Notes AADL Meeting Oct 29th to 31st, Washington DC

Peter Feiler, John Hatcliff, Jerome Hugues, Etienne Borde, Pierre Dissaux, Thierry LeSergent, Kyle Litwin, Jennifer Davis, Brian Larson, John Hudak, Bruce Lewis, Joe Siebel, Mark Brown, Alex Boydston

Remote – Dominique Blouin, Tyler Smith, Philip Alldrege, Ehsan Ahmad, Dave Gluch, Janet Lui, Darren Cofer, Danielle Stewart, Sam Procter

Minneapolis spring, May 2nd week of May

Bruce - News and Issues Presentation

Fall 2020–Suggestion to consider European Space Agency in Nov 2020. **Action**: Jerome will check to see if we can meet there.

Action: Bruce will check on SAE S-18's next meeting.

Action: Jerome will be the editor for the standard. He will clean up the old annexes to remove or label as superseded. Coordinate with Dorothy.

Jerome suggests offering a library of property sets with the standard for users to use.

Jerome: Runtime services depend on semantics of the core, could do as a small change to 2.2, then a larger change into V3. This is a reasonable approach.

SysML to AAL guidance, we need a day to work on it, Adventium, SEI, Collins, Ellidiss, Ansys, all would want to participate. May be too much for Toulouse (perhaps start). Report on what they should do. Should do in Minneapolis. Informal document on transformation but advise and use examples from what we have. Could expose Adventium approach in Toulouse to work on the set of requirements for the guidance document.

Perhaps the document is "Guidance For Transitioning From System Engineering (SysML) to AADL". Need to develop a scope. Could be in Toulouse.

Collins and SEI Lutz – we need a download site to auto integrate it. ALISA and Resolute are very connected. Bundling 3rd party software, SEI needs to say this is not OSATE but to bundle or make adding very easy. **Action**: Jen will speak to developers of Resolute and Agree (completed – they will set up separate download sites). **Action:** Kyle will coordinate with Lutz and Joe to resolve blocking.

Meta-Model simplification. We use text to be portable across tools. Instance metamodel is the key to analysis and will be about the same. John Hatcliff and Tyler Smith - some of our model transformations depend on the declarative model. RAMSES also does modifications of the textual model. Need to discuss more.

Language server – for cloud based use. We should discuss. Server does all the syntactic processing. Very flexible for exchange between tools. People can do work through the cloud. Certified for those servers. No impact on the standard. Not a standard issue, a tool infrastructure issue. We need a tool vendor forum. Graphical server protocol, serious for the fault tree. OBEO. Standardized instance model would help?

Pierre D - Some standardized way to manage AADL projects. Standard environment for properties. What is an AADL project. Can we export AADL project?

Etienne – What to do to finish network annex? Etienne will make a recommendation at the next meeting. He used the current defined annex properties for an AFDX analysis tool.

Adventium SysML to AADL and requirements for guidance document could be discussed in Toulouse or wait for Minneapolis. Coordinate with Peter for the next meeting, allocation of time between the SysML aspect and V3 Core. May have more time than we think.

Behavior specification in the core? Peter will demonstrate the work he has done to integrate Behavior and Error Model annexes into the standard.

Can we through informal ballots approve parts of V3 incrementally? Provides input to discuss what is in each section. We have a draft implementation of what is in the document. Need some form of executive summary of the changes. Once Peter has the executive summary of changes, we can start an informal ballot.

Tool site and Wiki. We need it outside the SEI website to have more freedom. **Action**: Jerome is working on.

Fix Sam Procter in the action list, but it should be Jerome Hugues, need update in the Action items using names, Bruce will send out.

Jen – Fix to above action. Resolute and Agree will have their own update sites. This will take a couple of months to be ready, to enable integration with OSATE. (Action done.)

Peter Feiler - Overview of V3

AADL V2.2 will be alive for some time even after availability of V3.

OSATE issue reports: https//github.com/osate

Lutz is making action language and type system available in v2.2. He treats it as an annex. ALISA has an expression language now. So as you would write in Resolute, you could write in the expression language, this helps us see what works by using 2.2. Joe can provide instructions to load it. Mark did it during the meeting.

For V3 development, we are writing the OSATE toolset from scratch.

Instance model representation will have minimal changes with documented API. Most analyses depend on instance model so they should be reasonably easy to update.

Translator from V2.2 to V3 has not been started yet. But will be provided as promised earlier.

