Architecture-led Requirements and Safety Analysis of an Aircraft Survivability Situational Awareness System

Dr. Peter Feiler, Software Engineering Institute, Carnegie Mellon Universityphf@sei.cmu.edu

Software cost in mission and safety-critical systems has been escalating exponentially due to high requirement error leakage into system integration. Furthermore, system tests are designed against a large percentage of ambiguous, missing, and incomplete requirements. The Architecture Centric Virtual Integration Process (ACVIP) is being investigated by the US Army to address these challenges. It is an adaptation of the System Architecture Virtual System Integration (SAVI) approach based on the SAE Architecture Analysis & Design Language (AADL). It is a model-based approach to detect and remove defects through virtual system integration and analysis. In this paper we describe an architecture-led approach to specification of verifiable requirements and to system safety analysis to improve the quality of requriemnts as well as sasfety hazards and their mitigation through derived requirements for a sasfety system. A primary objective of this approach is to achieve improved coverage of requiremetns and safety hazards.

INTRODUCTION

The Software Engineering Institute® (SEI) performed an architecture-led requirement specification and safety analysis in a shadow project of the United States Army Aviation Development Directorate (ADD) on the Joint Multi Role (JMR) Technology Demonstrator effort’s Joint Common Architecture Demonstration (JCA Demo) Project (Ref. 1) to investigate and mature the Architecture Centric Virtual Integration Process (ACVIP). ACVIP is a DoD process fashioned after System Architecture Virtual Integration (SAVI) (Ref. 2) performed by a consortium of aerospace organizations. Like SAVI, the purpose of the ACVIP is to address the affordability and associated risks of developing complex software intensive systems through early virtual integration and analysis before implementation.

Achitecture-led requirement specification (ALRS) addresses the problem of a high percentage of iambiguous, mssing, and incomplete requirements found in textual requirement documents that result in costly rework later in development. It improves the quality of requirements by assuring better coverage of requirements along two dimensions. (ALSA) Architecture-led safety analysis assures improved coverage of safety hazards through a fault propagation ontology and allows for automation of currently labor-intensive best safety analysis practices, e.g., SAE ARP4761.

ARCHITECTURE LED REQUIREMENTS SPECIFICATION (ALRS)

The Architecture Led Requirements Specification (ALRS) process utilizes the AADL ALRS adapts the eleven step process outlined in the Federal Aviation Administration (FAA) Requirements Engineering Management Handbook (Ref. 3). ALRS adapts the CPRET (Ref.4) representation of a system defined by the *Association Française d'Ingénierie Système* which is shown graphically in Figure 3.

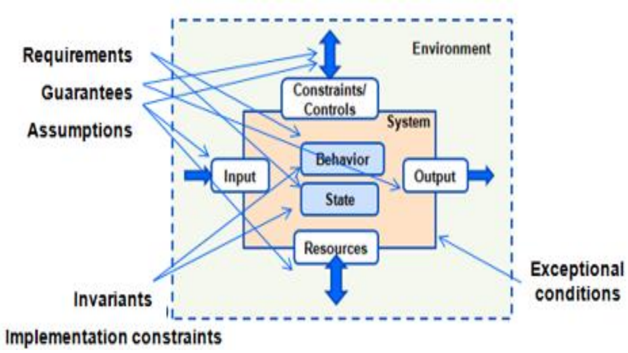


Figure 3- Elements of a System Specification

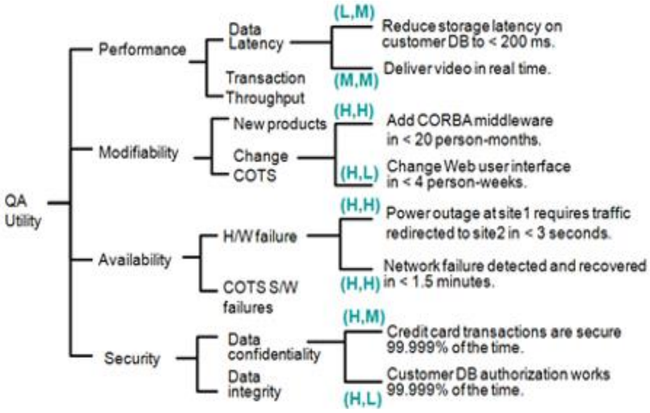
In the ALRS process a user models a system in its operational context as an AADL model of interacting systems. An explicit model of these interacting systems guides the user to specifying requirmenets regarding each of these system interactions in terms of input assumptions, output guarantees, invariants on system state and behavior, as well as asusmptions about resoruces being utilized, and interactions with supervisory capabilities.

