AADL standards meeting Nov 5-8, 2018 Draft 6

* Location London, UK.
	+ Meeting information
		- Aerotech Venue, the Radisson Blu Edwardian Heathrow hotel
		- 140 Bath Road, Hayes – Middlesex, UB3 5AW – United Kingdom
		- <https://www.radissonblu-edwardian.com>
		- AADL meeting – Private Room 373rd, Floor - Boardroom10

If staying at the Venue, the hotel room block for ASTC 2018 will close Oct 5. Please see the link below in order to make reservations in this hotel for the Aerospace Systems and Technology Conference this November.

<http://astc18.org/venue-accommodation/>

* + - Marriott across the street is worth checking as well. Several of us are there.
	+ All times are local, London, UK
	+ Fee is $300 if you prepay through <http://www.sae.org/servlets/works/committeeHome.do?comtID=TEAAS2C>

**On Site**: Peter Feiler, Bruce Lewis, Lutz Wrage, Pierre Dissaux, Tyler Smith, Brian Larson, Alex Boydston, Dominique Blouin, Dave Redman, Jerome Hugues, Mirko Jakovljevic, Thierry Lesergent, John Clatworthy

**Remote**: Steve Vestal, Joe Seibel, Ehsan Ahmad, Philip Alldredge, Dave Gluch, Tran Hai Nam, Charlie Payne, Etienne Borde, John Hatcliff, Brendan Hall, Cary Pool,