New experiments – use of JGraphT, JGraphT works on Java, very good and well documented graphics.

Ported Fault cause (FTA) and effect analysis, did the graphics output of the analysis in JGraphT

Metamodel in EMF, Xtext, and using with JGraphT to look at the instance model

JGraphT graphs are never stored just re-generated.

5 JGraph views – for error model, for fault traces …

Exploring in what way can we unify the behavior, error modeling, modes, and flow.

For later, 737 Max error model, we can cover it if you want. (Peter did later in the meeting).

Key V3 changes:

1. Packages and General Syntax

   Import of namespaces

   Property definitions in packages

   Private classifier and property definitions

   Simpler syntax: no section keywords, no matching end identifier

   Case sensitive

2. Composition of Component Interfaces aka. Component type

   Extends of multiple interfaces

   Interface without category

   Eliminates need for feature group type

3. Configuration Specification

   Finalize design

   Configuration assignment of subcomponents with implementation, features with classifier/type (Replaces “ refined to”)

   Assign final property values to any model element

   Annotate with bindings, annexes, flows

   Configurations are composable

   Parameterized configuration limits choice points (Replaces V2 prototype)

4. Property sets – now has an extension mechanism, a complex property composed of other properties

(Jerome – can we have a mechanism where we can plug in a component and it’s interfaces – refactoring of models? Peter – main difference between version 2 and version 3 instance model – if I enable you to configure into the connections, and flows, you would have that. I needed it for flow so we could do that, I just need to make sure that it is in the document. This is like the CASE model configuration to add filters. **Action**: Peter to explore plugging in a component and it’s interfaces with configuration.)

(Brian – how to use configurations to do parameterization? Peter will cover later.)

Unified Type System

Single type system for properties and data types

Records, lists, sets, maps, unions

International System of Units

5. Properties

Simplified property value assignment (default, final, override) – simplifying determining the property value from previous levels, declarations.

Meta information for properties? Will talk about. Virus – Peter's prototype is available, John – need to represent encodings, bit level as metadata, encoding and decoding. I require a protocol or as a property. John- in 6 months we can say more about what we need.

Explicit deployment binding concept

Binding points and binding declarations

Resources associated with bindings

6. Virtual platform support

Virtual memory

Connectivity between virtual bus, processor, memory

7. Flows

(virtual) platform flows (can use on hardware)

Flow graphs – (unifying behavior specification enables a tree, in place or layered)

8. Nested component declarations

Define nested components without explicit classifier (Pierre - bringing declarative and instance model closer together.)

Connections

Distinguish feature mappings

Reach down of connection declarations

- Into named interfaces (aka feature groups)
- Into subcomponent hierarchy

9. No more category refinement

Abstract component to other component

Abstract feature to other features

We will use bindings to map between abstract and the component architecture, processor binding, we do this because the functional and the component architecture are often different so that refinement does not work well.

10. Modes
11. Annex improvements
12. Unification of Behavior and Error Behavior specification

Including support for token system semantics

Runtime Services Presentations/Discussions:

1. **John Hatcliff – Runtime services**

AADL default is that inputs are read at dispatch, some legacy does not have this, how to handle.

Threading, communication and modes

Provides foundations for automated schedulability and latency analysis that rely on the model.

AADL provides a hybrid automaton – Initialize, activate / deactivate, compute entry point, recovery and finalize

We need to be really clear about state at the input port.

If you don't follow the AADL default, then you lose some analyzability. We have to specify to the user what that is.

We could have an Input Infrastructure ports before the input port and after Output port, then Output Infrastructure. This allows a modification of the input, output ports.

Dispatch status is not defined, although referred to. So we create it.

What happens when multiple ports have events, which ones get frozen?

Await dispatch is doing too much at one time. Easier to xxx then check status.

A single put for everyone is not enough distinction.

Generation of Port-specific Wrappers is important for Contact framework.

Standard needs to relate the runtime services to the behavior annex. Correlation of runtime services with thread states.

Thread is dispatched, dispatched trigger … about 7 ways to describe. We need to make common.

The idea of dispatch is release, then executed, but when did the ports get frozen. It is apparently frozen at the point of dispatch. How does receive input fit in.

Execution complete but when do we send output. Need to put on the hybrid automata.

Send output -does it say it is released to the output service or it is completed.

We should clarify these things.

Important for scoping standard writing activity: important for helping users understand the role of the RTS in supporting use of AADL.