When used in the context of an existing requirement document, users map the requirments to an AADL model. This mapping helps the user to quickly identify any gaps in the set of requrirements. It also lets the user see whether a requirement section cover one or more system components.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.
This material has been approved for public release and unlimited distribution.
Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM-0002390

ALRS utilizes utility trees from a Quality Attribute Workshop (QAW) (Ref. 5) or an Architecture Tradeoff Analysis Method (ATAM) (Ref. 6) to provide a framework for achieving coverage of non-functional properties, also known as operational quality attributes. Prioritization of the utility tree leafs driven by mission goals help the user ensure that critical requirements are well-specified. Such a utility tree is shown in Figure 4.

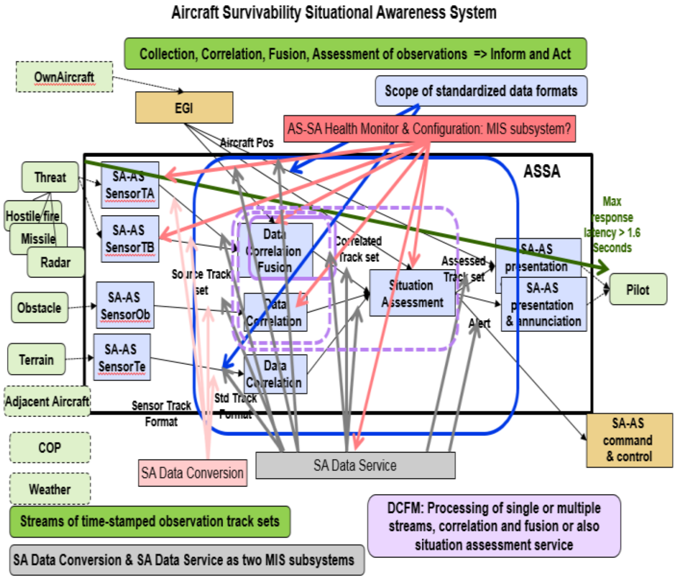


QA Utility
Performance
Data Latency
Transaction Throughput
(L,M)
Reduce storage latency on customer DB to < 200 ms.
(M,M)
Deliver video in real time.
Modifiability
New products
Change COTS
(H,H)
Add CORBA middleware in < 20 person-months.
(H,L)
Change Web user interface in < 4 person-weeks.
Availability
H/W failure
COTS S/W failures
(H,H)
Power outage at site1 requires traffic redirected to site2 in < 3 seconds.
(H,H)
Network failure detected and recovered in < 1.5 minutes.
Security
Data confidentiality
Data integrity
(H,M)
Credit card transactions are secure 99.999% of the time.
(H,L)
Customer DB authorization works 99.999% of the time.

Figure 4: Quality Attribute Utility Tree

Early in the development process the SEI team captured requirement information from the JCA Demo BAA and Stakeholder and Systems Requirements documents of the aircraft survivability situational awareness (ASSA) system as well as UML models made available to suppliers of a data correlation and fusion system. This analysis identified shortcomings in the system-level and component-level requirements. They included inconsistencies, and missing requirement information in the original documents, as well as defects related to safety, latency, and timing / resource utilization. This was achieved by modeling the system in its operational context as well as the functional and the system architecture of the ASSA itself. The resultant architecture model was generalized into an aircraft survivability situational awareness (ASSA) system, creating a reusable reference architecture for the domain of use.

