All enquiries related to this publication should be referred to:

System and Software

AVSI is a cooperative of Aerospace industries and government agencies.
It is administered by the Texas Engineering Experiment station
located on the Texas A&M University campus.

Summary Final Report

produced under the

System Architecture Virtual Integration (SAVI) Proof-Of-Concept Program

AFE #58

Document ID: **SAVI-58-00-001**

Issue: **Draft 1**

Date: **25 August 2009**

Author(s): ……..D. T. Ward

Approved: ……..PMC Chair

Approved:……..AVSI Director

All enquiries related to this publication should be referred to:

System and Software

AVSI is a cooperative of Aerospace industries and government agencies.
It is administered by the Texas Engineering Experiment station
located on the Texas A&M University campus.

All enquiries related to this publication should be referred to:

System and Software

AVSI is a cooperative of Aerospace industries and government agencies.
It is administered by the Texas Engineering Experiment station
located on the Texas A&M University campus.

Contents

List of Figures

List of Tables

EXECUTIVE SUMMARY

The System Architecture Virtual Integration (SAVI) project is a cooperative research effort sponsored by the Aerospace Vehicle Systems Institute (AVSI). The primary objective is to provide a model-based scheme of developing new aerospace systems more efficiently than the current development process (the "as-is" process) produces. The results of the work completed under AFE 58, the first part of the project aimed at proving the concept is feasible to implement, are summarized in this document. AFE 58 built and exercised high level models to demonstrate that the core concepts of a Model Repository (or Repositories) and the Model Bus are viable and provide promising avenues for carrying out a level of virtual integration that promises significantly lower rework costs and shorter development times for software-intensive systems. Three different sub-teams addressed the following tasks:

- Definition of acquisition models – "as-is" and "to-be", with emphasis on the differences between them,
- Demonstration of three different interactive high level architectural models, operating at three different tiers (hierarchical levels) of system development,
- Laying out a roadmap for future development of the SAVI concept.

The results of these three activities, along with an industry-directed, contractor-supported evaluation of the return on investment for SAVI-induced changes in the development approach for complex aerospace hardware/software systems, lead to the following conclusions:

- The demonstrations carried out during AFE 58 strongly suggest that virtual integration utilizing a common Model Bus with associated Model Repositories is not only feasible, but highly desirable for future systems development.
- With a focused and collaborative effort supported by both industry and governmental agencies a production-ready SAVI methodology can be achieved by 2014.
- The return on investment for a mature SAVI process should definitely be positive with a highly conservative estimate at over 2% per year (most likely estimate exceeds 20% per year) on the first full aircraft development that uses SAVI techniques.

1 Introduction

1.1 What is Systems Acquisition Virtual Integration (SAVI)?

The System Architecture Virtual Integration (SAVI) program is a multi-year, multi-million dollar program intended to implement the capability to virtually integrate complex hardware/software systems before designs are committed to physical form. SAVI is a methodology for managing the exponentially increasing complexity of modern aerospace systems. The program is led by industry and government participants working together through the Aerospace Vehicle Systems Institute (AVSI), a research collaborative of the Texas Engineering Experiment Station (TEES), itself a member of The Texas A&M University System. Membership in AVSI includes: Airbus; BAE Systems; Boeing; the Department of Defense; Federal Aviation Administration; General Electric; Goodrich Aerospace; Hamilton Sundstrand; Honeywell International; Lockheed Martin; NASA; and Rockwell Collins.

Increasingly complex hardware and software used in critical aerospace systems pose integration problems nearing the limits of complexity effectively handled by the current system acquisition process. Boeing and Airbus data show rapid increases size and complexity of software as indicated by the doubling every four years of the onboard software lines of code (SLOC) for their commercial aircraft systems. Individual companies cannot affordably solve these problems but the industry cannot ignore them. Collaborative, industry-wide, and reusable solutions are sorely needed. System Architecture Virtual Integration (SAVI) is the first cooperative, complete, and cost-effective attempt to provide such solutions. Figure 1 illustrates the core elements of SAVI, the Model Bus and the Model Repositories, and how they interact with other elements to allow virtual integration of a system of systems.

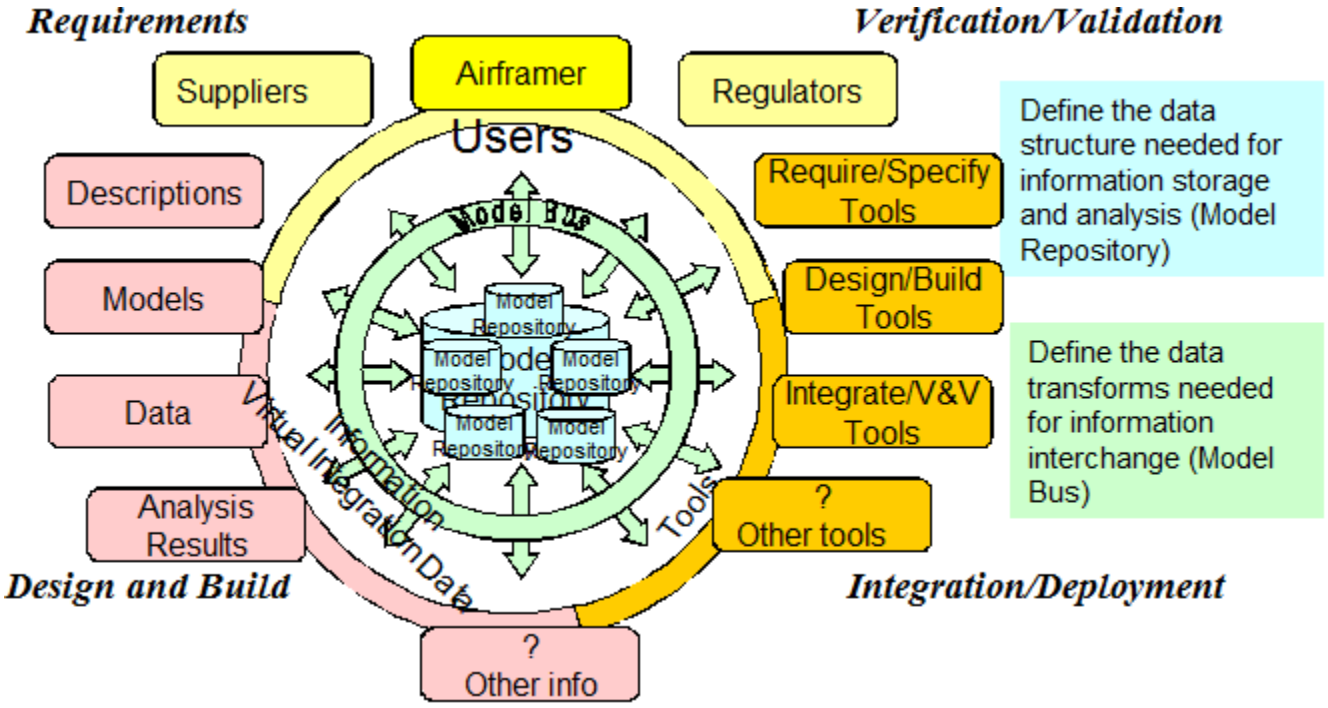

Figure 1. SAVI Core Elements and Interactions

The SAVI project is developing the definitions for the Model Repository (the data structure needed for information storage and analysis) and Model Bus (the data transformations needed for information interchange). These components enable multiple business entities to exchange

diverse forms of information utilizing multiple tools (both commercial and proprietary). SAVI's model-oriented approach encourages frequent and early interface checks ("virtual integrations") to help reduce rework, and thereby cycle-time and cost. Tools chains (integration of multiple tools) will enable new and more effective integration checks and analyses.

1.2 SAVI Proof-of-Concept (PoC) Demonstration (AFE #58)

1.2.1 PoC Objectives

AFE 58 is an AVSI development project addressing feasibility of the SAVI concept through Proof-of-Concept (PoC) demonstrations. This document summarizes results of this PoC effort described in AFE 58 [1]. The Program Management Committee (PMC) for the project created several sub-teams in August 2008, to focus energy in three distinct important areas identified by the Authority of Expenditure [1], section (iv) Project Objectives and Deliverables.

1) Identify Acquisition Process Differences Needed to Support Model-Based Specification,
2) Multi-Aspect Model Repository and Model Bus Prototype Demonstration,
3) SAVI Program Management/Technical Management Depicted by a SAVI Roadmap.

In addition to these three objective areas of evaluation, a realistic estimate of the Return on Investment for implementing SAVI concepts was one of the primary goals of AFE 58. Section 6 of this report summarizes the results of this first estimate of the cost and time reductions, emphasizing their economic impact, attainable with the SAVI Virtual Integration process.

AVSI's SAVI supporting contractor is Carnegie Mellon University through its Systems Engineering Institute (SEI). The objectives and tasks for the contractor included support for all the areas listed above, including participation in a benefits assessment supporting the RoI Analysis, which extended the schedule until August 2009 to complete this assessment.

SAVI participants include leading international aerospace companies and the U.S. government (currently the FAA and the DoD). The project seeks a system development methodology that detects integration issues early and continuously throughout system development using Model-Based Engineering (MBE) and Virtual Integration prior to physical integration. Assurance of architectural integrity and continuous integration testing throughout the development cycle minimizes the impact of errors through early detection and correction as errors are identified. Program objectives are reduced risk, lower system life cycle cost, shorter development times, better integrity, and improved overall quality.

1.2.2 PoC Process Flow

The overall process for meeting these objectives is shown in Figure 2 on the following page. This flow diagram also illustrates how the demonstrations interacted with the other elements of the overall effort. This flow diagram serves as useful frame of reference in understanding the structured approach taken by the AFE 58 Program Management Committee (PMC).

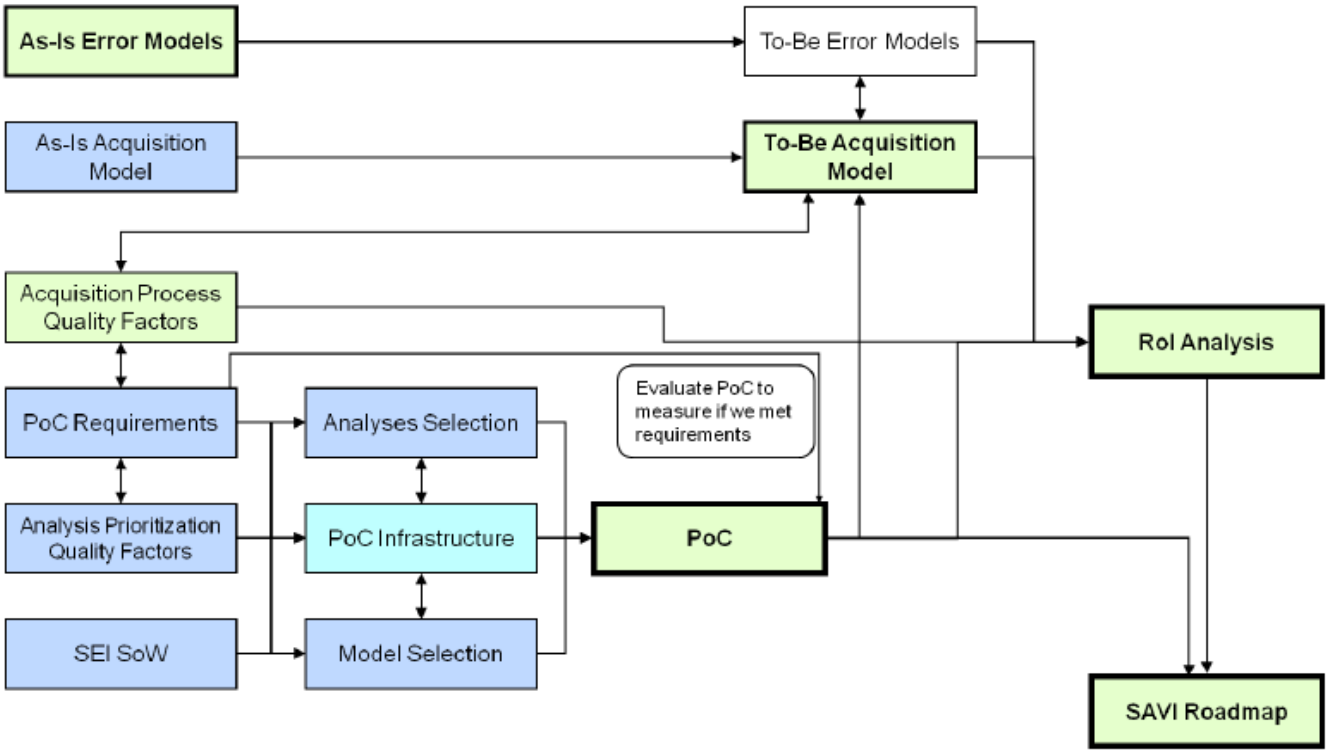

Figure 2. AFE 58 Proof of Concept Process Flow

2 References

This summary report is meant to provide a succinct single reference spelling out the most significant results from this effort for potential participants and current participants. All of the references below are available outside the project web site except the first one to help provide information to the public about the SAVI project. The sub-teams mentioned above produced a total of 13 individual reports to summarize the work done on the SAVI PoC Demonstrations, but they are not public information. The goal is for this report to summarize the results without divulging all the information that was funded by the participating organizations. (Participant organizations can currently access these more detailed reports at the following URL:

https://avsi-tees.tamu.edu/members/ssiv/AFE%2058/Shared%20Documents/Forms/AllItems.aspx?RootFolder=%2fmembers%2fssiv%2fAFE%2058%2fShared%20Documents%2fAFE58%20Deliverables%20-%20Final&FolderCTID=&View={DACE0E74-D297-4087-84A1-DBFE4E712C33}

Note: The SAVI SharePoint site is being revised to better handle the next phase of development; check with your PMC member if this site is no longer available.)

[1]	AFE #58	Chilenski, J. J., "System Architecture Virtual Integration Viability Proof-Of-Concept," AVSI Authority for Expenditure (AFE) 58 (Industry), July 2008. https://avsi-tees.tamu.edu/members/ssiv/Shared Documents/AFE 58/AFE 58 080701 Industry V1.doc
[2]	SAE AS 5506	Architecture Analysis and Design Language (AADL) 2004-11.
[3]	PoC Demo	http://www.aadl.info/aadl/savi/savi30minutedemo.exe
[4]		Potocki De Montalk, J. P., "Computer Software in Civil Aircraft," Sixth Annual Conference on Computer Assurance (COMPASS '91), Gaithersburg, MD, June 24-27, 1991.
[5]		"*The Economic Impacts of Inadequate Infrastructure for Software Testing,"* NIST Planning report 02-3, May 2002.
[6]		Allan, N.S. "A Practical Reliability Evaluation of Embedded Avionic Software," Proceedings of the 1987 IEEE Southern Tier technical Conference, 29 April 1987, pp. 238-243. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=716399
[7]		King, T., and Marasco, J., "What is the Cost of a Requirements Error?," http://www.stickyminds.com/sitewide.asp?Function=FEATUREDCOLUMN&ObjectId=12529&ObjectType=ARTCOL&btntopic=artcol
[8]		Dabney, J. B., and Costello, K., "Return on Investment for Software IV&V," Third Annual NASA Project Management Conference, March 2006.
[9]	-not yet numbered-	Helton, S. B., Hansson, J., and Redman, D. A., "RoI Analysis of SAVI (draft)," August 2009.
[10]		Boehm, B. W., Apts, C. W., Brown, A. W., Chulani, S., Clark, B. K., Horowitz, E., Madachy, R., Reifer, D. J., and Steece, B., **Software Cost Estimation with Cocomo II** , Prentice-Hall Inc., Upper Saddle River, N.J., 2000.
[11]		Boehm, B. W., Valerdi, R., Lane, J. A., and Brown, A. W., "COCOMO Suite Methodology and Evolution," *Crosstalk, the Journal of Defense Software Engineering*, April 2006, pp. 20-25.
[12]	-not numbered-	Helton, S. B., "AFE#58 RoI Analysis Summary Final Report (Draft)," PowerPoint presentation, June 2009.

3 Acquisition Models

System acquisition tends to be hierarchical. The elements of a complex system are acquired and integrated by a system integrator (SI) and are often systems (or subsystems) of lower level components acquired and integrated by a lower level system integrator usually called a supplier. This process is recursive, and thus an aircraft consists of systems of systems. There may be on the order of 13 levels in the overall aircraft system, resulting in thousands of system elements at the lowest levels. The term "system" is used to represent the higher tier in the acquisition, the entity into which a "subsystem" is to be integrated. The term "subsystem" is used to represent a lower tier in the acquisition, than the end item(s) being acquired. Acquisition models are described in this document from two perspectives: (1) the flow of activities in the process and (2) the error feedback paths used. In this summary only the activity perspective is used for the sake of brevity.

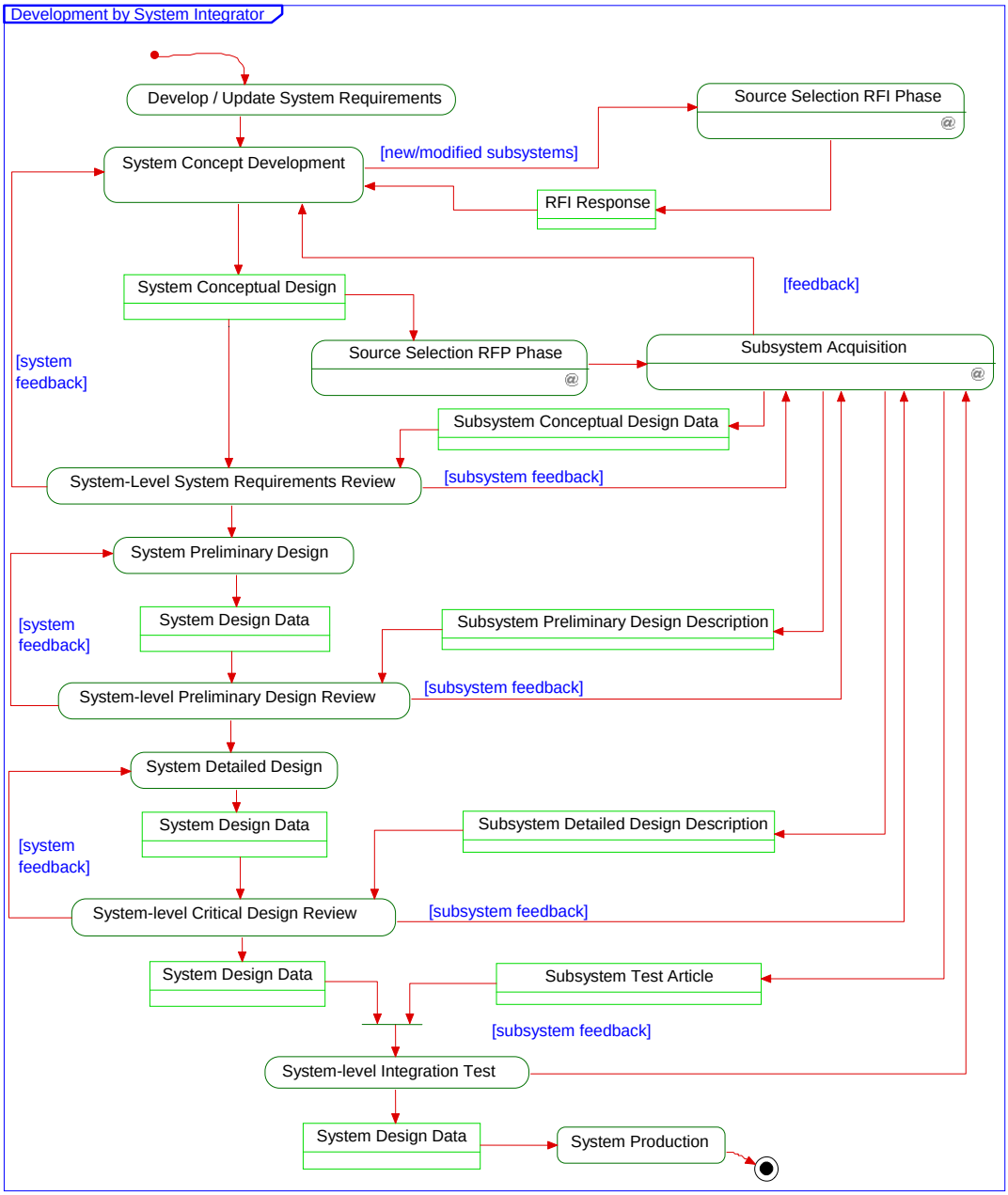

Figure 3. Activity Flow Diagram for "As-Is" Acquisition Process

3.1 Description of "As-Is" and "To-Be" Acquisition Models

A System Modeling Language (SysML) activity diagram depicts characteristics of the current system acquisition process ("As-Is") in Figure 3 (previous page) and Figure 4 (below) shows an activity diagram for the acquisition process after SAVI is implemented ("To-Be").

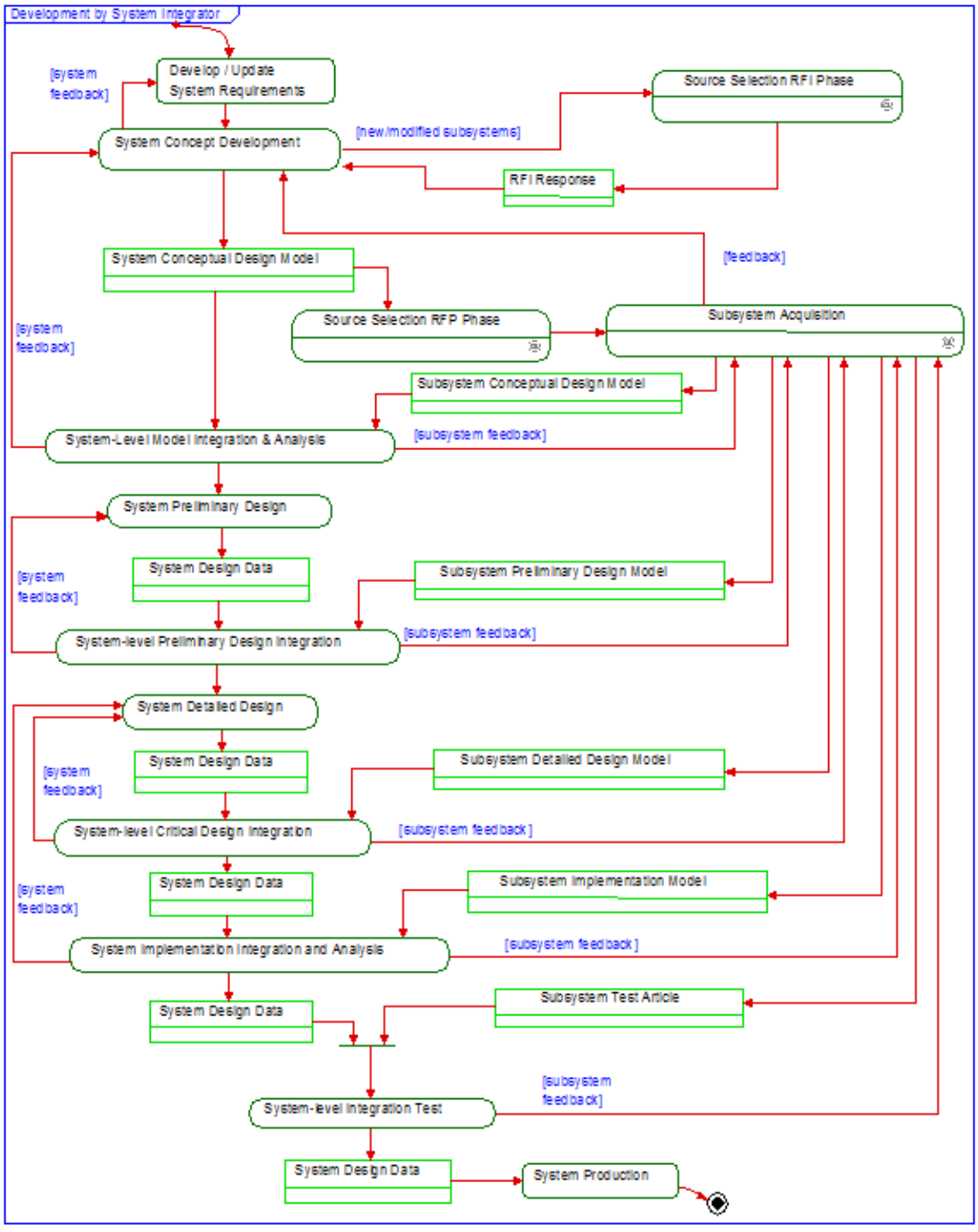

Figure 4. "To-Be" Acquisition Process Model

3.2 Differences between Acquisition Models

The emphasis in this section of the summary document is on the differences between the two acquisition models, but the two preceding activity diagrams illustrate the acquisition processes.

