DE_YOKES.fCmdDInput
     -> PortGroupConnection39 -> DE_CONTROL_SURFACES.fSurface
     {
      Latency => 150 Us;
      };
  fStickToSurface_NormalModes: end to end flow DE_YOKES.fCmdInput
     -> PortGroupConnection35 -> PR_FCP.fYokesToFGS1
     -> PortGroupConnection33 -> PR_FGS_L1.fPFCLAWS1
     -> PortGroupConnection23 -> PR_AP_L.fAPCLAWS1
     -> PortGroupConnection10 -> PR_SERVO_AIL.fSurfaceDemandB
     -> PortGroupConnection37 -> DE_CONTROL_SURFACES.fSurface
     {
      Latency => 25 Ms;
      };
```

Figure 10: End to End Flow Specifications

IMA computer resources (MIPS[6] for processors, MB for RAM and ROM, and bandwidth for networks), in addition to weight and electrical power are specified as properties, as shown in Figure 11. Similarly, we assigned computer resource budgets to the application subsystems and end-to-end latency requirements to the flows.

[6] MIPS is Millions of Instructions per Second.

```
processor PowerPC
  features
    backplane: requires bus access VME.impl;
    P5V: requires bus access PowerRail.impl {
      SEI::PowerBudget => access 35.0 W;
      };
  properties none ;
end PowerPC;

processor implementation PowerPC.PPC705
  properties
    SEI::NetWeight => 0.5 kg;
    SEI::MIPSCapacity => 66.0 MIPS;
    SEI::cycle_time => 15 Ns;
end PowerPC.PPC705;
```

Figure 11: Computer Hardware Specification

In our demonstration, seven of the IMA subsystems are contracted out to suppliers. The suppliers first refine the subsystem specification with interface details such as the

- exact number of ports
- type of port being sampled (data, message, or event)
- data type
- base type
- measurement unit
- input or output rate
- mapping into protocols such as ARINC429

Then they elaborate their subsystem into a Tier 3 model in terms of application tasks and communication between them. The tasks (as AADL threads) have periods, deadlines, and worst-case execution times. For sampled processing, the connections indicate whether mid-frame or phase-delayed communication is desired to minimize latency jitter.

The Air Data Computer (ADC) is a blackbox subsystem (represented by an AADL device), which the supplier elaborates into an ADC hardware platform and embedded application software. In addition to the task model, the ADC supplier has documented the application design details in UML and included an Ada-based implementation including build scripts and a local test harness. The UML diagrams and the source code are associated with the appropriate AADL model components by properties and stored in the model repository together with the AADL model.

At various times during this development process, the airframer virtually integrates the model and performs Tier 1, Tier 2, and Tier 3-level analysis. In that context, the airframer evolves the IMA model and aircraft model using AADL refinement mechanisms (*extends*) to specify configurations that include subsystems in their Tier 2 or Tier 3 elaboration and software to hardware deployment bindings.

3.3 Why AADL

An independent study that evaluated architecture languages and related technologies identified AADL as a prime candidate for the SAVI POC project. AADL was originally published in 2004 under the SAE International Avionics Systems Division with participation by a number of aerospace companies from the U.S. and Europe and revised in 2009 [2]. This industry standard was

specifically designed for modeling safety-critical, software-reliant systems by capturing not only the software design architecture, but also the runtime architecture, computer platform, and the physical system.

AADL introduces concepts to address each of those architecture views: the dynamics of architecture configurations in terms of system modes, the allocation of application software to the runtime architecture, and the deployment of the runtime architecture on the computer platform. The concepts are specified in terms of well-defined semantics to support unambiguous translation into analytical models. AADL has been designed to be extensible to support analysis of operational quality attributes along multiple dimensions. Such annotated semantic architecture models are the single source for different types of analysis. They support modeling of systems at increasing levels of fidelity early and throughout the development life cycle.

The standard suite includes a specification of the AADL Meta model, which is the core of the standardized AADL XML interchange format [33]. The Meta model-based specification of a model representation has the advantage of allowing the specification of additional constraints on models (e.g., expressed in the Object Constraint Language (OCL)) [25]. Furthermore, the meta model becomes the basis for generating model bus transformations (e.g., ATL [26] and Acceleo [27] both part of the TOPCASED tool suite [9]).

The SEI had implemented the full language including support for the AADL XMI standard and various architecture analyses with the Open Source AADL Tool Environment (OSATE) [12]. OSATE has been made available to the community at no cost and has been used for pilot projects and as a prototyping platform for integration of in-house, commercial, and research tools for analysis and auto-generation. The OSATE implementation leverages Eclipse, the Eclipse Modeling Framework (EMF) [24], and TOPCASED [3, 9]. TOPCASED provides a model bus registry for model transformers, such that they can be invoked automatically by the infrastructure. EMF is the same technology used for the specification of the UML2 meta model. This technology can generate XML schema from UML2-based meta models and vice versa, known as the XMI approach to XML Schema specification.

A number of industry initiatives have invested in tool infrastructure (TOPCASED) [9], methodology (SPICES) [10], and pilot projects (ASSERT) [11]. A commercial tool environment (STOOD) [13] has integrated AADL into its life-cycle support ranging from requirements to detailed design and system build. A UML profile for AADL is in development in cooperation with the OMG MARTE initiative [14] to provide a path into the UML tool base. Prototype implementations of a UML profile for AADL on commercial UML tools such as Rhapsody have demonstrated the feasibility of a round trip between a UML-profile-based AADL model representation and the AADL meta model standard based representation.