# Monday, Nov 5

* 0900-1000: AADL standardization committee news + action items (Bruce Lewis)
	+ **Action**: Dominique – will confirm the dates for Paris. June 10-13 2019 currently.
	+ Get dates for SAE meeting near Grand Canyon. Fall 2019
	+ Winter 2020 Toulouse
	+ Spring 2020 – Minneapolis
	+ Download FACE annex can be found on the Adventium camet/face page.
	+ Workshop – Flow presentations from SysML to AADL Analyses, early to late analysis.
	+ Google Forum, Tyler to do a web page, let ANSYS know. Next week send to Tyler, Peter, Bruce
* 1000-1045: FACE Overview with BALSA Example Translated and Explained (Tyler Smith)
	+ FACE provides a portable component concept to go across platforms
	+ App layer, Transport layer, platform services, and drivers (IOS)
	+ Is a data modeling standard fundamentally, huge metamodel for data modeling
	+ Then the different layers that use the data model to communicate.
	+ Restricts O/S calls, requires using the defined data types.
	+ There are 3 or 4 ways to model UoP’s in AADL, we need to specify how it is done so AADL models also are portable across contractors when using FACE UoPs.
* 1045-1100: Demonstration of FACE Annex in ANSYS Toolset (Thierry Le Sergent) (w break) Record
* 1100-1200: Review of FACE Annex w comments from informal ballot (Tyler Smith)
	+ If we want to go both ways, can we support in this document? Yes, another section of the document.
	+ What to have the wording not imply the direction rather than two section.
	+ UUID – AADL to FACE? UUID’s could become a standard AADL instead of text string under V3. This doc for 2.2. It comes from the project, each project provides it’s own to allow renaming but keeping it’s identity. Generator is based on time to make it unique. A time stamp but just an identifier. For change tracking. Used for requirements somethings as well. We have to provide a UUID if we go from AADL to FACE. From AADL you would want to generate it automatically. In tools it’s hidden. If you export AADL text, you would provide. Tyler will insert a statement on generating UUID from AADL for generated UoPs.
	+ UUID, is it required for component instances used for subprograms. You would have the option to provide.
	+ Need to explain why you are not using client server in AADL. In AADL it goes back to the same component, in FACE not restricted to the same component. Tyler will add explanation.
	+ FACE just has a number for period property. Assumed seconds, believe that is right but needs to be specified by FACE. Must translate into seconds or microseconds as needed in the AADL translator.
	+ TSS – AADL to FACE – Tyler will add a sentence to specify how the AADL will translate to a TSS. That will provide portability and explicit modeling.
	+ Error types, should we map between FACE (ARINC 653) and EMV2. Not necessary. Out of scope.
	+ Profiles to calls made to be applies to (all). We are not sure how they will model the TSS.
* 1200-1230: AADL v3- Roadmap (Peter Feiler)
	+ AADL V2.2 Errata: <https://github.com/saeaadl/aadlv2.2>
	+ OSATE issue reports: //github.com/osate
	+ Prototype //github.com/saeaadl/AadlV3Prototype
	+ Working slides //github.com/saeaadl/aadlv3/issues
	+ To finish in 9 months
* 1230-1400: Lunch
	+ ANSYS – FACE/AADL support
	+ Synchronizer – one language to another, bidirectional, push and pull
	+ One project may be sync with multiple other models, SCADE Architect, SCADE Suite
	+ FACE-AADL sync, makes a transformation.
	+ SysML -AADL would be another we plan.
	+ Input project file, transform, output new model.
	+ Library by library to provide scaling
	+ Chose what you convert, all the subcomponents you want. This UoP with this AADL model.
	+ Push to get AADL from FACE UoPs
	+ Obeys the constraints of each language, can not put processor in a process.
	+ Bidirectional, should get the same model back.
	+ Some manual transformation, part of the support to add processor to AADL model. FACE does not specify.
	+ If you go from AADL to FACE you transfer the max value of a range – Tyler
	+ Interfacing the data model of FACE to AADL, something we want. What about the feature group. Should the FACE view map to the feature group of AADL. Mixing of the view (wide set of data) that must be split to go to the components. Aggregate data, should this be a protocol, need an interface. Timing and when aggregation occurs. Problem of asynchronous aggregation.
	+ Concept of feature groups was a simplified way to transfer data from multiple ports to multiple ports. Inverse concept. Now called interface in V3.
	+ Does it make sense to say inverse or …
	+ Directionality – can be a handshaking protocol, data in both directions.
	+ Entity – data in a context. Car has 4 wheels, then data for that model.
	+ Feature groups as views, and Annex. Tyler will think about.
	+ Which feature group is an aggregation, which is a set of interfaces.
	+ **Decision**: Stick with what we have in the Annex. Get detailed comments from Thierry on the ballot.
* 1400-1530: AADLv3- Tutorial - Features and General Syntax: Examples & demo (Peter Feiler) Record?
	+ Allow property definitions and type in packages – **accepted**.
	+ Nested packages – metamodel much simplier, captured the semantics of decent
	+ Additional rules will be formally documented along with the metamodel.
	+ Standard document will be more formal.
	+ Property constraints – what will be allowed will be covered later.
	+ Prototype metamodel will be thrown away. Once standardized will develop from the beginning.
	+ **Decision:** Qualified nesting of packages: yes
	+ **Decision:** Dot notation for nested packages – deferred.
	+ **Decision:** Make part of a package visible – import – yes, could provide a local name here.
	+ **Decision:** Qualified name references are not required – yes but (Alexey, Jerome not yet, much discussion). Replaces with clause and renames declarations. You have to qualify if there is a name conflict. You can import and refine.
	+ **Public and Private sections** – Proposal, Eliminate public and private sections in packages, Allow classifier definitions to be marked as private. **Decision**: Yes
	+ Make AADL Case Sensitive: Decision: Jerome – is a tool issue, User ?, Lutz, Brian yes, all other languages are case sensitive. FACE is case sensitive with a larger character set. SysML is case sensitive. So we need to transfer between. Identifiers – yes, keywords all lower, identifiers would be mixture and could be all lower as well. Decision: Yes
	+ Arbitrary section ordering: Yes
	+ end is all that is needed at the end. Yes (all but Brian)
	+ # to indicate a property value follows. #Period=>50 ms; Decision:yes
	+ Component categories: Stay with abstract for now.
	+ Interface is not yet a component. It’s used in a component. Thread interface
	+ Nested subcomponent declarations, with no section labels but with {}, “is” with sections. Or all {}. Or {} with sections? Looks like an instance model representation.
	+ Can convert to a reusable or a single instance. Reusable is types and implementations for some components.
	+ Section labeling – how does it fit into this nested approach.
	+ Connection, binding … sections use the same ->, understood by the section.
	+ Nested has many implicit interfaces, not easy to go backward to reusable component definitions. Significant debate.
* 1530-1600: Break
* 1600-1730: AADL v3 -Interface Composition: discussion and prototype demo (Peter Feiler)
	+
* 1730-1800: AADL Behavior Annex errata (Pierre Dissaux)
	+ Waiting for runtime services to be defined. Then can address the issues that are open.
	+ Adequate for now. Exceptions are yet to be defined but runtime dependent.

# Tuesday, Nov 6

* 0900-1030: AADL v3- Configurations: discussion and demo (Feiler)
	+ Multiple Interfaces can compose a type.
	+ Takes the place of a feature group.
	+ Must be consistent with the component category
	+ Interface extension allows addition of features, flows, properties, adding data types to ports
	+ Can not have name conflicts, Brian – prefer the simple rule of no conflict.
	+ Reverse – reverses the directions of ports by interface.
	+ Multiple interfaces can be combined.
	+ Interface used like feature group.
	+ Single arrow denotes single direction, double arrow indications both directions, implies that there must be a reverse which would be specified on the component interface.
	+ Use as Aggregate Port - should we do this, needs all kinds of restrictions, specialized use for data ports.
	+ Nested interfaces –
	+ Does it make sense to have component level properties in composable interfaces? If a property belongs to a category it does not make sense.
	+ Don’t make properties more complex, don’t put in composable interfaces (Lutz, Tyler)
	+ Compositional effects of Error models for each interface?
	+ Feature name mapping for connections, reusable – keep on shelf, is it really needed? Just wrap one side to make it fit with the other.
	+ Reaching down with inline mappings (need for flows deeper than one level).
	+ Do not have special rules relative to name conflicts, keep simple in composition.
	+ Jumping back - In nested specification, you can include/require full interfaces. Readability/understandability is a key issue.