At first glance Figures 3 and 4 look very similar. Notice that “Model” in the “To-Be” process, replaces “data” and “description” in the “As-Is” process in most cases. This seemingly small difference, subtly suggests a complete paradigm shift from passing electronic documents back and forth, to virtually integrating early and often. So, these subtle changes mean that the SAVI paradigm is a significant modification in how such complex systems are acquired. Moreover, this alteration of the acquisition process is likely to be both iterative and evolutionary.

SAVI paradigm changes are expected to occur in overlapping phases. A SAVI infrastructure can be developed through a first phase of developing Model-Based Engineering (MBE) models, modifying related standards, and implementing automated infrastructure tools and processes. A second phase, intended to mature the SAVI product set and SAVI support libraries, is planned to provide a richer set of reusable product models and patterns to support system development. A projected third phase expands SAVI use to additional system life cycle phases (production, maintenance, technical refresh, COTS selection and control for hybrid systems, and product line support) and in domains outside aerospace providing a rich set of building blocks and reusable support methods.

SAVI methodologies are applicable to the entire spectrum of system development scenarios with more advantages and escalating return on investment (RoI) as SAVI matures and the evolutionary paradigm phases are realized:

- “Clean sheet” new developments are likely first beneficiaries of SAVI.
- Hybrid systems with both COTS and Non-COTS components are likely to follow.
- Technical updates of existing systems should benefit from existing SAVI models.
- Product Lines based on existing system models utilize SAVI concepts should further increase RoI.
- Rapid development of systems emphasizing COTS components is a natural fit once MBE-based subsystem models are available.
- Candidate systems with similar components should encourage spin-off projects.

SAVI methodology is based on a fundamental paradigm shift in the development process: simply stated, it is “Integrate, then Build”. It addresses the tendency for problems to remain undetected until integration testing or even the implementation phase of development. SAVI virtual integration should result in reduced program risks, lower cost, and shorter development times. In the SAVI process, system and subsystem integration is continuously evaluated, ensuring integrity and providing continuous visibility into development progress. At each step in the development, models of component parts are analyzed and evaluated for compatibility with the system and with each other. Integration is addressed before implementation.

Each system developed under SAVI produces models calibrated or verified against realized systems and their components, enlarging the number of building blocks for future system development. The concluding section of this document shows that SAVI development and application costs are amortized largely through software cost avoidance within a few (perhaps only one) major project development cycles, though much larger pay-offs should come with continued use.

4 Proof of Concept Demonstrations

The Proof of Concept Modeling and Analysis sub-team was tasked demonstrate a prototype Multi-Aspect Model Repository and Model Bus for this project. The results are summarized in the following sections, though there is more detail and substantiation in documentation produced for participating members (see the appropriate section of the AVSI, Members Only, web site if your organization is a participating member.)

4.1 Requirements for PoC Demonstration Prioritized

PMC members first had to prioritize requirements to be considered in this limited PoC effort. The prioritization choices were made on the basis of nineteen Quality Factors, the selection of which was an AFE task [1]. These Quality Factors, shown in Table 1, are characterized by:

a. being independent of each other;
b. being generally applicable at all levels (system, sub-system, sub-sub-system, etc.) of a development program;
c. ranging from directly quantifiable (like Cost) to more abstract (Difficulty).

Table 1. Quality Factors

Criticality	Post-Processing
Frequency	Cross Coupling
Difficulty	Availability
Cost	Timing
Breadth	Coverage
Effect of Error	Infrastructure
Required Accuracy	Model Fidelity
Typical Accuracy	SAVI PoC Feasibility
Setup Time	Proprietary Content
Duration	

Table 2. Top Ten Prioritized Requirements for the PoC

#	Requirement	Category
1	**Establish Model Bus infrastructure**	Process
2	**Establish Model Repository Infrastructure**	Process
3	**Inform RoI estimates through AFE58 performance and results**	Process
4	**Analyses be conducted across the system**	Analysis
5	**Two or more analyses must be conducted**	Analysis
6	**Analyses 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

As suggested in Figure 2, the Process Flow chart, and based upon the Quality Factors listed above, the top ten requirements (shown in Table 2 above) were next chosen by the PMC sub-group. Of the top 10 requirements identified for the PoC, 3 of them fell into the “Process” category, 5 into the “Analysis” category, and 2 into the “Model” category.

4.2 Prioritization of Analyses Conducted During the PoC Demonstrations

The four analyses which became the highest priority for the PoC demonstrations are network/databus loading, schedulability/scheduling, system latency, functional Integration. The index in Table 3 is a distillation of subjective weights assigned to the various analyses considered and the four with the lowest index were assigned the highest priority for the remainder of the effort. This table only shows the top ten in priority, out of a total of forty-seven analyses considered.

Table 3. Top Ten Highest Priority Analyses

Index	***Analysis***	***Breadth***	***Timing***	***Overall rating***	***SAVI PoC Feasibility***
3	Network Loading	*All_Subsystems*	*NotApplicable*	***1.1***	*Complete*
47	Functional Integration	*All_Subsystems*	*CDR*	***1.1***	*Possible*
13	System latency	*All_Subsystems*	*CDR*	***1.5***	*Complete*
10	Schedulability/Scheduling	*Subsystem*	*CDR*	***1.5***	*Probable*
26	CPU Time	*Subsystem*	*Concept*	***1.5***	*Probable*
27-30	CPU Time	*Subsystem*	*(other)*	***1.8***	*Probable*
14	System Latency	*Subsystem*	*PDR*	***2.2***	*Complete*
34	Percentage CPU Used	*Subsystem*	*Concept*	***2.4***	*Probable*
39	Percentage Memory Used	*Subsystem*	*NotApplicable*	***2.4***	*Complete*
32	Memory	*Subsystem*	*NotApplicable*	***2.5***	*Complete*

Three steps were outlined in the AFE tasking and applied to each of the selected analysis types: (1) essential attributes for each were identified, (2) requirements for the model repository and model bus were spelled out, and (3) essential extensions to Advanced Architectural and Design Language (AADL) [2] (and the associated Open Source AADL Tool Environment [OSATE]) were noted. A single model representative of both military and commercial acquisition processes was developed and placed in the model repository. The model was exercised at three different levels (Tiers) to provide information for evaluating the analysis types against the quality factors chosen. These evaluations led to the following major conclusions: (1) the Tier 1 model can use the AADL system component, as well as the bus component and the bus access connections, to represent an aircraft for systems engineering purposes, (2) Tier 2 satisfactorily specifies a flight guidance subsystem with its own subsystem (the IMA) and allows configuration of the model to exercise both levels as would be necessary for a supplier designing the flight guidance subsystem; and (3) a few Tier 3 components were implemented and connected in this initial demonstration to allow a limited look at components like CPUs and data concentrators. The modeling exercise allowed several different subcontractors to be emulated. Also, for systems engineering purposes, parameters like weight, power, processor budget, network bandwidth, and end-to-end latency analyses were exercised during the demonstrations.

4.3 Contractor (Software Engineering Institute - SEI) Support for the PoC

4.3.1 Statement of Work Tasks

The SEI SoW identified three areas where the contractor would support the PoC effort:

1. The ROI study;
2. PoC modelling and analysis; and
3. PoC tool environment.

The purpose of the RoI study is to provide evidence of the benefits of model-based engineering (MBE) through virtual integration of systems early and throughout the

development life cycle. SEI played a neutral partner role by producing a detailed evaluation of the assumptions made and the sensitivities of the RoI results to these assumptions. This evaluation was a parallel study to the abbreviated assessment of the RoI produced in early June by the RoI sub-team. The SEI study corroborated the RoI team's assumptions and their more detailed assessment also produced a more positive outlook for the benefits from implementing SAVI than the initial look.

SEI support of the PoC modelling and analysis included the following:

1. Participation in choosing example systems;
2. Development of the baseline architectural reference model, to include a June 2008 course for SAVI participants on practical use of AADL and mentoring during the model development;
3. Provision of analysis-specific extensions needed for the baseline reference model;
4. Guidance on how to inject faults into the baseline reference model and how to obtain resolution of the faults, which is the primary value of the demonstration; and
5. Development of a public case study in concert with the AFE 58 members, including development of a demonstration video suitable for illustrating the PoC results at appropriate conferences and for internal showing within participant organizations [3].

The PoC tool environment had to show that an architecture model augmented with information regarding analytical models derived through the model bus and stored in the model repository was feasible. This reference model approach contrasts with independently developed analytical models of the system, such as fault trees, scheduling models, and security models and should lead to earlier and more complete system verification. SEI used OSATE bundled with The Open-Source Toolkit for Critical Systems Development (TOPCASED) to demonstrate the feasibility of using a reference model repository and model bus to perform multi-fidelity and multi-dimensional analysis of system architectures. To support the PoC tool environment, SEI:

1. Specified and documented a reference meta model and its augmentation;
2. Documented an approach to version management control;
3. Provided OSATE as the core tool environment and utilized TOPCASED to demonstrate the model bus and model transformation technology availability;
4. Provided and demonstrated analysis tools to illustrate multi-aspect analysis by providing a resource budget analysis capability, a resource allocation capability, and/or an end-to-end latency capability; and
5. Demonstrated feasibility of analyzing non-performance related properties (like security) and their impact on performance, as well as system properties relevant to system engineers (illustrated through a power consumption analysis).

4.3.2 Packages and Extensions Added to AADL and Additional Analysis

It was not necessary to modify the core AADL language during the SAVI Proof-of-Concept demonstration, nor were any of the Predeclared Property Sets [AADL Standard] changed. Five property sets, having to do with mass properties and with electrical power units, were added to the available ones. A SAVI Common Property Set was also defined to support the modeling effort. To illustrate code generation, the Ravenscar Property Set was also needed. Six OSATE analysis plugins were created or modified to support the SAVI PoC.

It was also demonstrated during the PoC effort that the Groovy scripting language could be used to create, test, and run models directly in the current OSATE environment. Groovy

scripts integrate well with JAVA, making it easy to access the APIs provided in OSATE, and an analysis created in Groovy scripts can be shared in a subversion or CVS repository similar to any AADL project.

After completion of the first iteration of the SAVI PoC demonstration in December 2008, PMC members voted to add Functional Integration to the model between January and April 2009. This demonstration was successfully completed and is illustrated in the demonstration video mentioned above.