4 Proof of Concept Scenarios

The requirements for the POC demonstration include several development use scenarios:

1. aircraft system modeling
2. modeling of the IMA system as embedded software system
3. subcontracting support between airframer and suppliers
4. subsystem development by supplier
5. virtual integration testing by airframer

4.1 Tier 1 Aircraft System Modeling Scenario

The first scenario illustrates analysis early in the life cycle based on the Tier 1 model, whose results are revalidated throughout the life cycle as the fidelity of the model increases. The Tier 1 model is an aircraft system model of the physical subsystems.

4.1.1 Analysis Reporting

Since AADL is a strongly typed language, checking of the language syntax and semantics results in a certain level of system architecture consistency without specialized analysis tools. For example, the landing gear system type has been specified to require access to the power system and the hydraulic system. The AADL semantic checker will ensure that the correct bus is connected to the bus access feature and that all features that require a connection (indicated by a property) are connected.

We have introduced two sets of properties to represent physical characteristics relevant to system engineers: mass and electrical power. For that purpose, we have introduced an AADL property set called *SEI*, in which we define the desired properties.

```
WeightLimit: aadlreal  units SEI::WeightUnits applies to (system, proces-
sor, memory, bus, device);
```

Information regarding the mass of a system is typically kept in a spreadsheet that must be manually updated and analyzed from time to time. Instead, we associate mass information with the AADL model and drive the analysis from the model. By doing this, we can analyze the mass of the aircraft for the Tier 1 model and then revisit the analysis with more details about the mass at the Tier 2 level. The analysis examines the net weight, gross weight, and weight limit of the physical system components and connections. In the case of the Tier 1 model, we can run an analysis that adds up the gross values of the Tier 1 elements and compares the total against the limit for the aircraft.

We have implemented the analysis three ways: (1) an Eclipse plug-in to the OSATE toolset, (2) a script in the Groovy scripting language integrated into OSATE, and (3) an Microsoft Excel capability (which is shown in Figure 12). In the first two ways, the analysis works directly on an instance of the Tier 1 aircraft model; in the case of Excel, a model bus transformation translates the

relevant data into the CSV (comma separated values) file format for import into Excel (or other spreadsheet tools).

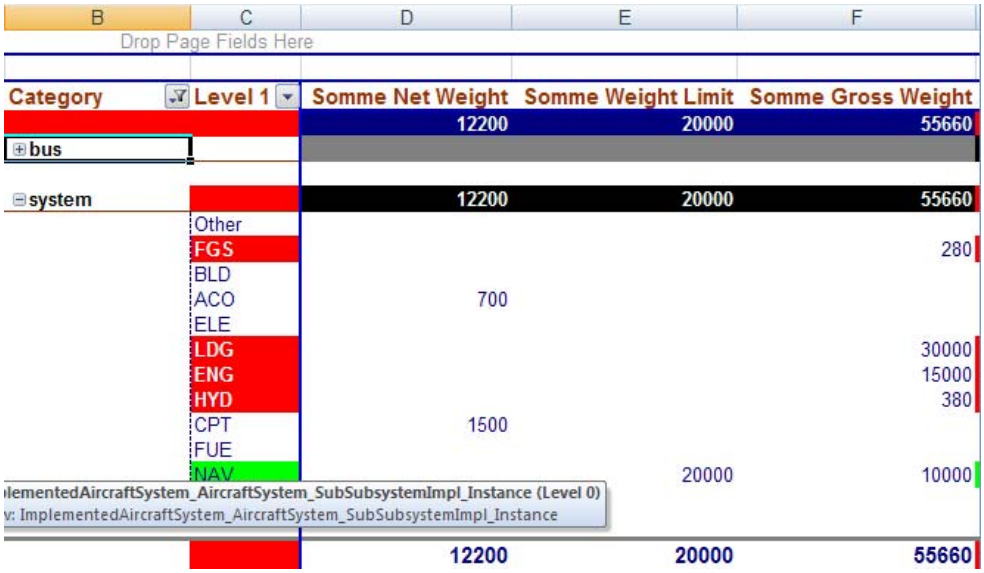

Figure 12: Excel-Based Mass Analysis

The electrical power information is recorded in the AADL model as *PowerCapacity*, *PowerSupply*, and *PowerBudget* properties. The analysis, in this case implemented as an OSATE plug-in, compares the power supplied to the power system (an instance of bus type *PowerSystem*) from both the engine and the auxiliary power unit against the power system capacity as well as the power budgets of components drawing power from the power system. This analysis can be revisited when the Tier 2 model of the IMA is available to look at the power distribution of the IMA power subsystem to the computer hardware components and compare the demand against the power budget assigned in the Tier 1 model.

4.1.2 Use Scenario

In our use scenario, the analysis reports that

1. The power supplied by the engine and the auxiliary unit exceeds the capacity of the power system.
2. The power budgets also exceed the capacity but are less than the power supply.

In the report, we can select the component in question (ENG) to see its specification. As a remedy, we choose a higher capacity variant of the power system (from the component specification library, an AADL package) and rerun the electrical power analysis. The second analysis shows positive results. We also rerun the mass analysis from the same model to ensure that the change in the power system has not exceeded any weight limits.

The AADL model of the system drives various high-level quantitative system analyses. From the same model, we can perform analyses for hydraulic pressure, fuel flow, and airflow once the component specifications (bus types and system types) are annotated with the relevant properties. The AADL component specification of these physical subsystems can be refined to more realisti-

cally reflect dynamics (e.g., fluid dynamics) or even associate detailed physical models using a specialized notation, such as Modelica, with a component.

4.2 Tier 2 Embedded Software System Modeling Scenario

The Tier 2 model refines the IMA part of the system into a networked computing platform and an interacting set of application subsystems and blackbox subsystems, which will get subcontracted to suppliers. However, before doing so, the airframer will analyze the Tier 2 model to revalidate the mass and electrical power results, by taking into account the more detailed architecture specification, and to validate properties specific to the elaborated IMA subsystem (i.e., computer resource usage and end-to-end flow response time). The analysis shows that the IMA power subsystem draws less power from the main power system and that the power consumption by the computer platform is at 60% of the locally available power. Therefore, we could consider reversing the earlier main power system upgrade (see Section 4.1.2).