1030-1100: break

* 1100-1230: AADL v3 Configurations (Feiler)
	+ Keeping the specification stable except what we want to leave open for change.
	+ Revise and add to existing architecture design structure
	+ Elaborate but do not change architecture structure
	+ As extension of component implementation, configuration, interface
	+ Can reach down vs parameterization
	+ Elaborate and annotate the leaf notes, the subcomponent substructure
	+ Annotate substructure with “final” property values, bindings, annex subclauses
	+ Assign configuration for existing subcomponent implementation
	+ \*.secure is confusing, is it needed or a special case?
	+ Best to put a generic type and then substitute rather than wild card. Wild cart will probably not be there.
	+ Adding annexes for additional analysis on a configuration
	+ Composition of flow configurations.
	+ Next meeting will discuss flows, Peter working on now.
	+ How to do configurations graphically?
		- What can we do to support, they want to work in graphics.
		- AADL does not require everything can be expressed in graphics since it is textual.
		- Graphical tools do not have to be all graphics- text boxes or tables.
	+ **Action**: (send email to Eric Feron that we may have some extra people – three or four, plus room to demo tools).
	+ **Action**: (organize a set of presentations for Thursday, time for each volunteer)
	+ **Action**: (determine exact dates for June meeting in Paris)
* 1230-1400: Lunch
* 1400-1530: Security Annex (Dave Gluch)
	+ ALISA Security Analysis Examples are available in //github.com/osate/examples.git
	+ ReqSpec of ALISA –
	+ Do we want to define plug-ins to support security?
	+ Slides show specification in ALISA, we should use natural language until we have an expression.
	+ Private key – should be a key ID number as a data type. Then put properties on it.
	+ Real problem is source of key, how it gets changed, how platforms use at different bases.
	+ Authentication – verifying paths
	+ Worth analyzing developing a state machine with the BA to control key management.
	+ Peter suggested layering that would not require a state machine. **Action**: Dave will look at what we have already.
	+ Action/Command Protection – must be authenticated to do something.
	+ Rule – two keys needed to launch is part of Command protection.
	+ Many aspects need to be established with behavior. It’s a dynamic architecture. Implementation of a thread conforms to a security policy. How much is it static analysis. Peter thinks much can be static. Charlie – its more about where the protection is needed in the architecture. We at the architecture level do not need to verify the process of Key management. It’s more about the protection level needed for architecture components – Tyler. **Action:** Dave will look at developing properties for an example – two keys required. Resolute used key as an object then used analysis.
	+ Attack tree – external attack tree provides the input that we use to consider the internal possible effects of attack. Propagation/resolution.
	+ **Action**: Dave – will provide a draft document for the next meeting.
* Steve Vestal – AADL FLOWS Tutorial
	+ Flow Analysis tools were the source of this tutorial
	+ Latency – Input to first output
	+ We only consider the data transfers that are guaranteed. It’s the worst case. Yellow is best case but not guaranteed. Green arrows represent what is guaranteed given no errors.
	+ Will produce an upper bound for each scheduler then integrate the times, with the impact of sync and asysc effects.
* Errata –
	+ Deprecaited Properties – do we have a way to specify. Discussed but not accepted.
	+ Bring over MIPS Budget and Demand. Only had brought over from SEI one of the
	+ Correction of levels of flow look down. Now can look down only one level. Accepted.
	+ Mode transitions cannot reference a feature in a feature array – proposal will be developed.

# Wednesday, Nov 7

Sessions in Conference: Integrated Architectures and IMA

**Multicore processors: How to Implement Multi-CAL Level Application with Security Stakes**

How to segregate processing. How to use the performance. How to certify.

IMA now mono-core, many core is more like network on chip, with switches.

SMP – one single OS shared among all the cores – one application spread across cores.

AMP- A private OS instance per core – sharing of resources difficult.

A virtualization layer hosting different operating systems in dedicated virtual machines. This is what we do. Layer discovers the cores. All interactions managed by layer.

Correcting for radiation upsets difficult since it’s internal.

Safety also involves interaction with other devices/computers the pilot may use.

Multi-core has hidden channels, must know if they are attacked.

Determinism Platform = prediction for execution+fult tolerance

Must demonstrate there is no hidden channel that can be attacked.

CAST 32 paper – Interferences aware safety & security process – Interference analysis, what mitigation for interference.

Manageable bounds

State of the art – determinism mastering

 Deterministic Execution Model, Deterministic Adaptive Scheduling, …

MILS is starting to take place in secure systems to support multiple applications at different security levels. Must combine safety and security, MILS and ARINC 653.

Q. How do you find one person who has safety, security, and multi-core expertise to certify (DER)

A. Hidden channels can open the door to DAL A applications, between processors.

**Model-Based Systems Engineering** –

A concurrent process that develops a design specification from customer needs.