4.4 Representative Examples

4.4.1 Model Repository and the Model Bus Requirements to Support PoC Analyses

No significant differences between the military and commercial acquisition processes were identified in task 1 of the AFE, so only one model was developed. [Subsequent to completion of the analyses, questions have arisen regarding interfacing the SAVI PoC AADL models with the Department of Defense Architectural Framework (DoDAF). The PMC is currently considering how best to address these questions.]

4.4.2 Structure of PoC Demonstrations

The POC demonstration is built around an aircraft system integrator (SI) working with subcontractors to develop a system using AADL. Typically, it is a three-phase process. The SI first defines a system architecture and produces initial specifications of subsystems to be developed by subcontractors. Then (sub)system specifications and interface control documents (ICD) are jointly refined. Finally, subcontractors produce their subsystems and iteratively deliver subsystem models to the integrator for virtual integration. The repository structure used for the POC is shown below and for the demonstration the repository was set up on a single server.

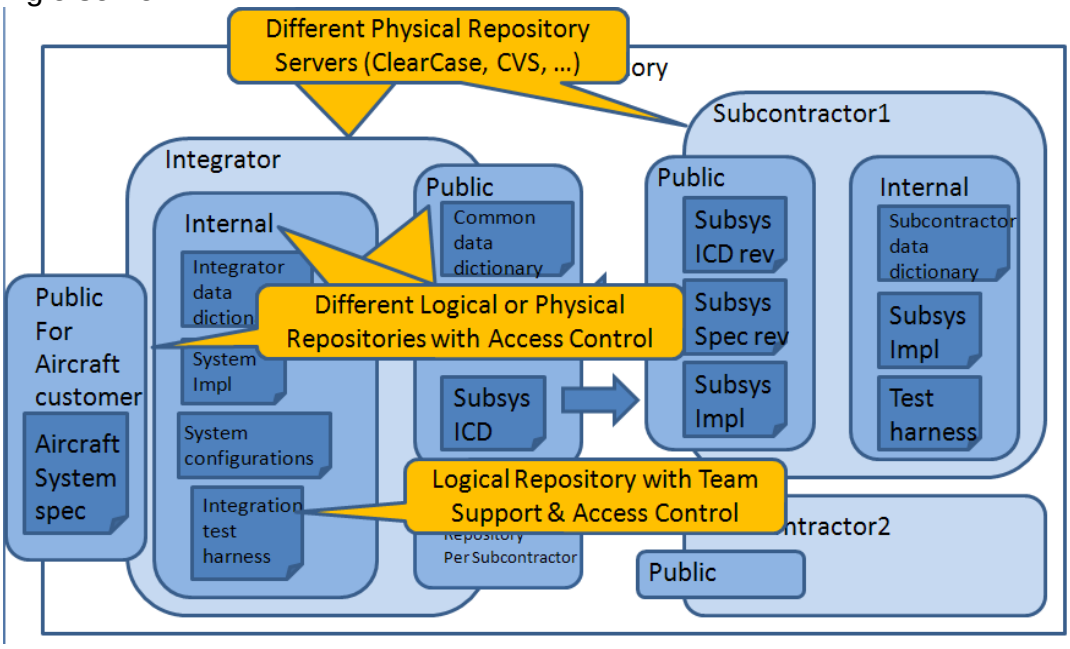

Figure 5. Integrator and Subcontractor Repository Structure

In the POC exercise the SI defines the Tier 1 (upper level) architecture and expands the flight guidance system, an Integrated Modular Avionics (IMA) system. The Tier 1 system supplies resources to the Tier 2 systems (as illustrated in Figure 6) with an AADL model. The bus

access connections represent fuel lines, hydraulic hoses, and other physical sources. The subsystems of the IMA are to be contracted out to subcontractors. Four subcontractors developing four subsystems are postulated. The flight guidance system (a Tier 2 subsystem) contains additional subsystems that can be contracted out as an additional exercise. For each subcontractor up a slightly different scenario was set up to illustrate negotiation of system specification, ICD, local data dictionary, and subcontracting of a black box system.

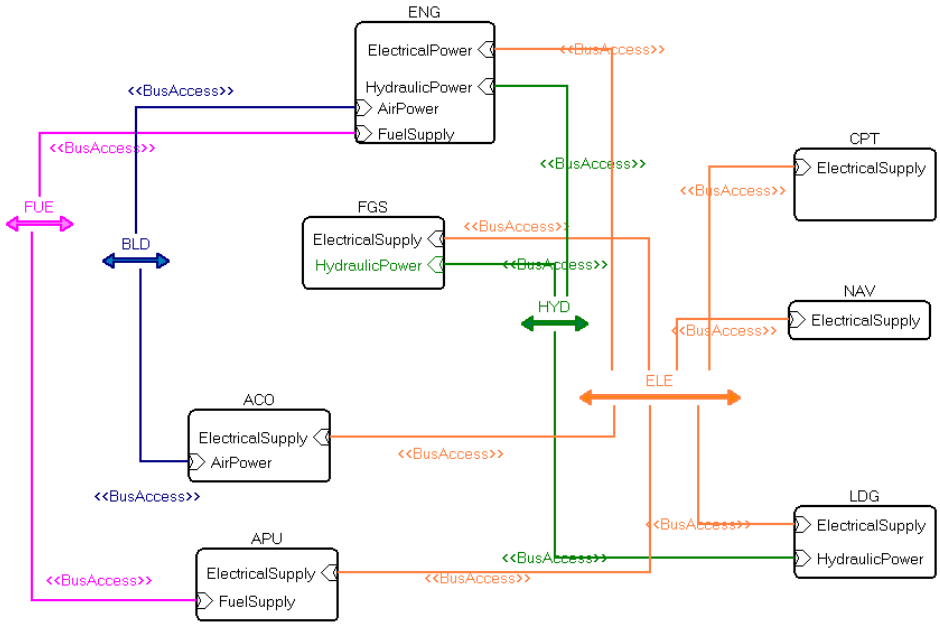

Figure 6. Aircraft System Tier 1 Model with Domain Resources as Bus

Simplifying assumptions made in modelling this system include:

- The entire IMA computing platform is placed inside the Tier 2 Flight Guidance (FG) system. This computing platform could be made accessible to other functionality and managed as a separate Tier 2 system.
- Tier 2 systems may have functionality implemented in software; currently this software is part of the Tier 2 system. Some of its elements could migrate to the IMA computing platform. Here, all software systems are mapped onto the IMA inside the FG system.
- One Tier 3 system in the FG system is an Air Data Computer (ADC). It is currently a black box (represented as an AADL device) and contracted out to a subcontractor. The subcontractor develops an AADL system model for implementation of the ADC.

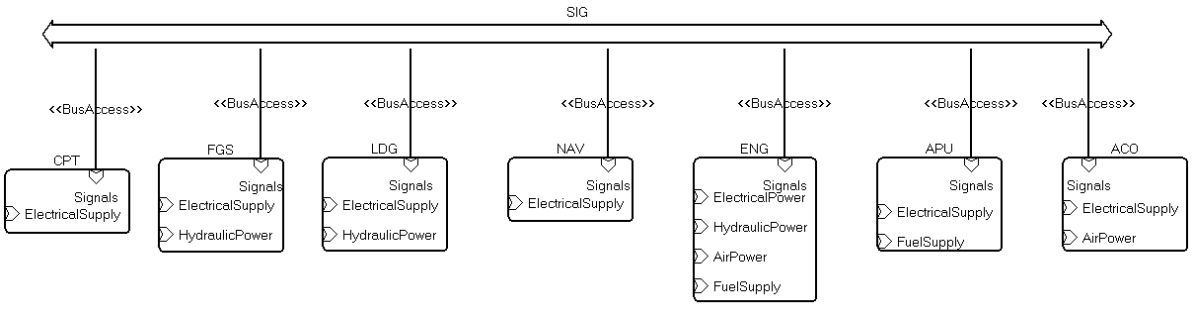

Figure 7. Signal Flow Connections in Tier 1 Aircraft System

All Tier 2 systems are connected to a signal flow bus, as shown in Figure 7, emulating an exchange of information. Figure 7 is a system interconnection representation.

The AADL bus also represents the connectors between the resource and the resource user (Figure 8); in other words, it can represent the hydraulic hoses or the fuel lines.

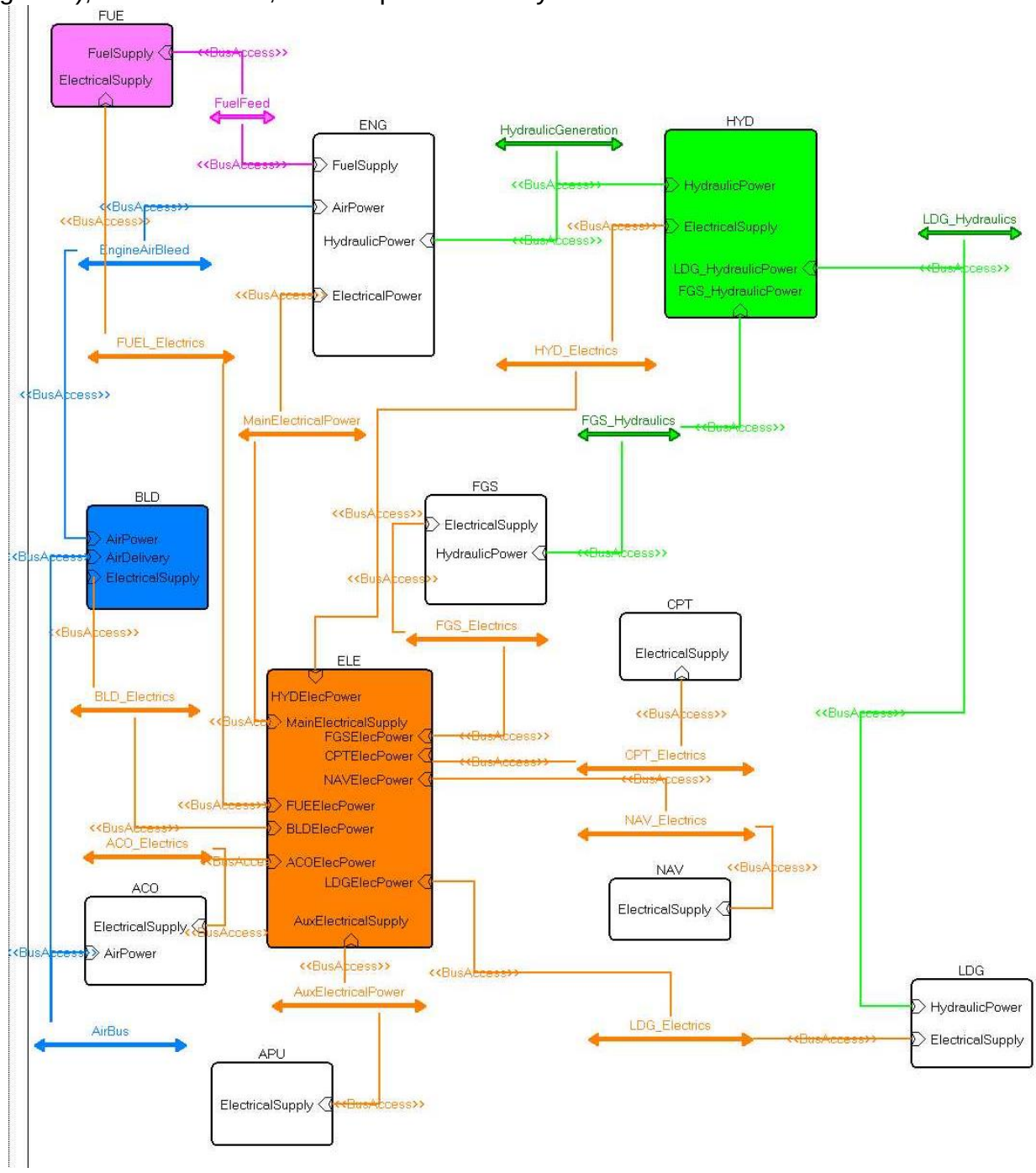

Figure 8. Aircraft System Model with Buses as Domain Resource Connectors

4.4.3 Connection Consistency

AADL is a strongly typed architecture modelling language. A system type or bus type declaration introduces user-defined types of entities in the aircraft domain. For example, a bus

type defines a fuel system type or a fuel line type. The aircraft Tier 2 system specifies that they require access to the fuel system or to a fuel line that connects them to the fuel system by naming the bus type as classifier in the “requires” bus access declaration. A Tier 2 system specification for the Engines system is shown in Figure 9.

```
system Engines
  features
    ElectricalPower: requires bus access ElectricalPower
      { SAVI::PowerSupply => access 20000.0 W;};
    HydraulicPower: requires bus access HydraulicPressure
      { SAVI::PressureCapacity => access 5000.0 psi;};
    AirPower: requires bus access AirFlow;
    FuelSupply: requires bus access FuelFlow
      { SAVI::FuelBudget => access 5000.0 GalpH;};
    Signals: requires bus access SignalFlow;
properties
  SAVI::System_Tier => tier2;
  SEI::NetWeight => 15000.0 kg;
  SAVI::Requirement => "Req 2";
end Engines;
```