4.2.1 Analysis Reporting

The computer resource analysis comes in two variants:

1. budget totals against capacity totals
2. budgets of deployed application components against the target resource if deployment decisions have been recorded

This computer resource analysis was demonstrated with an OSATE plug-in and can also be supported through a spreadsheet interface similar to the mass analysis (see Section 4.1.1).

4.2.2 Use Scenario

In our use scenario, the analysis reports that the MIPS budget totals exceed the total capacity of all processors. It also indicates that only a subset of the application components has a MIPS budget. The memory budget totals reflect 70% of the components and are below 20-40% of capacity. We can consider these alternatives for the processor resource:

- investigate whether the budgets are realistic by identifying their source and comparing them against historical data
- consider alternatives for increasing capacity

In our use scenario, we reduce the budgets and expect suppliers to meet them with their Tier 3 models.

At this point in the life cycle, or at a later stage, the system architect may make a first attempt at an allocation of major application subsystems to hardware. A variant of the computer resource analysis will consider the deployment in its results. In this demo, we perform the deployment analysis with the virtually integrated Tier 3 models.

Sampling jitter and changes in latency due to implementation decision regarding the runtime architecture can affect the stability of control systems [15]. The end-to-end latency analysis [16] at the Tier 2 level takes into account

- processing latency in the stick and surface (represented by a latency property on the flow specification of the respective AADL device type)

- communication latency associated with the connections involved in the flow
- processing and sampling latency of IMA subsystems involved in one of the two end-to-end flows

The subsystem executing as a partition with a specified partition rate contributes sampling latency. The latency analysis [17] calculates the minimum worst-case latency for the two flows (a lower bound that can only increase as the model is refined) and reports that in direct mode the response time requirement is met (121 microseconds versus a 150 microsecond requirement), while the IMA-based response time is almost twice the original requirement (46 milliseconds versus the required 25 milliseconds). A detailed analysis report tells us that the main contributor to the increased latency value is the sampling latency of the partition. We could reduce the latency by not sampling (moving to a data-driven architecture), double the partition execution rate (doubling our processor resource requirements), or renegotiate the response time requirement as an inherent property of the chosen runtime architecture. In our demo scenario, we pursue the latter option.

4.3 Airframer-Supplier Subcontracting Scenario

In support of subcontracting, we have organized the AADL model into a number of separate AADL packages that are version controlled through a model repository. For the demonstration, the AVSI organization hosted this model repository on a Subversion server with POC demo team members playing the airframer and the supplier roles located at two sites in the U. S. and two sites in Europe.

4.3.1 Repository Organization

We organized the repository into different access-controllable public and internal areas for the airframer and the subcontractors, as shown in Figure 13. We could have distributed this logical structure across separate physical repositories for the suppliers, with the Eclipse-based interface hiding the fact that they may be from different vendors. The standardized AADL XMI representation [33] enables inter-repository model interchange.

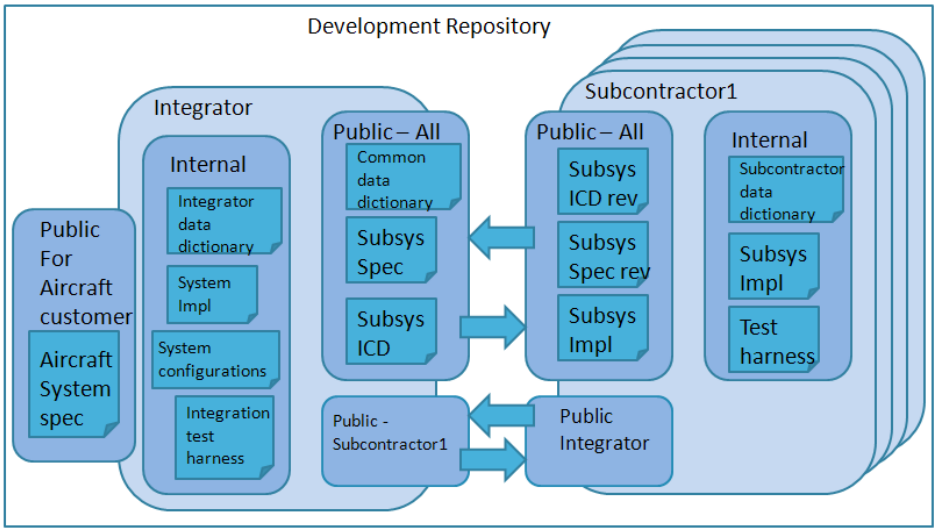

Figure 13: Distributed Model Repository

4.3.2 Use Scenario

In the scenario, as part of a request for proposal (RFP), the airframer makes available the AADL specification of the desired subsystems with a possibly partial interface specification (system types) including resource budget properties as well as expected latency requirement on flow specifications through the subsystem, and an interface control document in the form of port group type specifications. We have done so for seven subsystems to be contracted out.

The suppliers respond, in this scenario, with a completed subsystem specification including details about the exchanged data and its mapping into the ARINC429 protocol. Suppliers of subsystems that interact independently specified some of their data representations and the ARINC429 mapping. As a result, the airframer must ensure that these subsystem specifications are compatible before letting the suppliers proceed with development. The airframer verifies this compatibility by virtually integrating the AADL subsystem specifications from the suppliers as a variant of the Tier 2 model and performing functional integrity checks. Figure 14 shows reported inconsistencies that are traceable to the model (data that is also made available in spreadsheet and as a printable report formats).