Need to allow a pathway for legacy code (e.g., but some parts of the analysis will be not sound.)

Need to identify/clarify the impact on other parts of the standard, if you don't obey this then you will not be able to do this. Real time scheduling constraints.

Code generation annex – needs to be tracing the implementation to the model. RMODP – Reference Model of Open … how to argue to system implementation is compliant with AADL model"

We need to provide the traceability to compliance, necessary for using AADL in certification.

Make compliance notions more "language independent"

Current Code Generation (CG) annex focuses of Ada and C

Alloy could be used to state a contract, validated with an Alloy check.

Issues – General Cleanup Issues Summarized – see presentation slides on GitHub

Relationships between …

2. **Etienne – Runtime Services (RTS) and Code Generation Recommendations**

Thanks John for his presentation. Purest statement of needs he has heard.

Why do need runtime services – need to define input and output.

If it's legacy you don't need the API, but you need it if you are writing a new application.

John - You need the state changes in the port, frozen, I need to know that they are frozen

Peter – what is the computational model of the legacy code

AADL Convention style: subprogram call –

What are those that need to be available to the user, versus used by the tool developer.

Number of services for an end user is limited, perhaps 5

Compliance – get input multiple times for background thread

"next value"

Abstract set of runtime services, then instances for languages,

Brian - Whatever the language used for coding the system, I want my semantics be the same so I can make them behave the same way. Agreed.

3. Jerome on Runtime Services

– what do we want to do with the Hybrid Automata – for different execution environments we would need to add detail or need something generic so we can bring in different O/S that we will have to work with. This is something we need to revisit.

Mark – what you have now is 100G internet and a 2G processor. How do you hold on to the data that is coming in.

Jerome – what belongs in a library and what belongs in the core standard.

Models of Computation

Relevant MoCs

AADL goal is to capture …

AADL Light – presented in May 2016. Set of resolute checks to reduce the perimeter of AADL to simple elements

Basically a minimal set of AADL for code generation and scheduling analysis

No prototype, mode, feature group, arrays, etc.

A major clean-up, based on what could be generated from an instantiation process after resolution of many rules

Rationale is to have direct traceability between AADL and code generated for build.

Provide a foundation for a model of computation.

Synchronous MoC – One definition

Two formal semantics of a subset of the AADL by Yang et al. – Paper is available.

Filali and Talpin as co-authors

TASM and operational semantics for a synchronous subset of AADL

Periodic threads + data port

Only immediate considered?

Only one CPU, no communication, fixed set of properties

Implemented as Resolute checks.

Mark - What is needed is a way to evaluate AADL to see if it is good.

Jerome – I agree, we need to have that.

Ravenscar Commutation Model

Used widely in TASTE, supported by Ocarina

Informal description exits, e.g. AADS Paper available.

ARINC653 – used in scheduling analysis and code generation

We have never discussed two critical issues

Receive_Queing_Message – Not covered by AADL semantics.

No error code if message size mismatch

EMV2 error taxonomy not considered in ARMINC653 annex

Etienne – we don't want to make AADL specific to ARINC 653, we need to be general but support 653.

Thierry – you want to use AADL as an abstraction of the 653 system (from Jerome)

John -we would provide these MoC then allow the user to select. We would have to have a way to state which analyses are still sound. You typically take a meta approach, a guidance document and how you document you're your approach.

Bruce – can we add these issues that are missing to the ARINC 653 annex when we update it after V3.

Jerome, that is what we should do, we need to remember this.

Action: Jerome will continue to develop the models of computation, then consider submitting the errata against the 653 annex.

Pierre Dissaux – SysML to AADL – Constructing a Translator Demo.

Starting with MagicDraw output as a input XMI

Prototype an example SysML to AADL.

Foreign models – comes in XMI, unreadable

How we change the configuration file? Using LMP technology. LMP is logic programing applied to model processing.

Easily modified to remap the SysML to AADL.

Can do complex mapping, and processing since we are using prolog.

Can do model transformation like SysML to AADL.

Many ways to use these language transformations. Can handle specifying different SysML tools and semantics.

Peter Feiler – Discussion of the instance model for V3:

His experimental tools and examples are downloadable. Joe is writing up instructions on how to do the build.

Many examples in V3, examples of configurable Car, example with end to end flows, working set script what is the root component, so I can auto instantiate.

Exploring the instance model graphically. Demonstrated with the views Peter has provided with Sirius, Jgraph, has developed to handle internet level component graphics.