Figure 9. Tier 2 Engines System Specification

4.4.4 Subsystem Representation and Analysis

The FG system (or subsystem, depending on the perspective) includes both hardware and software elements. It is useful to look at this system from both a physical and a logical point of view, as shown in Figures 10 and 11, respectively. To analyze the subsystems (or the aircraft system) model, an instance model of the system to be analyzed can be created. The logical view is most useful for this purpose. Of course, the analysis to be performed dictates which and what kind of instance model is most useful. For example, on the FG IMA the sub-team carried out the following analyses: (1) Processor Budget, (2) Memory Budget, (3) Network Bandwidth, (4) Allocated Processor and Memory Budget, (5) End-to-end Latency, and (6) Partition Deployment.

As only one of several analysis outputs, consider the end-to-end latency analysis on an instance model of the FG system. Figure 12 (page 18) illustrates the *Problem View* of Eclipse where errors and information are reported. (There is a filter capability also shown that permits limiting how much information is displayed. In Figure 12 the red circle indicates how the filter is invoked.)

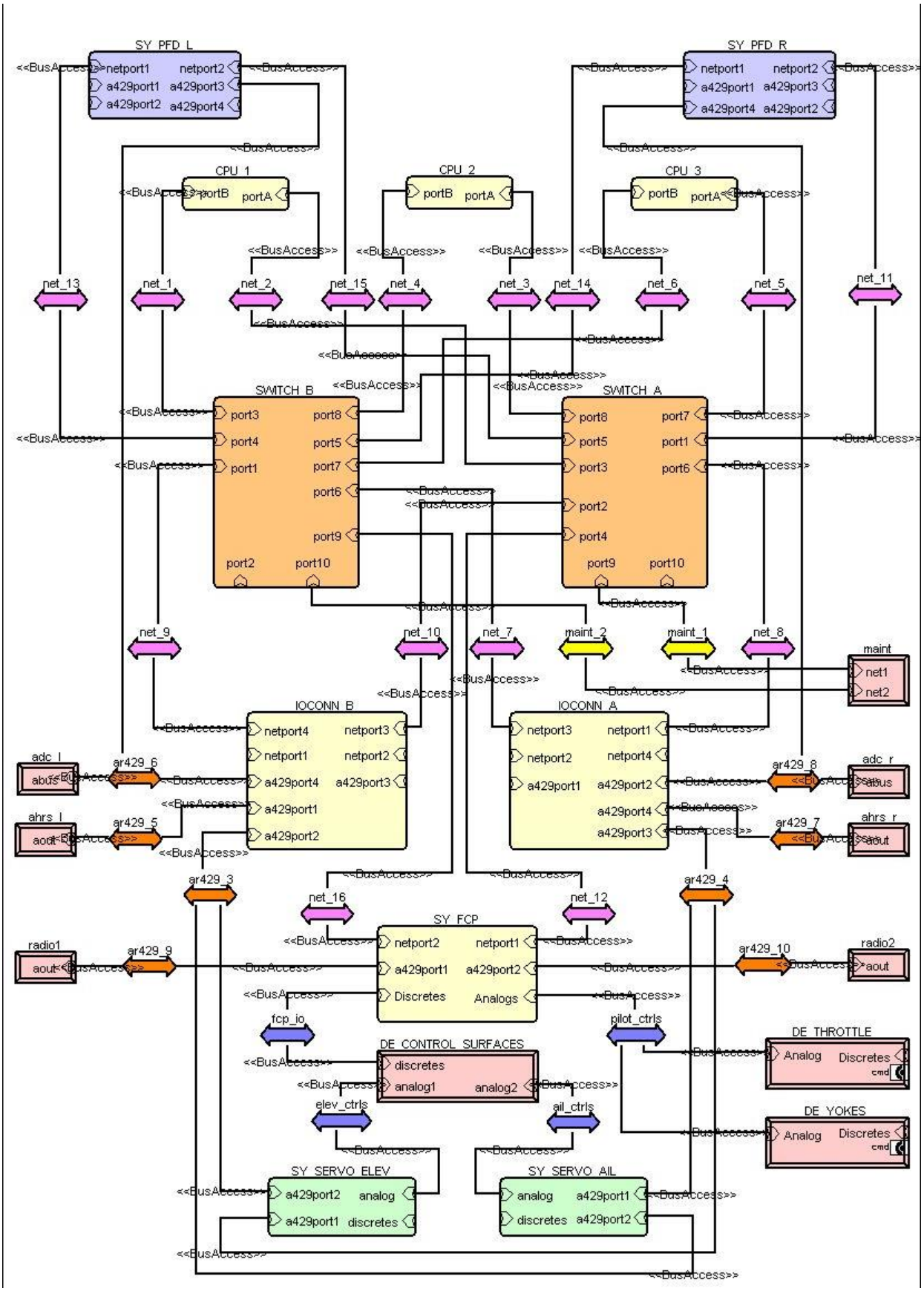

Figure 10. Physical View of Flight Guidance System

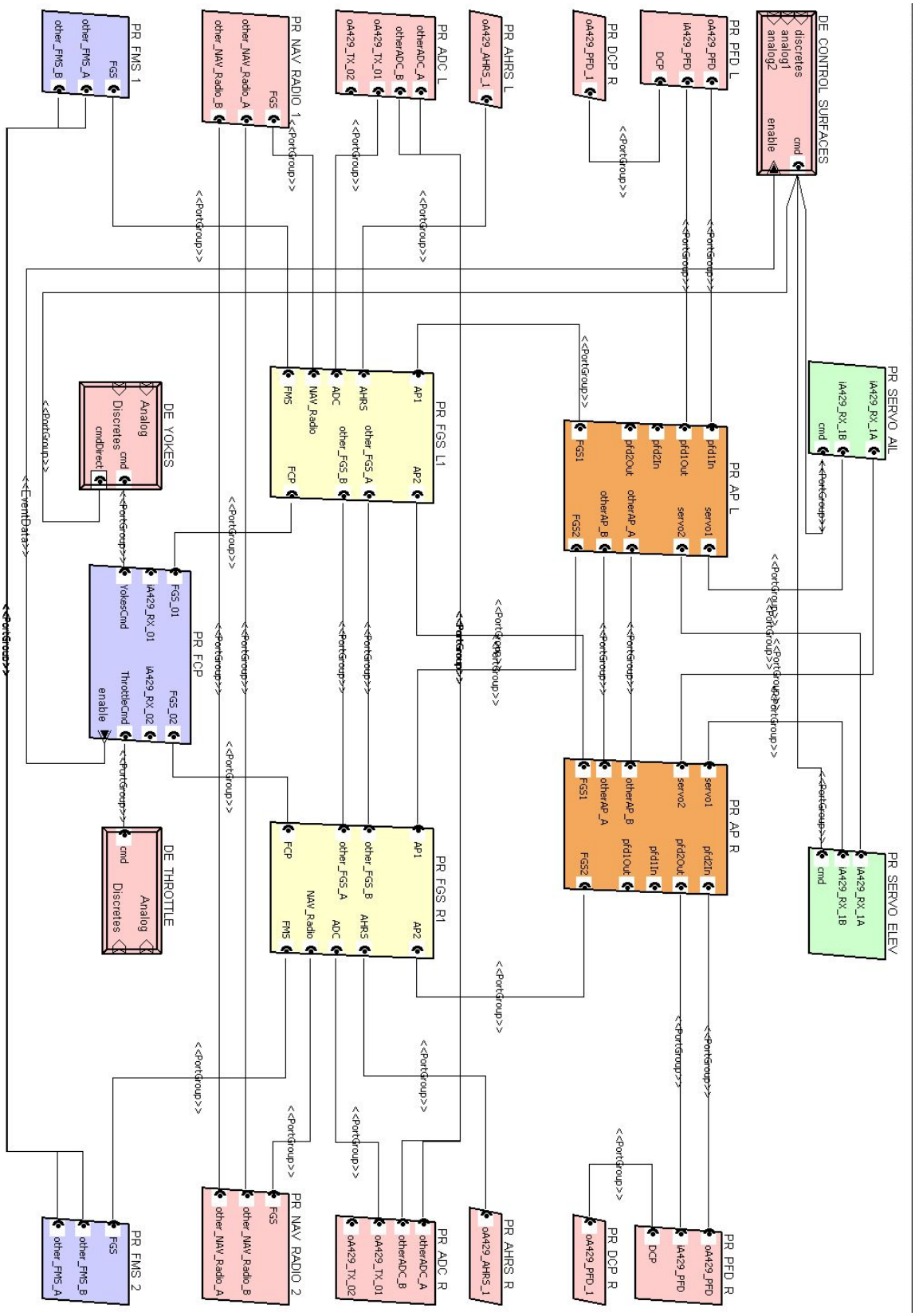

Figure 11. Logical View of Flight Guidance System

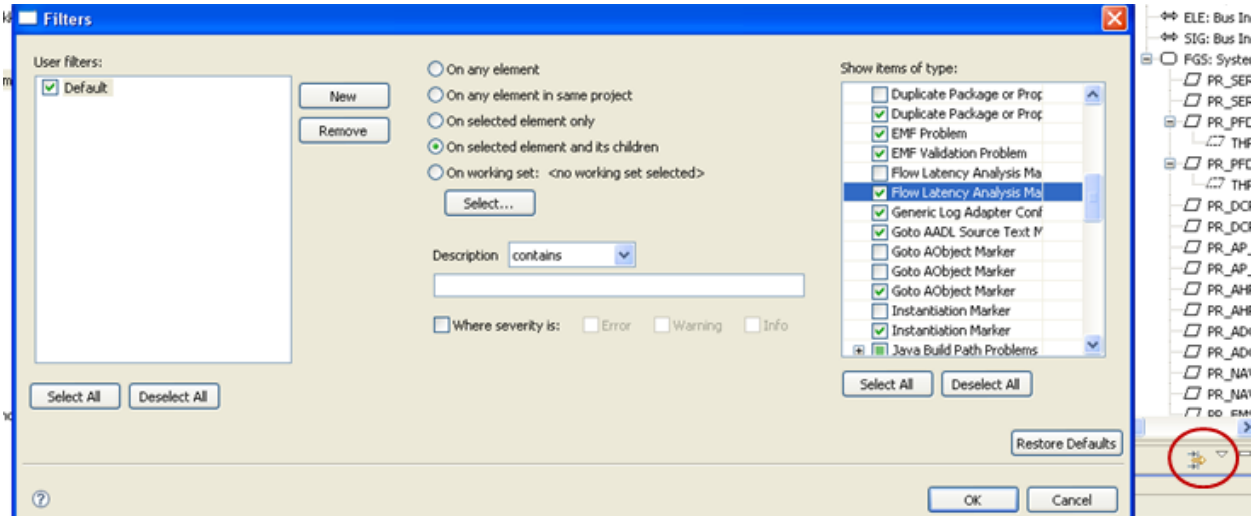

Figure 12. Filtering of Results in Problem View

4.5 Evaluation of Model Demonstrations

Recall that the purpose of this report is to show feasibility of the SAVI concept, not to produce data describing a specific implementation of a system. The preceding paragraphs have hopefully illustrated how the PoC team has created a set of models and modeling tools that allow and facilitate virtual integration. While they have only touched upon the uses of such a methodology, it was clear to the team members that, not only was the SAVI approach feasible, it was relatively easy to implement in this embryonic form. After completion of both iterations and exercising the models as described in the demonstration video, the AFE 58 PMC met to discuss the results. The concluding statement from that face-to-face discussion of the entire project was: The results of the demonstrations indicate that the SAVI concept is sound and should be implemented with further development.