DIMA – Distributed Integrated modular Architecture

Key challenges – Generic architecture, Variant technolocal implementations, Human-machine Interface, controller architecture.

ARCADIA/Capella Methodology – Operational analysis, system analysis, logical architecture, physical architecture

First diagram – data flow, sequence, modes, … together we have the dynamic behavior of the system (interaction of components).

Aircraft-level, system-level, item-level (software, hardware),

Functional development, allocation of functions, development of architecture, allocation of requirements, system implementation, data and documentation

OA-SA-LA-PA

Data flow is at multiple levels.

Physical Architecture has the hardware/653

Q – Does it support Safety and Security analysis? Once you have the architecture then you call apply.

**Using Multicore processors for Safety Critical Aviation**

Why :

End of silicon availability of single-core processors

We have very little influence on what is developed, driven by commercial use.

We always need more computing resources

SWAP – Space Weight And Power optimization

Shows flight control on an IMA architecture partitioned with 653

Design goals for VXworks - Enable IMA development & certification RTCA DO-297 with certification DO-178C

DO-297 provides a nice framework or process. Tasks - Module acceptance, application acceptance, IMA system acceptance, Aircraft integration of IMA system, Change of modules or applications, reuse of modules or applications.

Hypervisor-based module operating system used to cross multiple cores -SMP

Key characteristics – shared resources, strong enforcement of partitions.

CAST-32A Guidance - Software Planning, Planning and Setting of MCP Resources, interference Channels and Resource Usage, software verification, Error Detection and handling, and safety, dynamic behavior is not tackled. But we are looking toward, would need to speak to certification authority, self modifying code. Hyperthreading.

Does not include things like WCET, but DO-78C does. But does look at Resource Usage and interference channels.

Then WCET Test times, Vulnerability assessment etc. Hardware monitoring tools, if qualified can be really helpful for resolving interference.

Rockwell Collins / Wind River working together on Multi-Core. Website or paper referenced with web address.