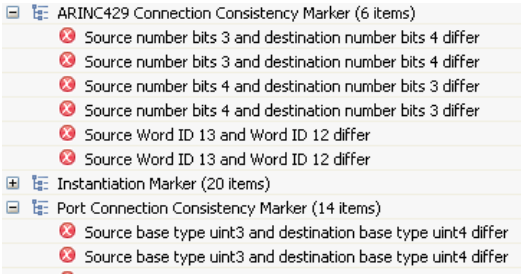

Figure 14: Functional Integrity Checking

4.4 Tier 3 Supplier Subsystem Development Scenario

The suppliers refine their subsystem AADL models to model their architecture and reflect implementation decisions. This example included three suppliers expanding their subsystem. In the case of the air data computer, we have included UML diagrams, Ada code, a test harness and automatic build scripts, as shown in Figure 15. In this case, the supplier allocates threads of the application task model to computer hardware and performs scheduling analysis. The scheduling protocol property of the processor determines which scheduling analysis algorithm is used. The reported results indicate that the subsystem is schedulable on its internal hardware with 45% utilization.

For scheduling analysis the worst-case execution time of threads is utilized. This figure may be an estimate early in the development that is scaled for processors of different speed. Once the source code exists and has been benchmarked on different processors the benchmark figures replace the estimates in the model resulting in higher fidelity results.

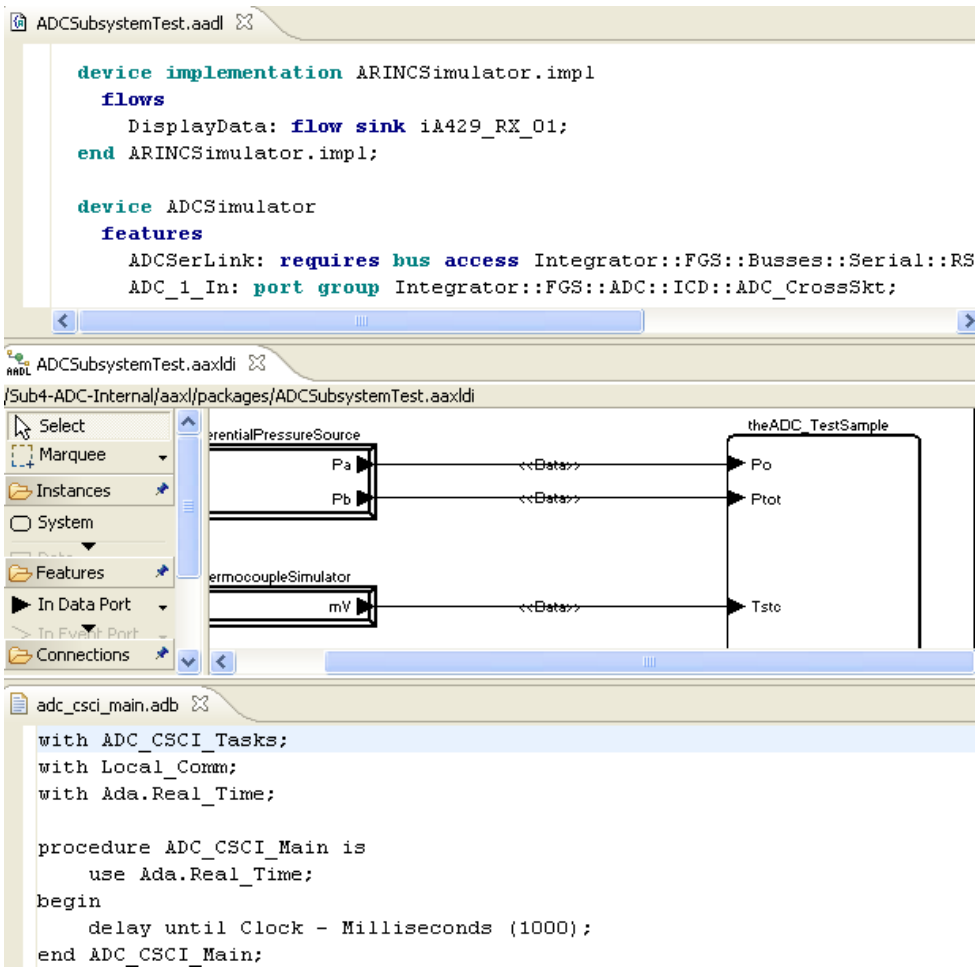

Figure 15: Supplier Subsystem Model & Analysis

4.5 Virtual System Integration Testing Scenario

At various stages of the development, each supplier delivers an AADL model with various non-functional properties of its subsystem architecture to the airframer. These updated models identify properties of the subsystems pertinent to integration, but do not necessarily include detailed design descriptions. The airframer virtually integrates them into an aircraft model refined down to Tier 3, while the supplier independently validates that the model properties of the architectural components reflect the detailed design.

4.5.1 Analysis Reporting

The models include properties that can be used for traceability to requirements. The airframer queries the model repository to determine which subcontracted subsystems are involved in satisfying certain requirements. This allows the airframer to focus on virtually integrating and analyzing the system with respect to requirements of greatest concern.

4.5.2 Use Scenario

The airframer revisits the mass and power analysis to include the Tier 3 details in the results. Similarly, computer resource analysis aggregates periods and execution times of the subsystem task

models, compares them against the assigned budgets and rolls them up for a comparison against capacity. The analysis results show that many subsystems were able to stay within the reduced budgets from the Tier 2 analysis.

As part of the scenario, the airframer revisits the end-to-end latency analysis, now taking into account any latency or latency jitter contributed by the subsystem task models that exceed the expected latency as recorded in the flow specification of the subsystems in the Tier 2 model. When task models are first delivered by the supplier, the airframer reruns the end-to-end latency analysis, which now takes the task model into account in its latency calculation. The airframer discovers that the minimum worst-case end-to-end latency for the IMA mode has increased considerably to 185 milliseconds on a synchronous hardware platform and to 196 milliseconds for processors operating on independent clocks. Examination of the detailed analysis report reveals a low-rate thread in one of the subsystems contributing a sampling latency of 100 milliseconds. Such a problem would normally only be discovered during system integration test.