5 SAVI Roadmap

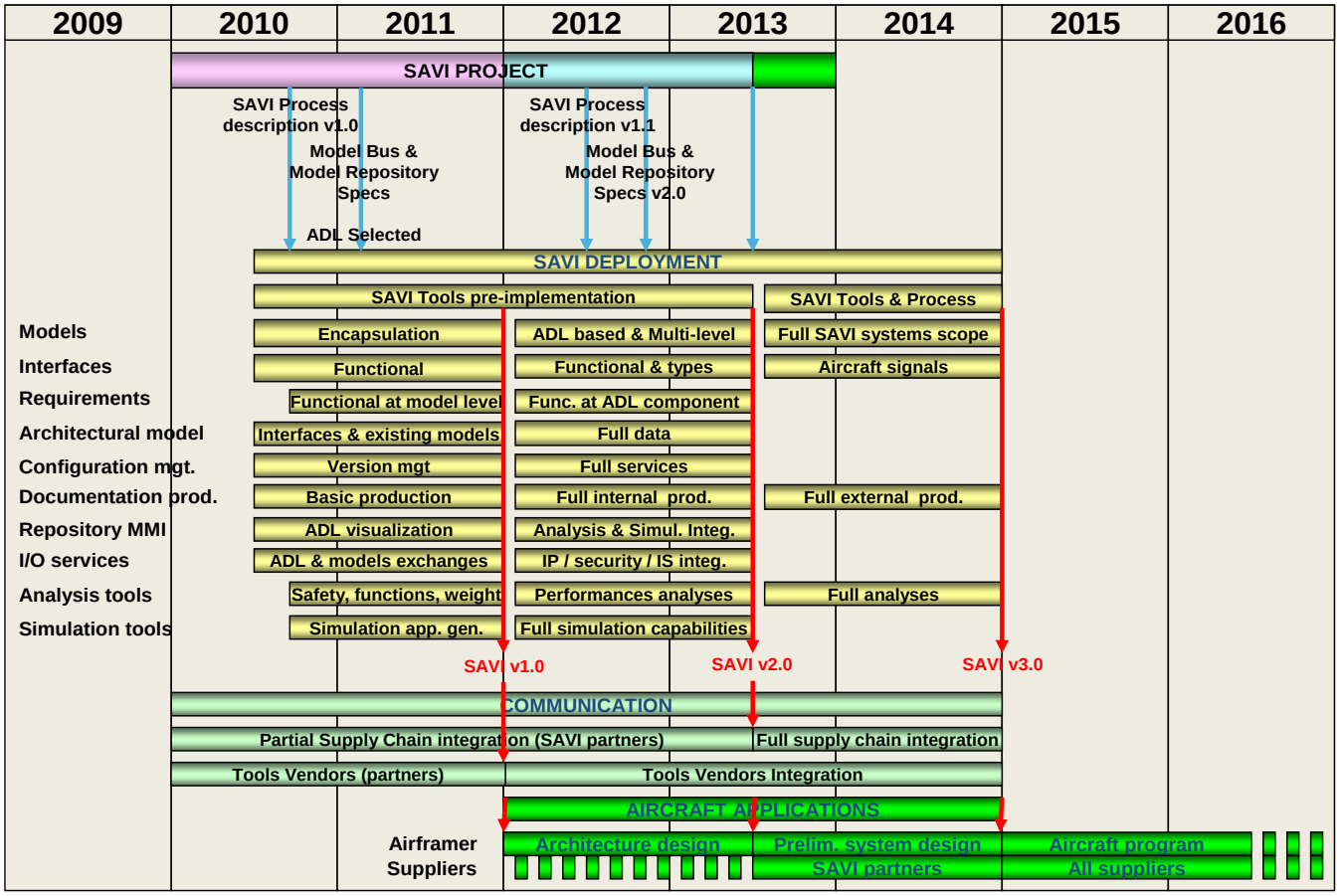

Figure 13. SAVI Project Roadmap

The purpose of this initial SAVI roadmap (completed as a deliverable under AFE 58) is to define how to go from a systems engineering methodology research project (that attained approximately a TRL of 2-3 as the result of the AFE 58 PoC effort) to industrial deployment of the paradigm, taking into account future aircraft programs' development needs as well as short term positive outcomes. Since a new aircraft program is currently not identified that will use the SAVI approach (at least as public information), the proposed strategy is to order developments and outcomes in order to satisfy new aircraft preliminary studies as well as in-service aircraft improvements. These criteria give indications for the remaining SAVI effort, broken down into Phase 1 and Phase 2 needs and priorities. The proposed high level roadmap is illustrated in Figure 13 and is based on the following logic.

- SAVI v1.0 is based on the Phase 1 work and will address mainly the upstream phase of aircraft design, in order to prepare a new aircraft development.
- SAVI v2.0 is based on the Phase 2 and will address the first virtual integration on an aircraft subset, with a partial industrial deployment both on tools and supply chain.
- SAVI v3.0 will be the result of industrialization and communication and will implement the full SAVI paradigm both within the Supply Chain and in tools delivered by tool vendors. (However, it must be emphasized that SAVI is not a software tool suite; it is a paradigm shift. The tools used are not provided by SAVI; SAVI simply allows more efficient use of available tools.) The output for SAVI v3.0 should be at TRL 9.

5.1 Priorities for SAVI v1.0

The priority of SAVI v1.0 is to evaluate the robustness of the concept (recalling that feasibility was demonstrated in the AFE 58 SAVI PoC project), and to deploy it within a first set of industrial partners.

This deployment follows two axes:

- An “aircraft schedule axis”: what phase of an aircraft program SAVI can address in the short term (during development of SAVI v1.0)
- A “scope axis”: how far SAVI capabilities can be deployed as SAVI matures.

5.1.1 Aircraft Schedule Axis

Since new full scale aircraft program requiring exchange of system interfaces between airframe developer and suppliers is not anticipated in the proposed time frame, the intent is to deploy SAVI in the first phase of an aircraft program, involving only one or more System Integrators (SI). This approach will allow implementation and use of SAVI on a real case while maturing aircraft systems architecture. This early deployment (SAVI v1.0) is not restricted from deployment by suppliers; the SAVI approach can be used to study new systems concepts, taking advantage of model-based concepts, without formally exchanging data with the SI. Finally, depending on opportunities, SAVI v1.0 should be deployed for at least one retrofit activity applied to aircraft already in service, with test and analysis data exchanged between SI and suppliers, but without performing a full aircraft Virtual Integration as envisioned for later versions of SAVI.

5.1.2 Scope Axis

Since Phase 1 results should allow defining an aircraft architecture using a model-based approach, and since the main activity is to implement it within each partner organization separately, a reasonable deployment objective is to focus on using SAVI with existing processes and analyses. Making new analyses or implementing some new way of working resulting from the To-Be SAVI Process should only be done on a case by case selection during Phase 1. A simplified Process Breakdown Structure (PBS) is given in Figure 14.

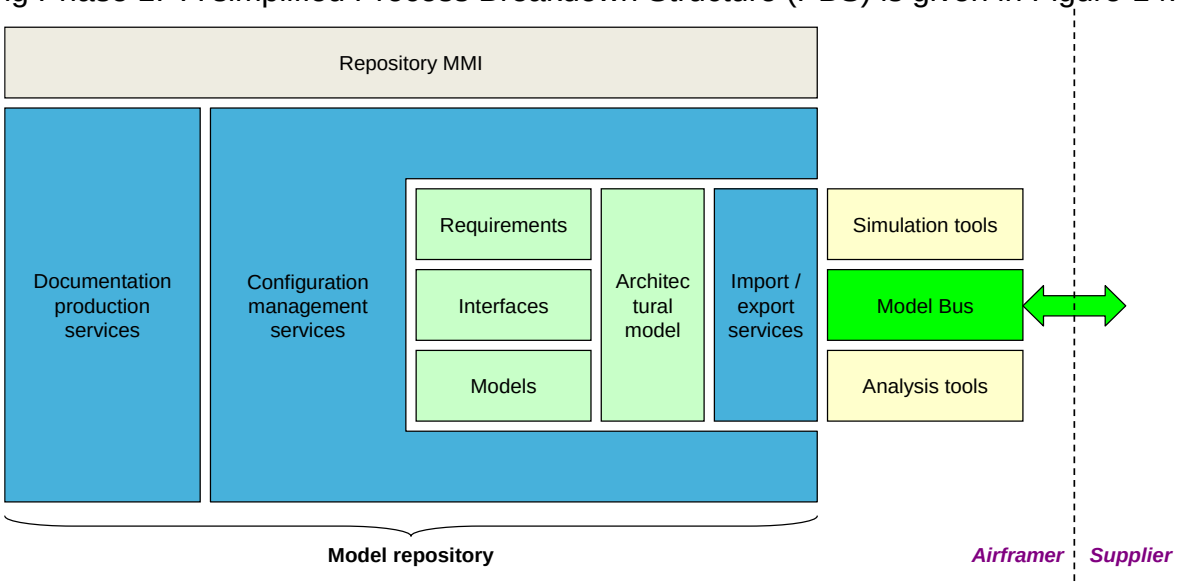

Figure 14. Simplified PBS for SAVI

5.1.3 Deployment Objectives of SAVI v1.0

According to the SAVI PBS, the deployment objectives of SAVI v1.0 are:

- Models – Create a viable set of models – In the current or existing format if they already exist, or new ones created using the SAVI ADL (Architecture Description Language)
- Interfaces – Postulate functional interface descriptions
- Requirements – Demonstrate links between textual requirements and exiting models
- Architectural model – Devise architectural models with interfaces and requirements linkages, based on the ADL, with model encapsulation capability so existing models (not in ADL format) can be utilized within the SAVI process.
- Configuration management services – Develop configuration management modules so version management of an aircraft architecture can be tracked and audited, one that his not limited to the System Integrator's usage.
- Documentation production services – Introduce automated documentation services to begin evolving a capability to automatically produce documentation linking requirements, models, and analysis results.
- Repository MMI – Visualize the ADL model and demonstrate import / export Services.
- Import / export services – Ensure that import / export services allow selection of a sub-part of the architectural model, export in a model bus format, retrieval of a model bus package, and insertion in the architectural model.
- Model Bus – Guarantee the model bus can encode data described in the architectural model.
- Analysis tools – Develop or modify analysis tools to be able to read ADL formatted information from the model bus, including the following priority: safety issues, functional behavior, and weight predictions. The list of viable analysis tools may be expanded depending on the level of tool vendors' participation in this phase of the SAVI project.
- Simulation tools – Be sure simulation tools are able to read ADL formatted information from the model bus and, based on interface definitions and encapsulation of existing models, are able to generate a simulation application from the architecture description.