* 1130-1200: Multi-Organizational Modeling with AADL (Tyler Smith, …)
	+ **Multi-Organization Model Integration**
	+ Do we save money? Can we use the same proven practices for software, like continuous testing, with models?
	+ Mission System Integrator, Customer, supplier, source of truth
	+ Integrate the model correctly. How successful
	+ Passing models then integrated by the single source of truth contractor. This enables changing MSI.
	+ Test harness to the supplier – captures all interface and requirements in the form of a model that is used as a test harness.
	+ We were able to provide privacy on part of MSI
	+ **No big spike at the end for integration costs!** In fact, linear increase in effort to complete program.
	+ Immediately detected overrun on CPU capacity, virtually. Corrected immediately. But also had reviews.
	+ Verified by the virtual integration and physical integration.
	+ Test harness specified what kinds of analysis would be needed so supplier could
	+ Summary - We protected IP, used continuous virtual integration, discovered issues early, and integrated smoothly models in AADL.
* 1200-1230: An Integrated Approach to Model Based Engineering with SysML, AADL, and FACE (Wang Zhe, Jerome Hughes, Thierry LeSergent, …)
	+ **SysML and AADL, FACE integrated Modeling – Jerome Hugues**
	+ SysML – Operational Analysis, System requirements and functional analysis, system Architecture Definition.
	+ AADL – System Physical Architecture Definition, Computer platform Architecture Definition, Application Software Runtime Architecture Definition
	+ FACE – UoP definition
	+ FACE-SCADE – UoP software Model Design, UoP code generation
	+ SysML is the way to design for the client – you know what he wants and have captured it to a level of abstraction. Operational Analysis.
	+ The only SysML version is the last one, that is the one you convert to AADL.
	+ Non-leaf function -> Package in AADL.
	+ Leaf function -> System
	+ Port (one data)-> Data port
	+ Port (several data)-> Feature group
	+ System Physical Architecture Definition – in AADL, define, then define platform architecture.
	+ SysML – can be modes or component state, need to separate in AADL into two modes, and BA
	+ Refine the type of connection, need to do design space exporation and talk to the System Engineer to see what he had in mind.
	+ What type of data must be sync’d. This is not in SysML, so may have to be directly captured from the engineer.
	+ Performance analysis – latency, safety, scheduling for instance.
	+ AADL to FACE gateway -working on the AADL annex. Gateway is not fully automated. Some aspects manual.
	+ SCADE 2019 01 will support this kind of analysis.
* 1230-1330: Lunch then reassemble at Connaught Suite B, RTOS and Software Platforms
* 1330-1430: Formal, Architecture-Driven Assurance for Cyber Security with AADL and Trusted Build (Mike Whalen, …)
	+ **Formal, Architecture-Driven Assurance with AADL and Trusted Build – Mike Whalen**
	+ Significant use of the AADL to make aviation system secure.
	+ We need to think about how to make vehicles secure.
	+ We were able to build a secure system that withstood major technology driven hacks.
	+ Auto – used on-star to take over a car 100 miles away. We are also susceptible in aviation.
	+ Hack of gps convinced the UAV to land. Iran stole the secret UAV.
	+ Security vulnerabilities that can lead to safety issues.
	+ Cyber Hard UAVs – Secure Analyzable Architecture – one we can analyze, captures all forms of communication (no back door). Secure Software components, no C buffer overflow. No C runtime errors,
	+ HACMS – $80M on air team, $20M on ground team. The ground team could not integrate their system, so they used the Rockwell tools and methods. So all $100M on the Rockwell/AADL approach. Three 18 month phases, at the end of phase two we had it working. We certified, did flight test, pretty much replaced the software of Little Bird.
	+ CAN is a bad bus, so could we secure the CAN bus?
	+ Put a new processor on the quadcopter to make it more like the ultimate target, the Little Bird.
	+ O/S was guaranteed partitions. Proved correct.
	+ A lot of issues come from bad architecture. The radio and the critical engine, brakes all on the same bus with no authentication.
	+ If we have an architecture that everyone has confidence in. We generate to build an architecture model that is correct. Components are correct – realizable contracts, components verified to implementation contracts. AADL gives you a system image for execution with incorporated components. This is what we needed.
	+ To trust the Architecture model we have to have ground truth, then generate. No back channels, in application or O/S.
	+ Once we developed the architecture we did vulnerability assessment.
	+ Before change – Red Team broke into Little Bird in 45 minutes. Immediately into the quadcopter. Baseline
	+ Secure comms, component errors, … We did also our own Red Teaming, you should be doing this.
	+ Secure dataflows, Memory safety, Cruntimie errors, Task blocking, Respond to DoS attacks
	+ Common Attack Pattern Enumeration and Classification, <http://capec.mitre.org>
	+ Extended OSATE, to do our own trusted build.
	+ Functional and architectural analysis with AGREE, Resolute, Lute,
	+ Agree on a set of AADL to do the trusted build, what part of AADL you need.
	+ Model Checking – Breaking down and composing up the proof of correctness.
	+ Assurance cases – had to combine proof from multiple sources, the Sel4, architecture (AGREE), memory safe components (Galois). Assurance cases informality helped combine the proofs. But the neg is also informal not strongly tied to the architecture. So we developed Resolute: An Assurance Case Language for Architecture. Example – if SeL4 then safe and if developed with safe tools, is safe.
	+ WE had to annotate the model with additional evidence. Not all the components were developed memory safe.
	+ Resolute can produce a failed assurance case. If no path, then failed. No backchannel if assurance case succeeds.
	+ UAV motors only execute commands from the ground station.
	+ Don’t fool yourself. Make sure it’s right and complete. Make claims on everything that runs. – Every component can run safely.
	+ Sel4 was key. They have full stack proof. Bugs eliminated – buffer overflow …
	+ Why Trusted Build – Ensure fidelity between models and system image. Proofs are over architectural models.
	+ AADL – Scheduling, representation and the underlying semantics,
	+ Mission system image. – build the make file to integrate the components,
	+ Data Recorder and External USB Flash Drivers were used to break into the system before trusted build. Walled off with a virtual machine and check it’s not being spoofed.
	+ We did not do GPS spoofing correction or encapsulation of radio control. So these were not in the experiment.
	+ Red Team 3 month assessment, had all source code, all artifacts: design documents, source code, architecture models, system images. They there not allowed to change our code.
	+ We did it without a performance penalty. Works as fast as original.
	+ Key drivers – thoroughly develop understanding of the architecture and insure that that trusted build provided it.
	+ All the comm was generated, glue code verified,
	+ CASE – is the follow-on
	+ Little Bird was qualified to Fly. Doug Stuart
* 1400-1430: Integrating the process, SysML and AADL Guidance (Brainstorming, Jerome Hughes)
	+ **Action**: Bruce contact Thierry, Jerome on a session on SysML/AADL Guidance for next meeting.
* 1430-1530 AADL Networking Annex (Alexey Khoroshilov)
	+ Networking Annex – safety layer of the network, how do we add a safety level. Tend to be proprietary, capture error models. How do we do. TT Ethernet – safety protocols, VISTUS – for a test bench, fidelity.
	+ **Action**: Brendan will send current draft to Mirko, Mirko will discuss Autozar use and requirements with Jerome, Brendan, Alexey. Only thing we may need now is some additional properties for TT Architecture.
* 1530-1600: Break
* 1600-1630: AADL Inspector 1.7 released - Presentation and Demo (Pierre Dissaux)
	+ We support import form SysML, NoMagic, but have very few examples. Also Cappela, Marte
	+ AADL Inspector
* 1630-1730: AADL V3 Type System (Lutz Wrage)
	+ Python is currently being used on the V3 prototype
	+ Current usage of types – data types, classifiers, AGREE, Resolute, … user defined types, port types,
	+ Current Composite types- range of, list of, record, Data implementations, property expressions provide syntax for literals, ReqSpec adds expressions, uses simple type inference
	+ V3 Goal - Data type, property type, types for annex sublanguage.
	+ Base types – Numeric, Boolean, String, Enumeration, Unit, Category (thread, processor, etc.), Classifier, Model Element, Range of Numeric (Compute\_Execution\_Time=> 10ms ..15ms)
	+ Composite types
	+ Array (ordered sequence of fixed length)
	+ List (ordered sequence of arbitrary length)
	+ Record, Union (named alternatives)
	+ Tuples (unnamed fields)
	+ Map (modal and binding specific property values
	+ Bag, Graph
	+ Which should be built-in? Probably not bag.
	+ Properties on error types.
	+ Remain strongly typed.
	+ Conversion types
	+ Expression Language:
		- Literals <package>.<enum tupe>.<enum literal>
		- Operations 2 constraint language support
		- Operations 3 Types, values, literals, Expressions, functions, recursive functions (now we have a programming language) but very powerful for constraint language
* MBSE – Raphiel
	+ We use standardized with informal semantics, but it is more formal than natural language, moving from documentation to models.
	+ Consistency is a strong requrements engineering challenge, can you ensure there is no conflict in 1,000 requirements, what about a million.
		- Based on a system engineering language, SysML – standard notation – we need to add method to the notation.
		- Function based definition (ARP4754A)
		- Functional simulation to ease early validation
		- Support of bidirectional traceability (certify ad & change mgt)
		- Consistent-by-construction (all elements are connected)
		- Pilot CASE OMS is a software intensive avionics systems that monitor health of the aircraft during flight.
		- Use Cases provides structure input requirements, then to sequence diagrams to capture interactions in the use case, these are the first ideas of the functions, then to activitiy diagrams to capture function interactions with the function as the view, then to the logical architecture showing interconnected functions. Finally you would go to the physical architecture where we have separation of software and hardware functionality.