The airframer allocates the application tasks from the different supplier task models to the various processors and performs scheduling analysis. Again, the scheduling protocol property on the processor determines which scheduling algorithm is used to determine schedulability. For a given deployment configuration of a three-processor system, the analysis reports that all deadlines are met with processor utilizations of 54%, 55%, and 75%. The airframer can validate the analysis results by applying a different scheduling analysis tool. In both cases, a model transformation is performed to generate a timing model in the representation acceptable to the analysis tools from the AADL model. A resource allocation tool [34] provides an option to explore alternative deployment configurations, showing that the system would be schedulable with 97% utilization on two processors, and suggesting a three processor allocation that better balances the task load, and supporting a quick what-if analysis of a four processor system to reduce the average processor utilization to a target of 50%.

Finally, the airframer performs network bandwidth analysis on the aircraft model with Tier 3 detail and a specific deployment configuration. This analysis identifies all application task connections that are routed over a particular network and determines the data volume from the size of the data communicated through ports and their transfer rate. This data volume is then compared against the bandwidth budget assigned at the Tier 2 level and against the capacity of each the network.

4.6 Independent Formal Validation

Two studies by Rockwell Collins and the SEI form the background to an independent formal validation exercise for the POC demonstration. In a Rockwell Collins report for NASA [18], the mode logic of the dual flight guidance (DFG) system had been modeled with Mathworks Stateflow, converted into NuSMV, and model checked for consistency under nominal operation and fault conditions in both synchronous and asynchronous system conditions. The SEI then investigated an architecture-model based approach to perform this verification by taking advantage of AADL semantics [19]. In this study, the AADL model included the logic of operational modes as AADL modes and mode transitions. This model was annotated with additional properties relevant to a transformation in an Alloy model, which became the basis of applying the Alloy model checker. In that context, we identified a potential race condition for the asynchronous system sce-

nario, not discovered in the earlier study that would result in a loss of events observed by sampling. The race condition was due to the Simulink model having led the modelers to communicate events by periodically polling state variables. In AADL, event ports can be used to communicate events; as a result, events are preserved in a port queue that can be polled by a periodic receiver thread.

In this independent validation exercise, an SEI team member not involved in the development of the original model played the role of an independent formal validation (IFV) team. This team refined the dual-redundant flight guidance subsystem with operational modes and properties relevant to the analysis. The IFV team then extracted the SAVI aircraft subsystem model from the model repository and integrated the annotated subsystem using the AADL `extends` capability. The team then applied the Alloy model checker to an instance of the extended subsystem model by transforming the mode logic and relevant properties into an Alloy representation. Once validated with the model checker, the model was committed to the model repository.

5 Achieving the SAVI Vision

Over two months, a four-member team completed the initial SAVI POC demonstration of a model repository populated with the three-tier aircraft model. The model included source code for the ADC subsystem and auto-generation of an executable. During the same period, the OSATE tool-set was extended by the SEI to interface to the CSV interchange format and to implement the mass and electrical power analyses. Seeing positive results, the AVSI executive board expanded the SAVI POC demonstration with functional integrity checking and protocol mapping validation to check consistency between supplier proposals.

This POC demonstrated how the SAVI concepts support the following:

- multi-tier modeling and analysis across levels
- coverage of system engineering and embedded software system analysis
- propagation of changes across multiple analysis dimensions
- maintenance of multiple model representations in a model repository
- auto-generation of analytical models via model bus
- interfacing of multiple tools to perform the same analysis
- distributed team development via repository

Virtual integration and analysis of the system architecture demonstrated the assurance of architectural integrity through early and repeated quantitative analysis at various levels of fidelity, validation of architecture consistency across subcontracted subsystem interfaces and independent protocol mappings, and discovery of intricate operational system-level faults due to design problems in the runtime architecture through formal techniques. The early discovery of system-level faults reduces risk, lowers system life-cycle costs, and improves quality.

The design of AADL contributed to the success of the POC demonstration. This standard supports modeling relevant aspects of the physical software-reliant system with well-defined semantics, including the computer platform, the software design architecture, and the runtime architecture. Components, such as processors, devices, or threads, are represented by component interface specifications (component types) that include interaction points (ports, shared data access, service calls), flow specifications, execution and fault behavior specifications, and component characteristics such as performance and security related properties. Component interface specifications are accompanied by blueprints for multiple variants of their implementation in terms of subcomponents, connections, flows, internal execution and fault behaviors—as well as other relevant information such as specification of detailed design in source code or detailed design models such as Simulink. The AADL `package` concept supports large-scale models developed by teams and by independent subcontractors.[7]

[7] The SAVI POC demonstration did not include several capabilities supported by the AADL standard and tool support, including fault modeling for fault impact and reliability analysis through the Error Model Annex standard [20], security modeling and analysis [21, 22], and auto-code generation from AADL models combined with Scade, Simulink, ASN against distributed runtime systems including ARINC653 [23, 35].

OSATE, used as tool platform in the SAVI POC project, was built by the SEI as a no-cost open source toolset on top of Eclipse. Its role was to be a low-threshold entry point for piloting the technology, for prototyping the integration of analysis models and tools, and as a research transition platform for university research groups and industrial technology groups. Several industrial initiatives involving as many as 30 partner companies have used AADL and OSATE in pilot projects. OSATE has fulfilled this role as prototyping and research transition platform successfully as evidenced by over 150 papers in refereed conferences and journals.