When do analyses have to go out to the declarative model? John H in his graphics uses both.

Peter – Integrating Behavior and flow graphs, and error behavior specifications.

To simplify EMV2 specifications

To simplify these specifications vs annex declarations

To leverage V3 unified type system and expression language.

Proposal: - Can I come up with a common syntax with variation in expression notation

Token semantics for use in Meta behavior (error, security)

Existing and Desired Mechanisms

Behavior Rules – stateless and stateful, general computational expressions & token logic

  Stateless – input condition-> <output> actions

  Stateful – current state-{input condition}-> target state {actions}

Sources and Sinks

  Source - Error flows are an event, concept of a generator in the error model. Identified in behavior rule condition as trigger for out action.

  Sink – Behavior rule without output action, currently we use explicit sink keyword to indicate no output action.

  Could use this for behavior contracts.

Representation of State

  Given a behavior state, use of enumeration type to define states of a state machine

    Reusable definition of state machine states

    Transitions tend to be context specific

State machine instance

  Effectively state variable holding current state

Token System Behavior

- Data or meta data associated with input and output
- Simplified condition logic and output actions
- Context determines whether data or kind of Meta data
    - -@EM for error type tokens (EM is for Error Model)
- Types of tokens
    - Any literal: type reference, enumeration literal, numeric, string, Boolean

Token Expressions

- Input constraint as condition element
    - Input token contained in specified set
    - Input1 in <tokenset>
    - Condition expression, Output action
- Named token sets

Do we stay inline with the current BA?

Representing Error Behavior

- Separate interface consistency from behavior specification
    - Interface consistency based on in/out propagation
        - Between components

Error Behavior rules

- Currently specified in @EM{} context

Jerome - Could have multiple BA annexes to describe

Peter may take Jgraph back into V2.

Stateless rules –

- current error flows (source/sink/Path)
- With condition logic
- Token propagation:

Partial Token Behavior Specification

If no behavior rules – assume error in propagates to all output ports.

Token propagation

- Transformation: matching input condition to specified output.

Use Cases

Product line Constraints –

Usage – verification that configured system meets constraints

Demo –

By putting the behavior annex and the error annex into the core, we can go from an error state directly to a mode state.

Behavior rules are now a contract. Give me behavior rules that have …

We will still have an annex mechanism. We may use a more direct communication mechanism

Behavior and error modeling would be in the instance model …

Mark Brown – Import XML into AADL Frameworks?

1. Architecture Configuration policy (XML or AADL)
2. Multi-core processor
3. Design of multi-core processor
4. Apply formal methods to this
5. Do – complex hardware item
6. Out pops an xml file
7. Python tool: import into AADL Framework
8. Map the virtual machines into AADL
9. Using with package declarations to build into the AADL Framework
   Get customer to put their software into this turing machine.
   Customer does not have to cover the hardware elements, write the software into the turing machine.
   Time sliced execution
   Turn off cores not being used.
   Could be – each room is a virtual processor (John) – In some cases we have no O/S so would be bare.
   Schedule the partitions
   Framework would characterize the processor elements. For certification.

   What kinds of analysis do you want to use AADL for on this framework
      Analyze hardware interference.

Peter – 737 Max problem

Continuous Safety Assessment

To illustrate the importance of including lower Design Assurance Level (FAL) components in a safety risk analysis – this addition to control attitude was considered a DAL C capability.

Pilot is DAL A

To perform safety risk analysis throughout the product life cycle- even when aircraft re already in operation.

They offered the aircraft with one or two sensors. What was the risk difference? They did not provide.
They designed for high altitude, then used a low altitude.

What is the minimum model to capture this?

DAL C component in the context of DAL A components
What has been captured by the model:
Automated Climb Control
Pilot
Angle of Attack discrepancy detection (different vendors had different qualities)
Multiple AoA vendors
Aircraft engines

Use Scenarios
DAL C component in context of DAL A components
- No pilot knowledge of ACC
- Non-working AoA discrepancy light – lack of knowledge of non-functioning ACC
- ACC standby mode indictor in two AoA sensor case
- Pilot/ACC tug of war
- Degrees of pilot knowledge & mistakes: ACC switch off – he may forget, mistake
- Quality of AoA
    - Omission, Bad value, sym. Approx. value if both bad
    - DAL based spec, known low product quality of one vendor
- One vs. two AoA sensors: purchase option without quantified safety risk buy down
- Unbounded degree of freedom on flight surface control (apply over and over)

Component Fault Behavior
- Error source for Service Omission and Bad Value
- Failure probability of 1.e-5 (DAL C) and vendor specific values reflecting track record.