# Thursday, Nov 8

* 0900-1030: Tutorial with Installation – RAMES AADL Code Generation with case study (Lego Mindstorm Robot) (Dominique Blouin)
	+ RAMES (Refinement of AADL Models for Embedded Systems) Roadmap and tutorial – will be a chapter on AADL for CPS.
	+ Context
	+ AADL for CPS: Lego Robot Case Study
	+ RAMSES and Code Generation for Case Study
	+ Mixed Criticality Scheduling with AADL
	+ COST: European Cooperation in Science and Technology – covers the cost of goinig to meetings to discuss research. Brings researhers together to discuss what they are doing. Sponsors schools (2 weeks, 700 Euro each).
	+ Multi-Paradigm Modeling for Cyber-Physical Systems – large European research project.
		- Model every part and aspect of a system explicitly
		- Avoid tweeking languages to make them work.
		- Most appropriate levels of abstraction
		- Using the most appropriate modeling formalisms
	+ MPM4CPS – 4 teams, wg1-Foundations, wg4-Education and dissemination, wg2, wg3 currently do not have leaders.
	+ Nov 18-21 training schools, three hours with AADL and tools, Analysis with OSATE and AADL Inspector, Code generation with RAMSES. Adopted Lego Mindstorm
	+ Book on Formalisms for CPS – Multi-Paradigm Modelling for Cyber-Physical Systems, Vol 1 Formalisms, will have the chapter on AADL.
	+ Modeling the Architecture of Cyber-Physical Systems with AADL
		- AADL has both cyber and hardware, can start with very abstract, refined with rigorous semantics, well integrated.
		- Line follower robot to carry objects in a warehouse
			* Pick-up object, follow a line, drop off object, avoid obsticals
			* Ha sonar, automatic C code generation from AADL from RAMSES NXT OSEK middleware, flashed to memory, Mostly glue code but BA can generate C code. //lejoe-osek…
			* Line follower looks for the edge, half black half white. Three approaches, best proportional PID .
			* Formalisms Transformation Graph and Process Modem (FTG+PM)
			* FTG=PM Framework for Multi-Paradigm Modeling See slide tracing method from System Overview to C code generation of the system with integrated components.
			* Context diagram in AADL based on system usage scenario and goals.
				+ Line following robot, Line, floor, obstacle, light source. Shows interaction points.
				+ Recorded requirements mapped through ALISA – example minimum luminosity for light source.
			* Functional Architecture: High Level Requirements
				+ Carry\_object\_function, follow\_line\_function
				+ Map and decompose the
			* Software Architecture: Follow line subprogram (could be abstract)
				+ Define interfaces and data types.
				+ refine data through extension
				+ apply standard properties
			* Software Architecture - Follow Line Subprogram
				+ Data flow through ports to output.
				+ State: stop, forward
			* Physical Plant Model: Hardware part – Line Follower NXT Robot.
				+ Gives us bus, device, etc to capture with more semantic meaning.
				+ Hardware Kit Specification used to model in AADL the hardware.
			* Deployment
				+ Integrating the hardware and software in AADL
			* Analysis
				+ Measure execution time of subprograms on brick, latency in steerng, scheduling with Cheddar,
			* Code Generation
				+ Develop tools that can be used independently (does not require RAMSES)

Switched Ethernet Flow Analysis person

Initial AADL models-> RAMSES model refinement-redo analyses, update model-> repeat cycle until satisfied->then RAMSES code generator->generation for target platform

RAMSES Refinement Rules – takes two threads communicating by periodic ports transformed into shared variable approach.