The implementation of the model repository and model bus through OSATE demonstrated the importance of a Meta model with well-defined semantics. To achieve the goal of a single-source model repository, the SAVI approach needs to define the model repository content in terms of a set of annotated representations mapped into a common underlying semantic architecture model and present the content in notations and formats that engineers are familiar and comfortable with (e.g., graphical diagrams and forms-based data entry) and in transformations into analytical models (e.g., formal model checking representations, fault trees, timing models, and spreadsheet data).

The SAVI management team concluded at the end of this POC project that the SAVI concept is sound and should be implemented as practice. The initial ROI study results were favorable despite the cost of implementing this paradigm shift and conservative assumptions. Assuming 33% early fault discovery, SLOC growth less than actually experienced, and 100% cost overruns, the results based on a single aircraft development show an expected savings of 40% per year. Details of this ROI study can be found in a separate report.[8] To support this paradigm shift, the SAVI roadmap outlines a multi-phased evolution through technology demonstration, translation into a SAVI-based practice, and practice sustainment. The magnitude of such an effort requires international participation by airframers, suppliers, certifiers, and standards organizations; accordingly, the SAVI program is expanding its number of participants.

[8] The ROI study is the subject of a forthcoming report (CMU/SEI-2010-TR-001).

6 Next Proof of Concept Steps

Four areas were not fully explored by the SAVI POC project and will be addressed by a second POC activity in the next phase of the project:

- scalability of the tool set
- relationship of AADL to other industry-standard architecture modeling notations (OMG SysML and OMG MARTE)
- migration of models and source code artifacts into a SAVI environment
- end-to-end validation of systems from requirements to models and system implementation

6.1 Scalability

OSATE uses a file-based implementation of the model repository, combined with any version control systems supported by Eclipse, including CVS, Subversion, and Clearcase. The current implementation of OSATE loads complete AADL models into memory and has handled models of 300MB and more. Different Java runtime system implementations impose different virtual memory limits. To improve scalability thus, for larger models one should take advantage of EMF's capability to perform lazy loading of model fragments and to utilize a database as the model storage back-end (options investigated by one of the member companies). There also existing a commercial tool environment STOOD (www.ellidiss.com) that supports AADL as part of its end-to-end development life-cycle support. It was developed over fifteen years ago supporting HOOD and HOOD-RT and used on projects by the European Space Agency, Airbus, and others.

6.2 Relationship of AADL Other Standard Modeling Notations

The MARTE [14] and the AADL standards committees have jointly defined a UML profile for AADL, which allows developers to use UML-based tools as one option to create architecture models and still benefits from a semantic architecture model, in addition to the textual graphical representation options in AADL as well as a forms-based data entry approach commonly found in database implementations of a tool set, which was prototyped by the SEI. The Meta model of an architecture modeling language together with its associated semantics plays an important role in realizing a model repository that ensures single truth analysis results. This semantic architecture model allows analytical models to be integrated by auto-generation from semantically consistent annotations. This semantic architecture model concept also allows us to better understand the relationship to other industry standard modeling notations. The AADL standard suite includes a specification of a Meta model for AADL and specification of its execution semantics, a textual and graphical syntax, and extensions that are semantically consistent with the semantics of the core semantic model of AADL, which includes a formal specification of the thread execution and port communication timing semantics. In other words, AADL represents time as an architectural abstraction, and allows you to introduce timers and clocks as components of an implementation model.

OMG MARTE is a UML profile for embedded systems that used the semantic architecture model of AADL as its starting point. MARTE includes elements in the profile for representing time through clocks and timers. They can be used to model the implementation of the execution behavior of the runtime architecture. This can make it challenging for time-related analyses to discover the intended execution semantics from such implementation details.

The focus of the OMG SysML, a UML profile in support of system engineering, is on system engineering early in the development life cycle. Its four pillars are capturing requirements in a model form, expressing the structure of a system, characterizing the behavior of the system and its components, and capturing the dynamics through a "parametrics" mechanisms that allows modelers to use equations in an equation language not prescribed by the SysML standard. SysML can be mapped into the semantic architecture model underlying AADL with some small extensions. AADL already supports abstract components, behavior specification, and the ability to annotate a model in sublanguage notations not prescribed by the core AADL standard.

6.3 Migration of Models

The semantic architecture model can be extended using the extensibility mechanism of AADL to support the capture of requirements and their validation. The Error Model Annex standard for AADL has demonstrated how a sublanguage mechanism such as the AADL sublanguage annex mechanism can extend the semantic architecture model in a semantically consistent manner. The feasibility of this approach has been demonstrated in a project called System Verification Manager (SVM) under the DARPA MoBIES program [29,30]. It demonstrated how requirements validation can be automated for engineering environments that involve system architecture descriptions, and engineering models in the control domain, such as Simulink and Dymola.

Current industrial projects used notations such as Simulink not only to represent control components, but also the architecture of how these control components play together and interface with the simulated physical system. The challenge is to map source models into a model repository not as a collection of separate models, but in form of an annotated architecture representation with associated component models. The SVM project has demonstrated the feasibility of extracting architecture information from existing detailed system models and associate different parts of the detailed system model to different components in the extract architecture model [29,30]. Similarly, a recent feasibility study has shown how a SysML model can be complemented with Modelica models to characterize the mechanical behavior of individual components. A feasibility study by Airbus has shown that a model of a system following the co-engineering approach to system modeling illustrated in Figure 5 can result in a high-fidelity co-simulation that comes close to the actual system execution on target hardware [31].