This ASSA system incorporates the MIS and the DCFM, both of which provide several functional services. Figure 5 shows the functional architecture of ASSA with a clear deliniations of its interface with the operational environment. In addition it shows three infrastructure services. Two services are provided in a layer below the situational awareness system, i.e., the data conversion service, and the data management service. The third service, a health monitor, resides in a layer above the situational awareness system to detect and report any exceptional conditions in the operation.



Aircraft Survivability Situational Awareness System
Collection, Correlation, Fusion, Assessment of observations => Inform and Act
OwnAircraft
EGI
Scope of standardized data formats
AS-SA Health Monitor & Configuration: MIS subsystem?
Aircraft Pos
ASSA
Threat
Hostile fire
Missile
Radar
Obstacle
Terrain
Adjacent Aircraft
COP
Weather
SA-AS SensorTA
SA-AS SensorTB
SA-AS SensorOb
SA-AS SensorTe
Source Track set
Data Correlation Fusion
Data Correlation
Data Correlation
Correlated Track set
Situation Assessment
Assessed Track set
Alert
SA-AS presentation
SA-AS presentation & annunciation
Max response latency > 1.6 Seconds
Pilot
SA-AS command & control
Sensor Track Format
Std Track Format
SA Data Conversion
SA Data Service
DCFM: Processing of single or multiple streams, correlation and fusion or also situation assessment service
Streams of time-stamped observation track sets
SA Data Conversion & SA Data Service as two MIS subsystems

Figure 5 – Layered Architecture of ASSA System

The resultant functional architecture also became the basis for quantitative analysis of the ASSA early in development, e.g., pre-PDR. As Figure 5 shows, the model included end- to-end flow specifications of a critical flow to represent response time requirements. It also captures a UML sequence diagram from the original documentation as an analyzable interaction protocol across ARINC653 partitions. The latency analysis capability of OSATE2 informed us of the latency overhead contributed by this protocol, and its effect on the critical flow, i.e., that in the best circumstances the requirement can barely be met.

ARCHITECTURE LED SAFETY ANALYSIS (ALSA)

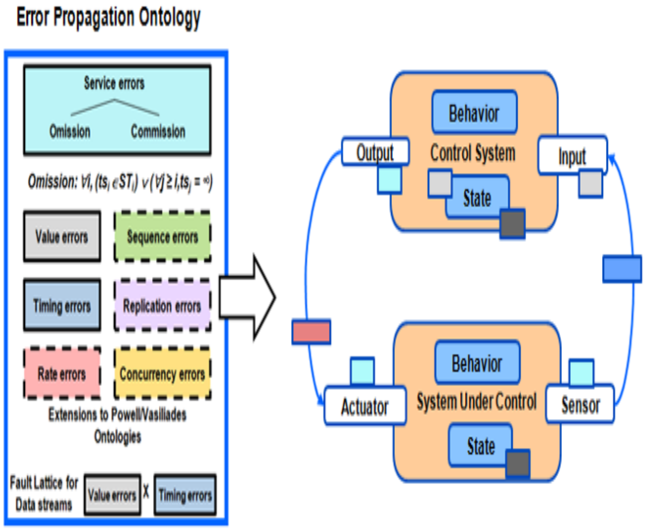


Figure 6- Identification of Hazard Sources and Impact

The ALSA process builds on the AADL created for the ASSA during the reuirement specification process. The user annotates an AADL model with fault information utilizing an error propagation ontology as illustrated graphically in Figure 6. The error propagation ontology addresses issues of service omission, commission, value, timing, rate, sequence, replication, concurrency, authorization, and authentication errors. Users can adapt this ontology to commonly used hazard guide words, such as loss of power. The propagation paths between system components are derived from the architecture specification itself.

This process leverages the AADL Error Model Version 2 (EMV2) Annex (Ref. 7) to support SAE ARP-4761 (Ref. 8) best system safety analysis practices, such as an FHA, FMEA and FTA. The analysis models, such as a fault tree, are generated from the annotated AADL model, and then processed by a FTA tool. In the case of FHA and FMEA the respective reports are generated directly from the annotated AADL model – as shown in Figure 7. In the SAVI initiative the SEI recently demonstrated how the SAE ARP-4761 process can be supported by an AADL model annotated with fault information using the Error Model Annex standard for AADL on an aircraft wheel braking system. FHA, FMEA, and FTA reports as well reliability/availability analysis reports have been generated from safety analysis performed with such a model.