Many cases. Do transformation on instance model, then regenerate the declarative model.

Based on graph transformation language, much more complicated that it first appears.

WE could consider instance to declarative model transformations in general case.

We have our own traceability. We can go back to the original model.

Supporting POSIX, ARINC653, OSEK

* + - * + RAMSES demo – plug-in for OSATE
			* Four step process
				+ Model Validation

Model transformation AADL->Validation report

Implemented in ATL

Resolute to simplify?

* + - * + Refinement

Model transformation AADL->AADL

Implemented in ATL

* + - * + Code Generation

AADL->Code (C or Ada)

Implemented in Java

* + - * + Code loading on system
		- Current Issues – ATL is slow
			* Low Maintainability of ATL EMFTVM
			* Issues with bugs, documentation
		- Issue – workflow needs to be automated
			* User selects the transformations he is interested in, could be trade-space analysis using multiple transformations.
		- Refactoring Runtime services – for code generation.
		- AADL and ROS (Robotics Operating System)
			* ROS is used widely, but has no real time aspect, needs code generation. Rapid prototyping. Higher level commands.
		- Roadmap for RAMSES
			* Performance and expressivity issues
			* Looking into LAMPS
			* Benchmark transformation tools
			* Study scalability by incremental approach.
			* Study bi-directional approaches
			* Model management – larger context of SSoT
			* Properties preservation of refinements.
		- Mixed Criticality Systems
			* Due to safety requirements you can not mix
			* Current methods waste computation power
			* Mixed-Criticality done by mode switching between standard and critical systems. Can have multiple levels.
			* Scheduling Muti-Periodic Mixed-Criticality as late as possible to give time to low criticality.
			* Small domain specific language to capture – did not use AADL for this.
			* Also have a workflow language to work with OSATE and other Eclipse.
			* **Action**: Dominique and Tyler – we should see if we can add an “Ant Task” to this workflow. Action: Dominique and Lutz – there are simplifications to OSATE that would help run it stand alone.
		- More on Model Synchronization
			* RAMSES and AADL are very good case study, very rich and complex, need for incremental approach.
			* miGMM DFG funded project
				+ three years, just started, Dagstuhl meeting on Dec 2-7, multidirectional Transformations and Synchronizations, have also done.
* 1030-1100: break
* 1100-1230: Tutorial continued.
* 1230-1330: Lunch
* 1330-1430: Code Generation and AADL Runtime Services Lessons Learned (John Hatcliff)
	+ The Slang Embedded Code Generation Framework
	+ Goals
		- Code generation from AADL
		- With integrated Verification of the AADL
		- Sireum Programming Language – uses Slang, and subset of Scala
		- Supports SPARK/Ada-like code generation and also … supports formal verification (not yet implemented). Code is generated in C.
		- Can have BLESS annotations optionally
		- JSON – designed by Jerome transformation
		- Example Simple Building Control
			* Sensor, temp control, fan, operator interface
			* Uses all the connection types.
			* Styles of communication between components can be specified.
			* Code generation using XML representation from Jerome Hugues
			* Supports periodic and sporadic threads,
			* Glue code and ports
			* Ports are overridden by user, to interface to the source code (GetValue)
			* Issue? Multiple instances of component (handled in Slang for C or Ada).
			* AADL RT Services dsipatchStatus, receiveInput, putvalue, getvalue, sendOutput
			* Port specific port commands generated. sendfanCmf.
			* We can watch as things happen in the running system, debuggers etc.
			* Slang Debugging. Allows insert of values at an output or input ports.
			* Other interactions – fault injection, etc., watchers. Filters – calls on event ports with time stamp. Capture live as the application is executed. Dynamic filtering. Could use to see faults and correction,
			* Code Generation for Genode OS Framework for separation kernals, Genode, with back-end pathways to Sel4 and Xen. Translation to Sel4 via CamKES, bypassing Part of DARPA CASE project with Rockwell Collins, Adventium, Data61. An evolution of Mke Whalen “Trusted Build” infrastructure. Works for CASE project demo targets,
			* Slang Embeddded infrastructure provides a reasonable mature first integration of code generation and component development for AADL.
			* Can target multiple backends.
			* Limitations – only port-based communication supported (no subprogram interactions)
			* We have a start on RT with built in timers, etc. but fuller integration with RT schedulers is needed
			* Conclusion: basic patterns of computation
			* Q - Back out the requirements and demonstrate fulfilled.
			* Q - Do you support the three port communication patterns –
			* Jerome – looks compatible, need to look in deeper. We need some in depth, so will have to have detailed conversations.
* 1430-1530: Runtime services and Code Generation Recommendations (Etienne Borde)
	+ C implementations in RAMSES
		- GetValue, nextValue, put value, get count/update, receive input/sendoutput, await dispatch, current system mode/set system mode
		- Await mode
		- Not yet implemented Set Error code/Get\_Error\_Code
		- Resolved – Dispatch Status not documented.
	+ Principle – Provide an AADL definition of the RTS: to be used in standard, used in RAMSES to refine AADL models towards code generation.
	+ Provide a reference implementation in C:
	+ RAMSES has been deeply reworked to use RTS presented
	+ Parts in RED are blocking and require decisions from the committee
		- Impact core, impact code generation annex
	+ Precisely defined RTS is required.
	+ Example documentation for standard provided - GetValue
	+ Mapping to C but also to Ada, particular code generators like OCARINA
	+ Most of the RTS are outside the use of the user, they are for the tool builder.
	+ User can use Set\_System\_Mode/Get\_System\_Mode (no need for user to control internal thread,subprogram modes)
	+ Issues related to the core AADL and Code generation annex
		- Data port -> ok if integration style is legacy
		- Event {data}port -> LIMITED if integration style is legacy
		- How to know from the source code that …
	+ Code generation annex quite outdated So which is the problem, concept or documentation?
	+ Should a subprogram parameter be part of the subprogram context data type?
		- Update core, or update Code Generation Annex, Add a rule stating .. Change request in the Code Generation Annex.
	+ Jerome - Subprogram parameters – should not be an input port. Will take some time to discuss. I take the issues. Action: Need time in next meeting.
	+ **Action:** Set up Errata for Code Generation Annex time for next meeting.
	+ Sampled, immediate and delayed communication – how is this related to runtime services. Proposal adopt the scheduling table property from the ARINC653 annex as a core standard
	+ About exceptions in the BA – RTS should return an error code.
	+ Path to follow is an iterative process: answer current issues, test, rework. It requires fixes to continuing towards the next step.
	+ Tool demos -