6.4 End-to-End Validation

End-to-end validation of systems involves validation of requirements against system models and system implementations. AADL properties support basic traceability between a requirements document and models, as well as traceability from models to the implementation in the form of detailed design models and source code. We can also extend the AADL with a requirements validation annex to support modeling of requirements as claims and validation activities as application of analyses and simulations to models and the source code – as demonstrated in the SVM

project [29,30]. Automated generation of a complete software system implementation from an AADL architecture model combined with detailed design models in Simulink has been demonstrated [28]. The validation of source code against models through model checking techniques and proof-carrying code has been demonstrated by the SEI Predictable Assembly of Certifiable Code project [36]. The feasibility of adapting existing IV&V processes has been demonstrated in an SEI project with NASA and JPL [32].

References

URLs are valid as of the publication date of this document.

[1] Object Management Group, "SysML—Open Source Specification Project," *Object Management Group (OMG)* [Online]. Available: http://www.sysml.org [Accessed: Oct. 27, 2009].

[2] SAE International. "Architecture Analysis & Design Language (AADL): SAE International Standards document AS5506A, Nov 2004, Revised Jan 2009." *SAE International* [Online]. Available: http://www.sae.org/technical/standards/AS5506A [Accessed: Oct. 27, 2009].

[3] X. Dumas, et al., "Supporting a Multi-formalism Model Driven Development Process with Model Transformation, a TOPCASED Implementation" in *Proceedings of 4th International Congress on Embedded Real-Time Systems (ERTS 2008).*

[4] RTI, "The Economic Impacts of Inadequate Infrastructure for Software Testing," National Institute for Standards and Technology, Washington, DC, NIST Planning report 02-3, 2002.

[5] D. Galin, *Software Quality Assurance: From Theory to Implementation.* Boston: Pearson/Addison-Wesley, 2004.

[6] B.W. Boehm, *Software Engineering Economics.* Englewood Cliffs, NJ: Prentice Hall, 1981.

[7] Wikimedia Foundation, "Petri net XMI Interchange format, ISO/IEC 15909," Wikimedia Foundation [Online] Available: http://en.wikipedia.org/wiki/Petri_Net_Markup_Language [Accessed: Oct. 27, 2009].

[8] B. Berthomieu, et al., "Fiacre: an Intermediate Language for Model Verification in the TOPCASED Environment" in *Proceedings of 4th International Congress on Embedded Real-Time Systems (ERTS 2008).*

[9] TOPCASED, "The Open Source Toolkit for Critical Systems (TOPCASED)," TOPCASED [Online] Available: http://www.topcased.org [Accessed: Oct. 27, 2009].

[10] Support for Predictable Integration of mission Critical Embedded Systems (SPICES), Information Technology for European Advancement (ITEA). "*Welcome to SPICES!" SPICES-ITEA* [Online] Available: http://www.spices-itea.org [Accessed: Oct. 27, 2009].

[11] European Space Agency, "Automated proof-based System and Software Engineering for Real-Time Systems (ASSERT)," *European Space Agency* [Online] Available: http://www.assert-project.net [Accessed: Oct. 27, 2009].

[12] Software Engineering Institute. "Open Source AADL Tool Environment (OSATE)," *Software Engineering Institute* [Online] Available: http://www.sei.cmu.edu/dependability/tools/osate/ [Accessed: Oct. 27, 2009].

[13] Ellidiss. "STOOD: real time software toolset supporting UML 2, HRT HOOD, AADL, C, C++, and Ada 95," *Ellidiss* [Online] Available: http://www.ellidiss.com [Accessed: Oct. 27, 2009].

[14] Object Management Group, "Modeling and Analysis of Real-time and Embedded systems (MARTE)," *Object Management Group* [Online] Available: http://www.omgmarte.org [Accessed: Oct. 27, 2009].

[15] A. Cervin, K.-E. Årzén, and D. Henriksson, "Control Loop Timing Analysis Using TrueTime and Jitterbug" in *Proceedings of the 2006 IEEE Conference on Computer Aided Control Systems Design (CACSD)*, pp. 1194−1199.

[16] P. Feiler and J. Hansson, "Impact of Runtime Architectures on Control System Stability" in *Proceedings of 4th International Congress on Embedded Real-Time Systems (ERTS 2008)*.

[17] P. Feiler and J. Hansson, "Flow Latency Analysis with the Architecture Analysis & Design Language (AADL)" Software Engineering Institute Technical Note, CMU/SEI-2007-TN-010, Dec 2007.

[18] S. Miller, et al., "A Methodology for the Design and Verification of Globally Asynchronous/Locally Synchronous Architectures," NASA: Langley Research Center, Hampton, VA, NASA/CR-2005-213912, 2005.

[19] D. de Niz and P. Feiler, "Verification of Replication Architectures in AADL" in *Proceedings 14th International Conference on Engineering of Complex Computer Systems* (ICECCS 2009), pp. 365-370.

[20] SAE International, "Architecture Analysis & Design Language (AADL) Annex Volume 1: Annex E: Error Model Annex, SAE International Standards: AS5506/1," *SAE International* [Online] Available: http://www.sae.org/technical/standards/AS5506/1 [Accessed: Oct. 27, 2009].

[21] Jörgen Hansson and Peter H. Feiler, "Enforcement of Quality Attributes for Net-centric Systems through Modeling and Validation with Architecture Description Languages" in *Proceedings of 4th International Congress on Embedded Real-Time Systems (ERTS08)*.

[22] Julien DeLange, Laurent Pautet, and Peter Feiler, "Validating safety and security requirements for partitioned architectures" in *Lecture Notes in Computer Science (Volume 5570/2009).* Berlin/Heidelberg: Springer, pp. 30-43.

[23] Lasnier Gilles, Zalila Bechir, Pautet Laurent, and Jérôme Hugues, "OCARINA: An Environment for AADL Models Analysis and Automatic Code Generation for High Integrity Applications" in *Lecture Notes in Computer Science (Volume 5570/2209).* Berlin/Heidelberg: Springer, pp. 237-250.