5.2 Priorities for SAVI v2.0

Since SAVI v1.0 focuses on deployment of the SAVI paradigm at SI (airframer) and supplier levels without fully demonstrating data exchanges and interface compatibility, the primary goal of SAVI v2.0 is to demonstrate the data exchange process for joint early aircraft development phases as illustrated in the Figure 13. The intent is to move all capabilities to at least TRL 7, though TRL 8 is desired, at the end of this phase of work.

5.2.1 Aircraft Schedule Axis

This phase of the development can be performed with selected suppliers active in the SAVI project, on some carefully chosen systems, though only partial Virtual Integration may be achieved in the Pilot or Use Cases available.

5.2.2 Scope Axis

SAVI v2.0 must focus on SI / supplier interactions, that is, on use of the Model Bus and on addressing associated configuration management issues.

5.2.3 Deployment Objectives of SAVI v2.0

Since the aircraft architecture will be defined in SAVI v1.0 largely from the SI point of view, the primary goal of SAVI v2.0 will be to assess functional and performances issues with suppliers' inputs. The efficacy of the SAVI process to exchange information will be demonstrated and developed further from that shown in SAVI v1.0.Nevertheless, interfaces will still be defined at functional (what is the data) and type (what is the bus) levels, since aircraft signals will still not be fully defined.

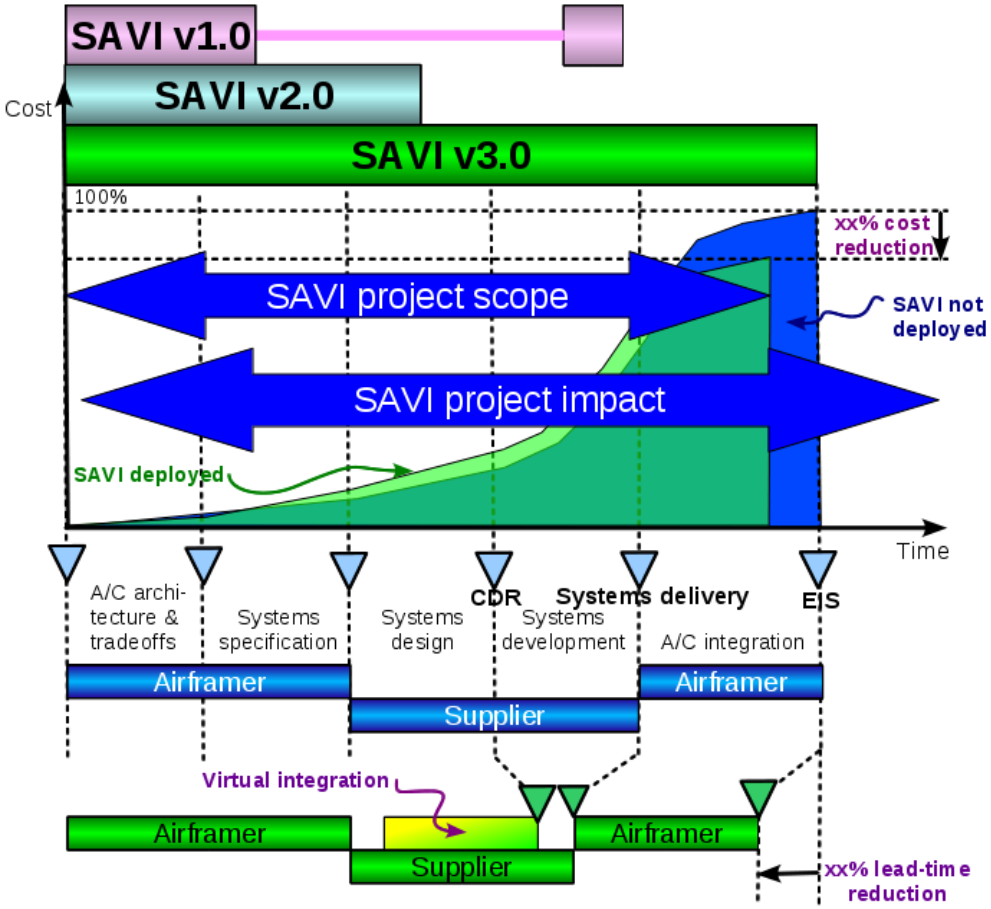

Figure 15. SAVI Deployment Scope Vision

5.3 Priorities for SAVI v3.0

SAVI v3.0 will aim to make the full industrial deployment of SAVI, mainly by enlarging the scope of involved suppliers, in order to perform virtual integration at aircraft level. Success of this deployment hinges on two major aspects that link back to SAVI v1.0 and SAVI v2.0:

- Building communication links with the supply chain on how to apply SAVI processes at least from the beginning of SAVI v2.0.
- Involving tool vendors to adapt tools at least from the end of SAVI v1.0.

This description above and Figure 15 show that SAVI v3.0 deployment will be mostly local evolutions of the SAVI product that introduce new analysis tools, extend interfaces management to manipulate aircraft signal names and payloads, and further refine the capability to generate documentation usable in the certification process. SAVI v1.0 and v2.0 are the keys to success in that end game result.

5.4 Concluding Remarks - SAVI Spiral Development

The SAVI program will continue to follow a spiral approach, operating at two levels:

- **SAVI Internal Spiral -** During the two phases of the project development leading to SAVI v1.0 and v2.0, a spiral approach will be used for each of these phases in order to synchronize and refine activities between work packages, which are defined in planning for future AFEs. These planned work packages continuously collaborate during this spiral evolution.
- **SAVI External Spiral –** This spiral delivers the three SAVI steps for the external community.

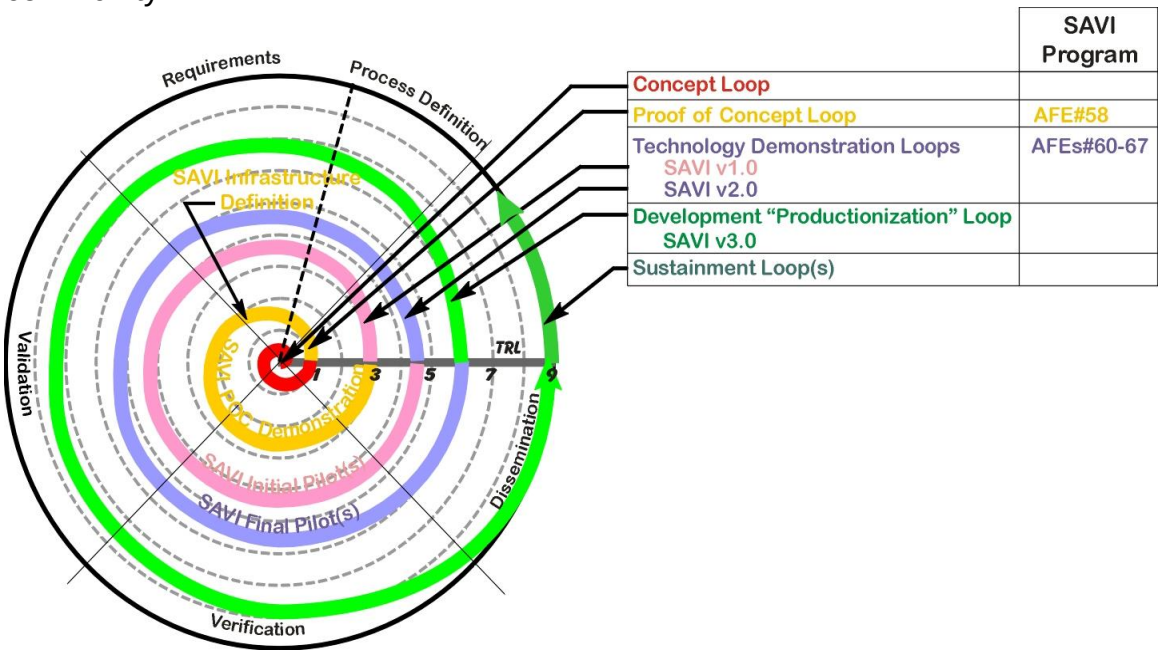

Figure 16. SAVI Spiral Development

Figure 16 graphically summarizes how each of the SAVI versions builds on the assumptions and principles from preceding effort, especially the SAVI Proof-of-Concept work of AFE 58, to drive the Technology Readiness Level from 2-3 to at least 6 (at SAVI v2.0) and then to TRL 9 at SAVI v3.0. Each new version is an iteration refining the SAVI processes, requirements, validation and verification (by application to pilot programs or Use Cases). AFEs 60-67 are planned to provide the overall structure to carry out this iterative development.

Ultimate success in the evolution of the SAVI process depends on four key activities:

1) Developing strong and robust versions of the core modules (Model Bus, Model Repository, and Meta-Model Interfaces) during SAVI v1.0 and demonstrating them within "industrial strength" pilot project(s) or use cases.
2) Garnering enthusiasm for modifying and linking software tools from both the architectural domain and the analysis domains.
3) Securing early and continuous involvement of airworthiness authorities from representative areas of the aerospace industry – civil and military; American, European, and Asian; and all classes of aircraft.
4) Aggressively broadening the base of industrial and governmental support for developing the SAVI process.

6 Return on Investment

The business case underpinning this conceptual evaluation strongly supports continued development of the SAVI Virtual Integration process. The RoI results are summarized in this section of the summary report.

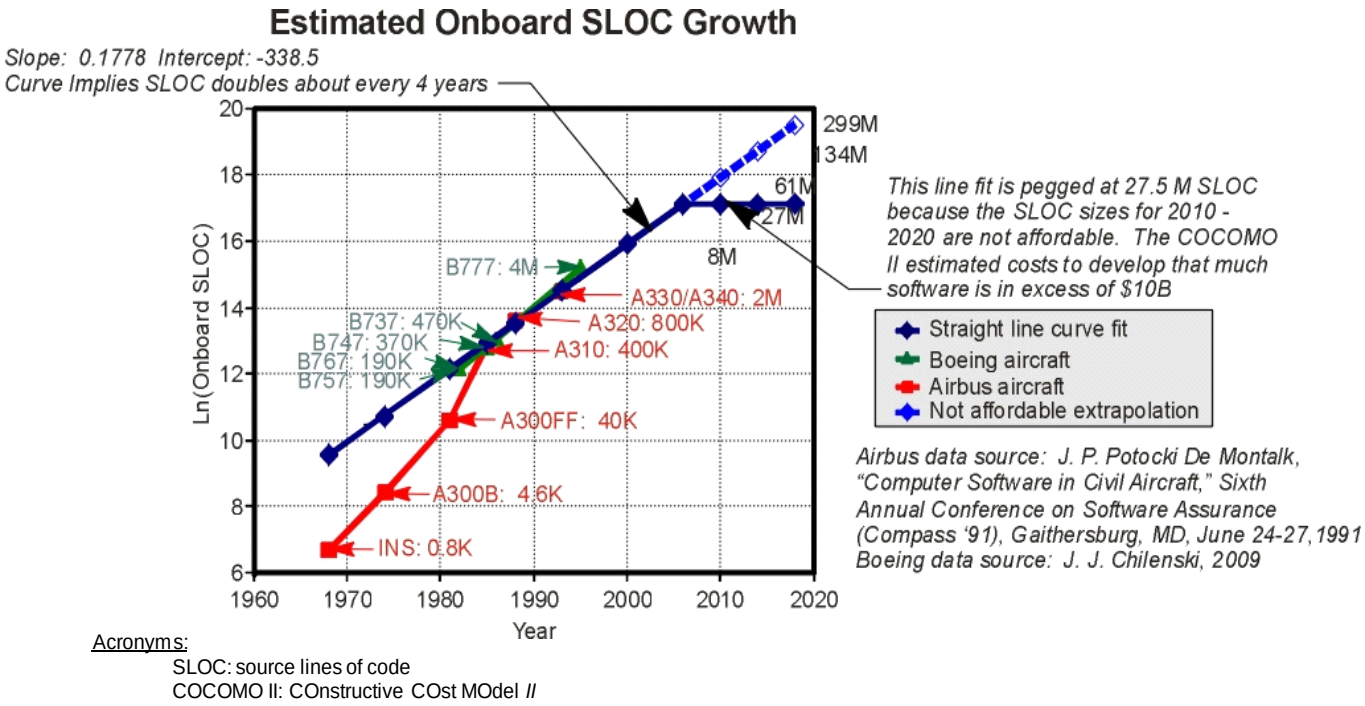

Figure 17. Onboard Software Growth Trends in Commercial Airliners [4]