#

 WebEx information:

**Monday & Tuesday:**

AS-2C AADL Meeting
Every day, from Monday, November 5, 2018, to Tuesday, November 6, 2018
8:30 am | GMT Time (London, GMT) | 10 hrs 30 mins

JOIN WEBEX MEETING
<https://sae.webex.com/sae/j.php?MTID=mcfdc8face78cf2e1dc706d7fccdedcbd>
Meeting number: 622 656 709
Meeting password: Monday1

JOIN BY PHONE
1-866-469-3239 Call-in toll-free number (US/Canada)
Tap here to call (mobile phones only, hosts not supported): tel:1-866-469-3239,,\*01\*622656709%23%23\*01\*

1-650-429-3300 Call-in toll number (US/Canada)
Tap here to call (mobile phones only, hosts not supported): tel:+1-650-429-3300,,\*01\*622656709%23%23\*01\*
Access code: 622 656 709

Global call-in numbers:
<https://sae.webex.com/sae/globalcallin.php?serviceType=MC&ED=735129352&tollFree=1>

Toll-free dialing restrictions:
<https://www.webex.com/pdf/tollfree_restrictions.pdf>

Add this meeting to your calendar (Cannot add from mobile devices):
<https://sae.webex.com/sae/j.php?MTID=me11ba76c3c0f9f7870b7615646d8058c>

**Wednesday & Thursday:**

AS-2C AADL Meeting
Every day, from Wednesday, November 7, 2018, to Thursday, November 8, 2018
8:30 am | GMT Time (London, GMT) | 9 hrs 30 mins

JOIN WEBEX MEETING
<https://sae.webex.com/sae/j.php?MTID=m4dfbd665aad87e82bca566466509b03c>
Meeting number: 629 820 463
Meeting password: Wednesday1

JOIN BY PHONE
1-866-469-3239 Call-in toll-free number (US/Canada)
Tap here to call (mobile phones only, hosts not supported): tel:1-866-469-3239,,\*01\*629820463%23%23\*01\*

1-650-429-3300 Call-in toll number (US/Canada)
Tap here to call (mobile phones only, hosts not supported): tel:+1-650-429-3300,,\*01\*629820463%23%23\*01\*
Access code: 629 820 463

Global call-in numbers:
<https://sae.webex.com/sae/globalcallin.php?serviceType=MC&ED=735128652&tollFree=1>

Toll-free dialing restrictions:
<https://www.webex.com/pdf/tollfree_restrictions.pdf>

Add this meeting to your calendar (Cannot add from mobile devices):
<https://sae.webex.com/sae/j.php?MTID=m6616cce4e0414161ae653110cbea4f1c>