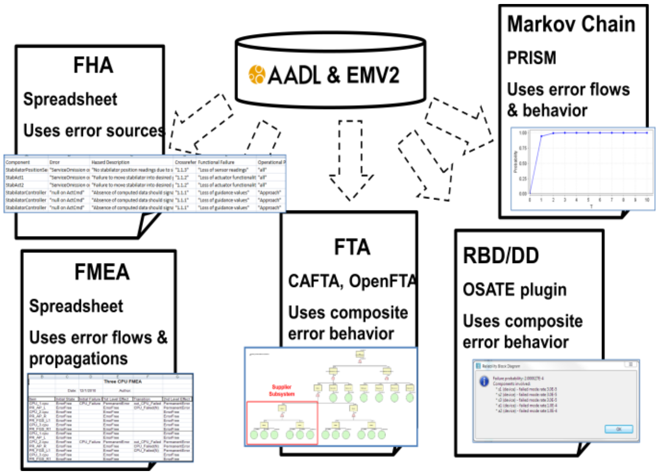

AADL & EMV2
FHA
Spreadsheet
Uses error sources
Markov Chain
PRISM
Uses error flows & behavior
FMEA
Spreadsheet
Uses error flows & propagations
FTA
CAFTA, OpenFTA
Uses composite error behavior
RBD/DD
OSATE plugin
Uses composite error behavior

Figure 7- Safety Analyses from Annotated AADL Models

An Architecture-Led Safety Analysis (ALSA) was conducted on the ASSA. In the common safety analysis practice ASSA was assigned a design assurance level E with respect to flight worthiness. However, since aircraft does get lost due to enemy threats, obstacles, and terrain variation, we considered it a critical component that get the attention of a safety analysis. In addition to complete failure of providing the ASSA service, the hazards considered included providing false information such as false positives in the form of alerting the pilot of threats and obstacles that do not exist, false negatives such as not alerting the pilot when these threats and obstacles exist. In addition the timeliness of information was taken into account, i.e., how much information delay is acceptable to the pilot. Subsequent to citing the hazards, the potential error sources were systematically identified that can propagate as one of the identified hazard categories to the pilot. A fault ontology provided as part of the AADL Standard Error Model annex was used as a checklist of fault propagation categories to consider in the process.

The insights from this analysis lead to a set of derived safety requirements for the health monitoring system that were lacking in the original System Requirement document. The primary focus of the health monitoring system was on detection and reporting, i.e. it is responsible for recognizing when one of the identified hazard conditions occurs and then informs the pilot to that effect.

JCA DEMO ACVIP ANALYSIS FINDINGS AND LESSONS LEARNED

Previous studies have shown that peer review is a very cost-effective means of defect detection, partly because it was the only traditional method that could be applied in early development phases. The ACVIP researcher's experience is that many defects were detected during model development even before analysis tools were applied. This is achieved by mapping terms in the document into concepts expressed by AADL. Users quickly realize different terms used in different sections of the documents for the same concepts, and conflicting statements about specific attributes of model elements, e.g., two different numbers for range of operation. Strong typing in AADL ensures that interactions between virtually integrated system components are consistent, e.g., that measurement units and interchange protocols are used consistently. In other words, the rigor of the AADL focuses attention on ambiguous and incomplete elements of a natural language document and eliminates potential system integration problems early in the process. This is consistent with earlier reports that a significant benefit of modeling is more precise specification; many defects are found during the model development phase (Ref. 9).

Earlier studies showed that providing reviewers with structured guidelines (often called reading guidelines or techniques in the inspection literature) improved the quality of reviews. In model-based engineering, the model development task could be viewed as a particularly well-structured review method (Ref. 10)

The ACVIP related goals for JMR Mission Systems Architecture Demonstrations (MSAD) such as the JCA Demo are to identify, validate, mature and transition methods and tools to support an architecture centric virtual integration process. This exercise also generated new modeling guidelines and tool requirements (as well as bug reports for tool developers and errata for the AADL standards committee).