6.1 Bases and Assumptions for Return on Investment Assessment

The complexity of aerospace systems, which are often dominated by embedded software, is growing at an exponential rate. Airbus and Boeing data show software growth alone tends to double about every four years (Fig. 17, above), making software lines of code (SLOC) a driver in the overall complexity growth. Current generation aircraft software likely exceeds 25 million SLOC, much of it safety-critical. This software is often an integral part of subsystems and has real-time requirements, making such subsystems significantly more costly than conventional, general-purpose, and non-critical software-intensive systems. Their design is intrinsically complex due to the high reliability requirements, special security demands, strong connectivity among subsystems, and extensive safety-critical sensor dependencies. All of these factors affect design, integration, verification/validation activities, and all subsequent operational phases of the system life cycle (SLC). Software development, especially detecting, locating, and removing defects, is one of the more expensive and time-consuming development activities, often causing projects to overrun both budget and schedule, especially when defects are discovered in later phases of product development or even in the operational stage of the SLC. The dominant role played by software in today's complex aerospace systems by no means takes away the important role that hardware and hardware-software integration issues play in system development. But this dominance does point to a way to quantify the value-added when a complex system is developed with the SAVI paradigm.

The majority of software defects are introduced prior to actually writing code during such a development, specifically in requirements and design, but only a fraction of these defects are

detected and addressed early. Currently, about half of all software defects are not found until hardware/software (physical) integration is completed). Also the nominal cost of removing software defects increases as systems grow in size, connectivity, and complexity and when these defects go undetected until physical integration. So, the nominal cost of managing defects propagating through the development life-cycle spirals upward, a problem well understood by system integrators. Two observations illustrate the severity. The nominal cost for removing a defect introduced in pre-coding phases but detected in post-coding phases is, for safety-critical systems, often two orders of magnitude higher relative to the cost of removing it prior to code development. Seventy-nine percent and sixteen percent of the rework cost [5], [6], [7] is due to defects in the requirements and design phases, respectively. Rework cost normally is 30%-60% [6], [7], [8] of total development cost, and large systems and quality attribute-intensive systems trend toward the higher values.

6.1.1 Methodology Used in the AFE 58 Return on Investment Analysis

This section describes the AFE 58 approach to quantifying economic effects on the development of software-intensive systems for aircraft when deploying SAVI. The basis for comparison is the existing development process. A return of investment (ROI) analysis, incorporating conservative assumptions and using net present value (NPV) estimating techniques, is underpins the business case for implementing SAVI. Reuse of software, with percentages of reuse that are conservative, was assumed for this estimate. The rationale for estimating these and other parameters conservatively is simple – an analysis under these assumptions that produces a positive ROI represents a lower bound on the ROI.

The ROI analysis is still evolving. During AFE 58 a simple, conservative framework for calculating the ROI was built. It has since been developed with more detailed assumptions, but the report on that evolution is being finalized [9]. In every case but one, the simple model built during AFE 58 erred on the side of constructing a more conservative calculation. In that one exception, the most likely output from COCOMO II [10] is used instead of the upper bound of the COCOMO II estimate. By itself, that assumption is not unreasonable, but the more detailed analysis corrects to a better lower bound on the ROI. However, using the most likely output from COCOMO II instead of the upper bound is more than offset by the other changes due to the more detailed assumptions, and the resulting ROI increases. Of course, the ROI analysis will continue to evolve during AFE 59, the next phase of SAVI development. Since the more detailed analysis is still being refined, this report deals only with the range of results from the earlier analysis.

The approach is described below:

- First, estimate the future size and complexity of avionics software systems by extrapolating data (SLOC) from previously built aircraft. These predictions (Fig. 17, adapted from reference [4] and including additional data from Boeing) have been checked for validity with AVSI system integrators. Two scenarios based on previous generation aircraft systems pointed to a future system containing approximately 60 million SLOC. That level of software growth is self-limiting, given that the cost of such a complex system is in excess of $10B, likely exceeding a limit of affordability.
- COCOMO II, [10] a popular tool for estimating system development costs for software-intensive systems, was selected. Developed and continuously improved at the University of Southern California [11] and used by the U. S. Army, this tool estimates total cost for developing representative systems.

- The system of interest consists of a single kind of subsystem that does not differentiate among safety-critical, highly critical, and not critical subsystems. The more detailed analysis will model the distribution among these three types, and will assume the total code base is distributed among them at rates of 30%, 30%, and 40%, respectively. These assumed proportions allow differentiation of the cost for different types of subsystems with respect to their requirements. This distinction carries through the estimation by setting switches in COCOMO II reflecting aerospace industry practices (reliability, nature of development, documentation needs, constraints on execution and storage, product complexity, and distributed development and supply chain).
- Each subsystem is realized through both reuse of existing code and new code development. The proportions of new code developed for each analyzed subsystem is 30%, 40%, and 50% (varied with each type of subsystem) of the total code base.
- A nominal labor rate of $22,800/month, based on a nominal $150/month labor rate in 2006, was input to COCOMO II for the initial estimates. The later, more detailed analysis [9] adjusts the labor rate for inflation at an average rate of 4% annually, as an input to COCOMO II, since the cost of software development on an aerospace program is usually spread over several years.

- Using the predicted “as-is” costs as a base, the cost reduction, ROI, and NPV for the “to-be” (after deployment of SAVI) scenario, the approach assumes:
  - COCOMO outputs three total cost estimates: optimistic, most likely and pessimistic ones. The most likely estimate from COCOMO II is of primary interest in this report. In the more detailed analysis [9], the optimistic (more costly) estimates are favoured for computing ROI and NPV since they give a more conservative (lower RoI) prediction bound.
  - To emphasize the effects of rework cost, the relative amount of time and cost to manage defects introduced in various phases is estimated. This estimate is based on empirical data from case studies and observations showing the number of defects typically introduced for system of a defined size and complexity, and the nominal cost of removing defects when detected in the SLC.
  - Three scenarios of rework cost are considered: 30%, 40%, or 50% of total development cost is considered rework cost due to defects and errors and due to requirements changes (30% is low, based on documented evidence for aerospace systems) [6], [7], [8]. Since defects drive rework cost, detecting and removing a defect earlier in the SLC lowers cost of rework. Cost reduction is estimated for all three defect removal rates.
  - ROI and NPV are based on an estimated cost of $86 million to realize SAVI [9], [10]. Results and details of the calculations outlined above are available [9] to fully understand the methodology and how various parameters drive results. The RoI subgroup's goal was to allow any organization, using its own unique assumptions and conditions, to credibly generate similar estimates. The following flow chart (Figure 18) summarizes the RoI estimation process.

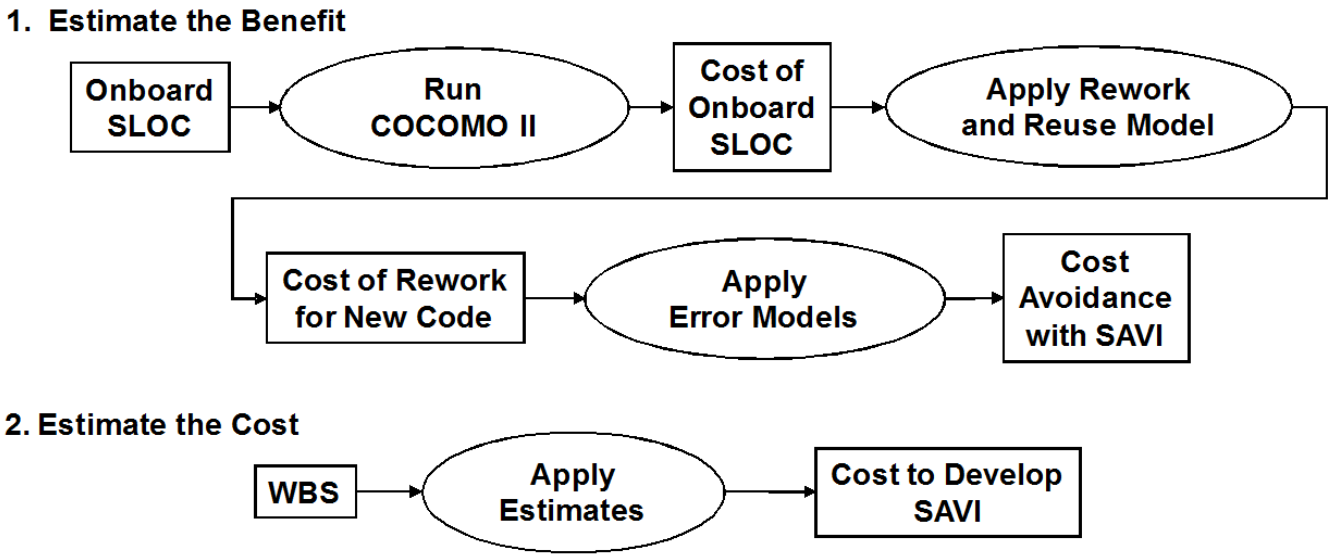

Figure 18. RoI Flow Chart [12]

6.1.2 Estimating the Cost of Implementing SAVI

Prior to initiating AFE 58 a top down estimate of the resources needed to bring SAVI concepts to a usable form (TRL of 9) was $50 million. However, after some initial break downs of the work to be done on the Virtual Integration process [and specifically the Model Bus and Model Repository (MBMR) concepts] during AFE 58, this cost estimate was revised with another top down look [10]. Table 4 summarizes the results of that second iteration cost estimate.

Table 4. Top-Down Estimate of Cost to Develop SAVI Concept [12]

	2006	2010	2011	2012	2013
Labor Rate (per hour)	$150.00	$175.48	$182.50	$ 189.80	$ 197.39
Cost		($26.1 M)	($27.7 M)	($28.3 M)	($26.1 M)
Total Cost					**($108.1 M)**
Discount Rate					10%
NPV					**($85.7 M)**

6.2 Summary of Findings

Based on the assumptions detailed above, analysis produced the following observations:

- Rework cost is dominated by the cost of removing defects generated during the requirements phase and the design phase but not detected until later in the SLC life. Analyses [5], [6], [7] show that these defects amount to 79% (defects traced to requirements) and 16% (defects traced to design) of rework cost.
- There are two reasons for this dominance. Only a small fraction of such defects (less than 10%) are detected in these early phases under the “as-is” development paradigm. Thus, defects leak through to successive development phases and are detected late,

most often in the integration phase, where the cost is one to two orders of magnitude higher to manage the defect once detected.

Table 5 summarizes the key conclusions from the RoI estimates done during AFE 58 (before the more detailed analysis being completed in Reference 9). In using this table, the conservative nature of the assumptions must remain firmly in mind. Notice that a simple multiplier is used to estimate hardware effects in both RoI evaluations.

Table 5. Simple RoI Results Summarized [12]

	NPV (Cost Avoidance)	Multiplier to include Hardware	Total Cost Avoidance	NPV (Cost to Develop)	ROI % per year
Pessimistic	$64 M	1.55	$99 M	($85.7 M)	**2%**
Expected	$256 M	1.55	$398 M	($85.7 M)	**40%**
Optimistic	$768 M	1.55	$1.193 B	($85.7 M)	**144%**