AoA discrepancy detector
- …

Potential Refinement of AADL Model
Reflect design issues in ACC
- Unbounded degree of freedom on flight surface control due to repeated ACC activation
- Change in degree of freedom of ACC control
- …

Quantified Safety Risk Assessment
Different aircraft configurations

Failure probability calculation … See documentation, presentation and paper

Charlie Payne – Tutorial on MILS, RMF

Charlie and Dave have met, this could fold into the Security Annex.

Architecture Analysis for Cybersecurity
NIST 800-53
Is correct, complete, resistant to direct attack, and resistant to by-passable, tamper …
Impact of this analysis is every system that has embedded cyber controls.
Focus on questions that we can answer through architectural analysis
Annotate the model so that architectural changes impact analysis results
Understand ground truth of the security problem
Isolation requirements dictate that only an AADL execution platform mays serve as a Cross Domain Solution (CDS) for MILS Analysis
Need to design and verify correctness with the same toolset.
Usable in the continuous virtual integration capability that Adventium has
DoDI 8540.01 – Cross Domain Policy
DoDI 8510.01 – Risk Management Framework
DoDI 8500.01 – Cyber security

Alex – Is there another tool to recognize attack surfaces

Access Cross Domain Solution (CDS), or a Transfer CDS
AADL really shines to understand shared resources and common binding, these are required for other analyses and the design of the system. AADL naturally supports an incremental and automated RMF analysis.

System access points graphed relative to the components and connections, types of connections.

You can run the tool on everything from a black box to a fully defined system supporting incremental assessment and alerts to violations when they occur rather than late qualification discovery.

RMF Analysis Tool
- Reduce the risk that ...
- Currently we have a weak enforcement of the requirement – like a cherry on top, PM's are not doing this depth of analysis now on programs.
- Charlie - we have provided a chain of evidence to show that it is correct, evidence of what components, their security levels, let the tool do the analysis. You need a validation change that a human can validate. With 30 years of experience, I know this is critical. We have designed for that.
- Independent test can be used to cross validate the analysis.

- We still need to verify that the control work correctly.
- Modeling information flows is critical for any attack tree analysis.
- AADL gives us the ability to reason about these flows at all levels in the system architecture
- Once we have that, we can look at the attack tree and cross verify that we have what we need.
- Confidentiality, Integrity, Availability
- Enforced_ccis – these area assessment obligations, example requirement – "user input of sign on will lock up after 3 attempts"
- Low confidentiality may impact the high confidentiality, may interfere -slide 19
- Tool reports the errors, but it also includes context to interpret those errors.
- Existence Analysis – every process must implement all the controls needed for the flow.
- Would also have to make sure that the operating system also has the same controls.
- Non-Bypass ability Analysis – may implement flows on execution platforms shared by other flows. Must verify that they can run in isolation.
- You could develop an attack tree, run the tool, find the bypasses, therefore rank the bypasses on multiple potential attacks.
- Slide 27 – includes the needed fix for the bypasses discovered.
- Cary – how do you prioritize these, a manual effort now. Could the bypasses be ranked? Charlie – we show the criticalities of the flows you could disrupt to could show impact of this , so could rank.
- Could do more analysis from the information provided to find the best fixes.
- Tamper Resistance analysis – we said three attempts, tamper would be to change to 300, now the control allows potential penetration.
- Since policy CCIs take many forms (file, registry, database, etc.) the tool produced warnings, you can move the process, but someone would have to review.
- Key takeaways – you run mixed criticality analysis to verify flow isolation before adding hardware, you run Existence, non-bypass, and tamper resistance to verify they are well placed on the platform. Cary – we need the architecture specifications to use this. Legacy systems, how would you get the specification? That would be part of the analysis approach, to build the specification if it does not exist.
- Cary – Penetration tests are not complete; they relate to what we decide to test. Formal analysis looking at the as implemented architecture (we can generate to the specification) can look at every possible approach.

Dave Gluch – Security Annex Update

What Charlie is doing is very complementary to what I am doing. Charlie and I are talking. Should be able to add to the Security Annex in more detail on MILS and also RMF.

New structure of the document proposed to enable capture of analysis approaches.

Noteworthy Changes to the draft reviewed - see presentation.