The ACVIP researchers provided reports citing around 85 findings, 70 that were attributed to requirements analyses and 15 to timing analyses that will be rolled up in the JCA Demonstration Final Report. Some notable areas identified by the ACVIP team included:

- Relationship of component states and MIS system state not being fully specified
- Lack of a specification of currency/staleness for the data
- No identification of end-to-end timing requirement for hazard data
- Partition schedule not meeting ARINC 653 scheduling rules
- Non-clarity in protocol from MIS to support multiple or single instantiation of DCFM
- Non-clarity in data storage requirement between the DCFM and MIS
- Ambiguity on the MIS system Operational State when a clock timer expires
- Lack of a requirement for the number of source tracks the aircraft survivability sensor provides
- Possibility of track jitter will be seen in integration
- Multiple sensor stream rates may have implications on integration.
- Cross partition timing issues in the ARINC 653 schedule
- Inconsistency in the area of threat ranges between the DCFM and MIS making it unclear how alerts would be handled
- Potential memory leaks in MIS identified
- Ambiguity in the requirement to correlate 50 source tracks within 1 second and concern over meeting the requirement.

Some of these issues with relation to the DCFM were also cited by the DCFM vendors independently of the ACVIP researchers. At the time of this paper's writing the MIS team were able to confirm several of these and other findings by ACVIP; however, several are still to be confirmed in integration testing. A spreadsheet of the findings by the ACVIP team was sent to the MIS team to confirm the findings. The findings by the ACVIP team demonstrated that in a real program that these issues would have been identified and corrected even prior to solicitation which could have led to a cost savings and / or development schedule reduction.

CONCLUSION

ACVIP is an architectural centric model based approach that will revolutionize the way in which we analyze our systems. Results of the JCA Demo ACVIP Shadow effort demonstrated that ACVIP has potential to provide strong architectural analysis to identify and aid in the early resolution of issues. AADL is being used in many company and organization research efforts. ACVIP and its guidance, tools, and processes are going through maturation and require further refinements and maturation to be effective for future DoD acquisition of aviation mission computing systems. JMR Mission Systems Architecture Demonstrations will continue to work with the ACVIP researchers and ensure that the exercise, documentation and lessons learned mature these processes and tools so that they can effectively be used by avionics and systems engineers in the future. Industry and Government need to work together to improve ACVIP so that future development / integration efforts can benefit from early virtual integration, validation and verification.

[1] Department of the Army, Army Contracting Command. "*A Joint Multi-Role Technology Demonstrator (JMR TD) Joint Common Architecture Demonstration (JCA Demo) Broad Agency Announcement (BAA)".* Location ACC-RSA-AATD-(SPS), 2014. Solicitation Number W911W614R000002.

[2] Aerospace Vehicle Systems Institute. http://savie.avsi.aero. [Online]

[3] DOT/FAA/AR-08/32. "*Requirements Engineering Managmeent Handbook".* June 2009.

[4] Association Française d'Ingénierie Système. CPRET: System Process as Constraints, Products, Resources, input Elements and Transformations. [Online] http://en.wikipedia.org/wiki/Process_%28engineering%29#CPRET.

[5] CMU-SEI. Quality Attribute Workshop. [Online] http://www.sei.cmu.edu/architecture/tools/establish/qaw.cfm

[6] CMU SEI. Architecture Tradeoff Analysis Method [Online] http://www.sei.cmu.edu/architecture/tools/establish/atam.cfm.

[7] SAE International, AS-2C. *Architecture Analysis and Design Language (AADL) Annex Volume 3 Annex E: Error Model Annex, Draft.* Dec 2013. AS 5502/3.

[8] SAE International, SAE ARP-4761. *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment.* 1996

[9] Edmund M. Clark, Jeannette M. Wing, "*Formal Methods: State of the Art and Future Directions", ACM Computing Surveys.* 1996.

[10] Laitenberger, Oliver, "*A Survey of Software Inspection Technologies, Handbook on Software Engineering and Knowledge Engineering".* 2002.