[24] Eclipse. "Eclipse Modeling Framework (EMF)," *Eclipse* [Online] Available: http://www.eclipse.org/modeling/emf [Accessed: Oct. 27, 2009].

[25] Wikimedia Foundation. "Object Constraint Language (OCL)," *Wikimedia Foundation* [Online] Available: http://en.wikipedia.org/wiki/Object_Constraint_Language [Accessed: Oct. 27, 2009].

[26] Wikimedia Foundation. *"ATLAS Transformation Language (ATL)," Wikimedia Foundation* [Online] Available: http://en.wikipedia.org/wiki/ATLAS_Transformation_Language [Accessed: Oct. 27, 2009].

[27] Acceleo. "Acceleo: Model-driven Development," *Acceleo* [Online] Available: http://www.acceleo.org [Accessed: Oct. 27, 2009].

[28] G. Raghav, S. Gopalswamy, K. Radhakrishnan, J. Delange and J. Hugues, "Architecture Driven Generation of Distributed Embedded Software from Functional Models" in *Proceedings of the 2009 NDIA Michigan Chapter Ground Vehicle System Engineering and Technology Symposium (GVSETS2009),* August 2009.

[29] Bruce H. Krogh, Peter H. Feiler, Shiva Sivashankar, and Bill Aldrich, "SVM: System Verification Manager for Model-Based Development of Embedded Control Systems" in *Proceedings of EMSoft 2003*.

[30] Bill Aldrich, Ansgar Fehnker, Peter H. Feiler, Zhi Han, Bruce H. Krogh, Eric Lim, and Shiva Sivashankar, "Managing Verification Activities Using SVM" in *Proceedings of Sixth International Conference on Formal Engineering Methods (ICFEM*).

[31] J. Casteres and T. Ramaherirariny (Airbus), "Aircraft integration real-time simulator Modeling with AADL for architecture tradeoffs" in *Proceedings of 9th Design, Automation & Test in Europe Conference*.

[32] P. Feiler, D. Gluch, K. Weiss, and K. Woodham, "Model-Based Software Quality Assurance with the Architecture Analysis and Design Language" in *Proceedings of AIAA Infotech @Aerospace 2009*.

[33] SAE International, "Architecture Analysis & Design Language (AADL) Annex Volume 1: Annex C: AADL Meta Model & XML Interchange Format Annex, SAE International Standards: AS5506/1," *SAE International* [Online] Available: http://www.sae.org/technical/standards/AS5506/1 [Accessed: Oct. 27, 2009].

[34] Dio de Niz and Peter Feiler, "On Resource Allocation in Architectural Models" in *Proceedings of the 11th IEEE International Symposium on Object/service-oriented Real-time distributed Computing*, May 2008.

[35] J. Delange, L. Pautet, A. Plantec, M. Kerboeuf , F. Singhoff, and F. Kordon, "Validate, Simulate, and Implement ARINC653 Systems using the AADL" in *Proceedings of SigAda 2009*, Nov. 2009.

[36] J. Ivers and G. A. Moreno, " Model-driven development with predictable quality" in *Companion To the 22nd ACM SIGPLAN Conference on Object-Oriented Programming Systems and Applications Companion* OOPSLA '07, Oct. 2007.

Appendix List of Acronyms

Acronym	Description
AADL	Architecture Analysis & Design Language
ADC	Air Data Computer
ATL	ATLAS Transformation Language
AVSI	Aerospace Vehicle Systems Institute
BAE	British Aerospace Engineering
CSV	Comma Separated Values
DMA	Direct Memory Access
DoD	Department of Defense
EMF	Eclipse Modeling Framework
FAA	Federal Aviation Administration
GE	General Electric
ifv	Independent Formal Validation
IMA	Integrated Modular Avionics
LRU	Line-Replaceable Unit
MARTE	Modeling and Analysis of Real-time and Embedded systems
MIPS	Microprocessor without Interlocked Pipeline Stages
NASA	National Aeronautics and Space Administration
NIST	National Institute of Standards and Technology
OCL	Object Constraint Language
OMG	Object Management Group
OSATE	Open Source AADL Tool Environment
POC	Proof of Concept
RFP	Request for Proposal
RIF	Requirements Interchange Format
ROI	Return on Investment
SAVI	System Architecture Virtual Integration
SLOC	Source Lines of Code
SysML	System Modeling Language
SVM	System Verification Manager
TOPCASED	The Open-Source Toolkit for Critical Systems
TRL	Technology Readiness Level
UML	Unified Modeling Language
VHDL	VHSIC (Very High Speed Integrated Circuits) Hardware Description Language
XMI	XML Metadata Interchange
XML	Extensible Markup Language

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	November 2009	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
System Architecture Virtual Integration: An Industrial Case Study	FA8721-05-C-0003

6. AUTHOR(S)

Peter H. Feiler, Jorgen Hansson, Dionisio de Niz , Lutz Wrage

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2009-TR-017

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	ESC-TR-2009-017

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

The aerospace industry is experiencing exponential growth in the size and complexity of onboard software. It also seeing a significant increase in errors and rework of that software. All of those factors contribute to greater cost; the current development process is reaching the limit of affordability of building safe aircraft. An international consortium of aerospace companies with government participation has initiated the System Architecture Virtual Integration (SAVI) program, whose goal is to achieve an affordable solution through a paradigm shift of "integrate then build." Key concepts of this paradigm shift are an architecture-centric model repository as single source for analytical system models, accessed through a model bus, used as a single source for analytical models, and multi-level, multi-fidelity analysis of multiple operational quality attributes of the system and embedded software system architecture. The result is discovery of system-level faults earlier in the life cycle—reducing risk, cost, and development time. The first phase of this program demonstrated the feasibility of this new development process through a proof of concept which is the topic of this report.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Virtual integration, architectural modeling, distributed development	47

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL