Transforming AADL Models Into SysML 2.0: Insights and Recommendations

Kyle Litwin[1], Isaac Amundson[1], Dinesh Verma[2], and Tom McDermott[2]
[1] Collins Aerospace
[2] Stevens Institute of Technology

Abstract

In recent years, the increasing complexity of modern aerospace systems has driven the rapid adoption of robust Model-Based Systems Engineering (MBSE). MBSE is a development methodology centered around computational models, which are instrumental in supporting the design and analysis of intricate systems. In this context, the Architecture Analysis and Design Language (AADL) and Systems Modeling Language (SysML) are two prominent modeling languages for specifying and analyzing the structure and behavior of a cyber-physical system. Both languages have their own specific use cases and tool environments and are typically employed to model different aspects of system design. Although multiple software tools are available for transforming models from one language to another, their effectiveness is limited by fundamental differences in the semantics of each language. The upcoming release of SysML Version 2 provides an opportunity to address these shortcomings thanks to several improvements that bring the two languages closer together. In this paper, we embark on an exploration of a transformation pathway between AADL and SysML v2, while identifying the existing gaps and challenges that persist. Furthermore, we provide recommendations to overcome these issues. Our approach's feasibility is demonstrated using an open-source AADL model employed in a Defense Advanced Research Projects Agency (DARPA) research project as a case study. We also outline several transformation rules required for converting the AADL model into a syntactically correct and semantically equivalent SysML v2 model.

Introduction

The need for powerful Model-Based Systems Engineering (MBSE) languages in the toolkit of a practicing systems engineer has never been greater. As complex systems continue to evolve and proliferate, it has become crucial to have a comprehensive set of tools for computational architecture and design, development, and analysis at our disposal. In this paper, we provide a preliminary exploration of the upcoming Systems Modeling Language (SysML) v2 standard, focusing on its utility and capability to represent Architecture Analysis & Design Language (AADL) models; and to assess if SysML v2 is a viable replacement to the AADL language.

SysML 1.x, a general-purpose graphical modeling language used for systems engineering, underwent a substantial overhaul, represented in version 2. The Object Management Group (OMG), which oversees the SysML standard, states that “SysML v2 enables modeling of increasingly complex systems as part of the evolving practice of model-based systems engineering. It provides improved precision, expressiveness, consistency, usability, interoperability, and extensibility over SysML v1.x” [1]. Simultaneously, AADL has emerged as a model-based, SAE standard for the analysis and design of real-time, safety-critical embedded systems, providing valuable solutions for industries such as aerospace and automotive.

Despite their separate development paths, the potential interoperability between SysML v2 and AADL bears notable significance, particularly regarding the representation of AADL models within the SysML v2 environment. Exploring the compatibilities and potential synergies between these two languages is a central goal of this research, with an eye towards consolidating modeling approaches to facilitate the design, analysis, and assurance of complex systems. SysML v2 is based on the Kernel Modeling Language (KerML), which is an application-independent modeling language with a well-grounded formal semantics for modeling existing or planned systems [2]. SysML uses KerML as its semantic base, with specific libraries modeled using domain specific SysML syntax (see Figure *1*). The expressiveness of the underlying KerML semantic specification is a key enabler for bridging the gap between AADL and SysML.

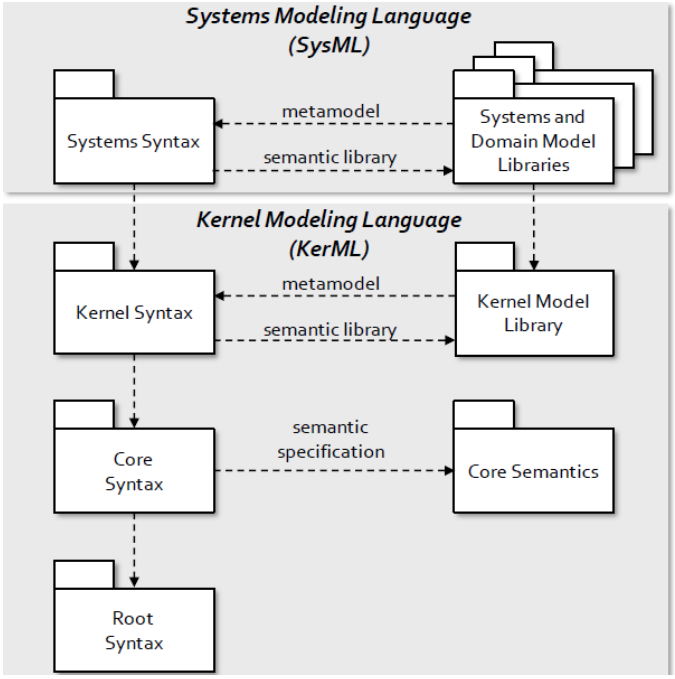

Figure 1. Relationship between SysML v2 and KerML. Reproduced from [3].

This study will delve into an analytical comparison of SysML v2 and AADL, examining how SysMLv2 can be used to equivalently represent the AADL language using a case-study AADL model and identifying any gaps or opportunities in the ability of SysML v2 to represent and analyze AADL models. The research aims to scrutinize the adequacy of SysML v2 as a representative platform for AADL models, probing into potential challenges and advantages. It is anticipated that the results of this study will inform future discussions and development around these languages, potentially guiding their

Pre-print for the following paper published by the SAE:

Litwin, K., Amundson, I., Verma, D., and McDermott, T., “Transforming AADL Models Into SysML 2.0: Insights and Recommendations,” SAE Technical Paper 2024-01-1947, 2024, doi:10.4271/2024-01-1947

future trajectory and enhancing their applicability to real-world engineering problems.

The following sections leverage an open-source AADL small Unmanned Aerial Vehicle (UAV) model, used on other research and government projects, and shows how the same model can be represented in a semantically equivalent manner using the SysML v2 syntax. Code from both the source AADL model and proposed equivalent SysML v2 will be provided with explanation as to why the SysML V2.0 model is equivalent. Finally, recommendations for widespread adoption on the use of representing AADL semantics via the SysML v2 will be provided along with areas of future research not addressed in this paper.

Background

As of July 2023, the SysML v2 standard has not yet been officially released by OMG and final approval is expected sometime in 2024 [2]. As such, existing research related to SysML v2 is limited. Our research is based upon the reference version of SysML v2 released in March 2023, Release 2023-02 [4]. To the best of our knowledge, there is no published research relating AADL and SysML v2 representations, although we are aware of several research groups currently exploring this topic. While AADL to SysML v1.x conversion tools currently exist, such as Galois' SysML to AADL Bridge Tool [5], no such tooling currently exists to support the conversion of AADL models to SysML v2.

MBSE has influenced the field of systems engineering, transforming conventional document-based methods into a model-centric approach, which facilitates improved precision, communication, and productivity. However, the current landscape of MBSE is hampered by the existence of multiple modeling languages, such as SysML, AADL, and MathWorks System Composer, which are employed across various platforms. These modeling languages, despite their respective strengths, often exhibit compatibility issues with each other, creating a significant impediment in transforming or translating models from one language to another. This issue is not merely a technical difficulty; it's also conceptual. SysML and AADL, for instance, each possess unique underlying principles, concepts, and structures, which resist straightforward translation. The resulting lack of interoperability stifles the advancement of MBSE, curtailing efficiency and escalating the probability of misinterpretation and errors during the model transformation process. Consequently, this issue underscores a critical challenge in fully capitalizing on the benefits of MBSE, particularly in contexts necessitating collaborative integrations of complex systems across diverse fields and sectors, and complex supply chain.

Efforts have been undertaken to seamlessly integrate an MBSE-process incorporating AADL and SysML to leverage the strengths of both languages. AADL being a cyber-physical systems-based modeling language and strongly typed, excels when used in the context of a formal semantics. SysML v1.x is a general-purpose MBSE language, offering broad use across multiple fields and commercial tools which have been developed over many years. However, SysML v1.x lacks a textual representation, relying entirely on graphical notation which is not easily machine readable to produce formally provable correctness. This lack of a textual notation results in tool-specific exports of SysML models which limit the ability to share models across different SysML toolsets.

This lack of textual description also presents a challenge when attempting to perform model transforms between AADL and SysML. A tenet in MBSE is the concept known as "single source of truth". As opposed to document-based design, where a system's requirements and design are distributed across several documents, in MBSE, the model is the sole source of truth. When a single system design model is represented across multiple languages, keeping the entire model in sync and consistent is a major challenge. The ability to seamlessly transform a model from one MBSE language to another represents a key capability in maintaining a self-consistent single source of truth.

Model Transformation Approach

MBSE has surfaced as a solution for managing the complexity of contemporary systems development. In this context, the transformation from AADL to SysML has gained attention due to its potential to bridge the gap between high-level architectural description and detailed system modeling. In prior research and in industry, limited support exists when it comes to performing transformations between AADL and SysML v1, particularly due to the considerable semantic differences between the two modeling languages. As SysML v2 is currently not ratified officially, no AADL to SysML v2 translations exist publicly. This study presents a first step at a methodological approach to model transformation, which encompasses the creation of mapping rules, implementation of the transformation process, and validation of the resultant SysML v2 model. The transformation model in this study is centered on a set of semantic mapping rules that align the core elements of AADL (e.g., system, process, thread, data) with corresponding SysML v2 blocks, parts, ports, and connectors. This mapping is accomplished via a combination of axiomatically declaring equivalence between certain AADL base elements and SysML v2 base types. The goal is to maintain the integrity and accuracy of the system representation during the transformation, while accounting for the more expressive and detailed design space that SysML v2 offers.

Validation of the transformation process involves both syntactic and semantic checks. Syntactic validation ensures the correct transformation of AADL elements into valid SysML v2 per the current SysML v2 standard. This is accomplished by the Eclipse-based reference implementation publicly available to automate syntax validity of the resulting SysML v2 model. On the other hand, semantic validation proposed in this paper guarantees that the underlying meaning of the AADL model is accurately represented in the SysML v2 model. This research contributes to the existing body of knowledge by addressing the gaps in AADL-SysML v2 transformations. A future goal could be to develop a fully comprehensive AADL library within SysML v2. By providing an initial library of AADL elements and constructs within the SysML v2 language, this research enhances interoperability, promotes the reuse of system models across various stages of development, and broadens the utility of MBSE in complex systems development. An end-goal of follow-on research in this area would be determining if the SysML v2 language has the sufficient comprehensive expressiveness to model all aspects of the AADL language. If so, it would allow a consolidation of two influential modeling languages into a single language, eliminating the need to develop model transformation tools, developing models within two modeling environments, and simplifying MBSE, ultimately providing less expensive and faster development of systems to customers.

This research started with an open-source AADL model of a small Unmanned Aerial Vehicle (UAV) [6]. This AADL model was originally developed on the Defense Advanced Research Projects Agency (DARPA) Cyber Assured Systems Engineering (CASE) program. The CASE program's goal was to "develop the necessary design, analysis and verification tools to allow system engineers to design-in cyber resiliency and manage tradeoffs as they do other nonfunctional properties when designing complex embedded computing systems" [7]. The UAV model was chosen as it was of non-trivial complexity (and thus representative of real-world AADL models), publicly available, and has been vetted previously for research purposes.

The AADL UAV system is modeled hierarchically, with the UAV package file representing the top-level of the UAV (Figure 2).

```
package UAV
public
    with MC;
    with FC;
    with UAS_Buses;
    -- UAV
system UAV
    features
        RadioRecv: in event data port;
        RadioSend: out event data port;
        RFA: requires bus access UAS_Buses::RF_Bus.Impl;
end UAV;

system implementation UAV.Impl
    subcomponents
        MissionComputer: system MC::MissionComputer.Impl;
        FlightControlComputer: system FC::FlightController.Impl;
        SerialBus: bus UAS_Buses::Serial_Bus.Impl;
    connections
        bac1: bus access SerialBus <-> MissionComputer.UARTA;
        bac2: bus access SerialBus <-> FlightControlComputer.UARTA;
        bac3: bus access RFA -> MissionComputer.RFA;
        c1: port RadioRecv -> MissionComputer.RadioRecv;
        c2: port MissionComputer.RadioSend -> RadioSend;
        c3: port MissionComputer.UartSend -> FlightControlComputer.UartRecv;
        c4: port FlightControlComputer.UartSend -> MissionComputer.UartRecv;
    properties
        Actual_Connection_Binding => (reference (SerialBus)) applies to c3, c4;
end UAV.Impl;

end UAV;
```

Figure 2. Top-level AADL UAV model representation.

Before representing this model in SysML v2, AADL semantically equivalent classifiers were created in the SysML v2 language. For example, a *system* in AADL is a composite element that can encapsulate software and hardware subcomponents into a distinct unit within an architecture [8]. An analogous element, a *part*, exists within SysML v2. A part in SysML v2 is the definition of a class of system or part of a system which is mutable and exists in time and space [3]. Parts, like systems in AADL, can contain sub parts (subcomponents in AADL), allowing a hierarchical definition of a system. However, to create an AADL system equivalent classifier within the SysML v2 language, we defined an explicit part to represent an AADL system – *AADLSystem*. Other AADL base components were similarly defined in this research. We define the SysML v2 declarations in Figure 3 to be semantically equivalent to their corresponding AADL definitions – that is, an *AADLSystem* in SysML v2 represents and behaves identically to an AADL defined system component.

```
library package AADL_Base_Classes{
    import AADL_Port_Actions::*;

    abstract part def AADLSystem; // AADL 'System'
    abstract part def AADLProcess; // AADL 'Process'
    abstract part def AADLThread; // AADL 'Thread'
    abstract port def AADLPort; // Generic AADL 'Port'

    port def Data specializes AADLPort;  // AADL 'Data port'
    port def Event specializes AADLPort; // AADL 'Event port'
    port def EventData specializes AADLPort; // AADL 'Event Data port'

    // Equivalent to an AADL 'Bus Access' feature
    abstract port def Bus_access specializes AADLPort;

    // Further refinement of an AADL 'in event data port'
    port def in_event_data_port specializes EventData{
        in item event_data;
        // defining the behavior action of an AADL event data port.
        action AADL_Event_Data_Port_Action_CAmkES;
    }
    // Further refinement of an AADL 'out event data port'
    port def out_event_data_port specializes EventData{
        out item event_data;
        action AADL_Out_Event_Data_Port_Action;
    }
    // Equivalent to an AADL 'Bus'
    abstract interface def AADL_Bus{
        end Start: AADLPort[*]{
        }
        end End: AADLPort[*]{
        }
    }
}
```

Figure 3. SysML v2 declarations of equivalent AADL components.

In addition to declaring equivalent AADL component types within the SysML v2 language, this research proposes using *metadata* definitions in the SysML v2 model as a semantically equivalent representation of *property sets* used in the UAV AADL model shown in Figure 4 and Figure 5Figure 4

```
Base_Type : list of classifier ( data ) applies to
( data, feature );
--  The Base_Type property specifies the base type of a data
--  component type. The classifiers being referenced are those
--  defined in the Base_Types package or from user defined
--  packages.
Dimension : list of aadlinteger applies to
( data, data port, event data port, data access );
--  The Dimension property is used to specify the dimensions of a
--  multi-dimensional array, ordered.  This property shall be used
--  in conjunction with the Data Representation property.
Number_Representation : enumeration (Signed, Unsigned)
applies to ( data, feature );
--  Number_Representation specifies whether an integer data
--  component is signed or unsigned.
Data_Representation : enumeration
  (Array, Boolean, Character, Enum, Float,
   Fixed, Integer, String, Struct, Union)
   applies to ( data, feature );
--  The Data_Representation property may be used to specify the
--  representation of simple or composite data types within the
--  programming language source code.
--  Note: An implementation is allowed to support only a subset of
--  these structures.
```

Figure 4. AADL property sets.

```
library package AADL_Data_Model{
    import AADL_Data_Types::*;
    private import ScalarValues::*;

    metadata def Base_Type_Property_Set{
        attribute Base_Type;
        //Corresponds to the "applies to" of AADL property (data)
        :>annotatedElement : SysML::AttributeDefinition;
        //Corresponds to the "applies to" of AADL property (feature)
        :>annotatedElement : SysML::Feature;
    }
    metadata def Data_Representation_Property_Set{
        attribute Data_Representation_Property : Data_Representation;
        :>annotatedElement : SysML::AttributeDefinition;
        :>annotatedElement : SysML::Feature;
    }
    metadata def Dimension_Property_Set{
        attribute Dimension_Property :  Integer;
        :>annotatedElement : SysML::AttributeDefinition;
        :>annotatedElement : SysML::PortDefinition;
    }
    metadata def Number_Representation_Property_Set{
        enum Number_Representation_Property : Number_Representation;
        :>annotatedElement : SysML::AttributeDefinition;
        :>annotatedElement : SysML::Feature;
    }
    enum def Number_Representation{
        enum Signed;
        enum Unsigned;
    }
    enum def Data_Representation{
    enum Array;
    enum Boolean;
    enum Character;
    enum Enum;
    enum Float;
    enum Fixed;
    enum Integer;
    enum String;
    enum Struct;
    enum Union;
    }
}
```

Figure 5. Proposed equivalent SysML v2 metadata definitions.

Data types defined in the UAV AADL model (Figure 6) were also defined in the SysML v2 model (Figure 7).

```
package Data_Types
public
    with Data_Model;
    with Base_Types;

    data KeepInZone
        properties
            Data_Model::Data_Representation => Array;
            Data_Model::Base_Type => (classifier (Coordinate.Impl));
            Data_Model::Dimension => (2);
    end KeepInZone;

    data implementation KeepInZone.Impl
    end KeepInZone.Impl;

    data Coordinate
    end Coordinate;
-- This structure holds the lat/long/alt values of a coordinate
    data implementation Coordinate.Impl
        subcomponents
            Latitude: data Base_Types::Float_32;
            Longitude: data Base_Types::Float_32;
            Altitude: data Base_Types::Float_32;
    end Coordinate.Impl;
-- This structure contains 2 coordinates on either side of a region.
    data Map
        properties
            Data_Model::Data_Representation => Array;
            Data_Model::Base_Type => (classifier (Coordinate.Impl));
            Data_Model::Dimension => (2);
    end Map;
```

Figure 6. AADL data types.

```
package AADL_Data_Types{
    import AADL_Data_Model::*;
    import Base_Types::*;
    import SI::*;

    attribute def KeepInZone {
        @AADL_Data_Model::Data_Representation_Property_Set
        {Data_Representation_Property = Data_Representation::Array;}
        @AADL_Data_Model::Base_Type_Property_Set {Base_Type : Coordinate;}
        @AADL_Data_Model::Dimension_Property_Set {Dimension_Property = 2;}

    }
    attribute def Coordinate{
        attribute latitude : Base_Types::Float_32;
        attribute longitude : Base_Types::Float_32;
        attribute altitude : Base_Types::Float_32;
        }

    attribute def Map{
        @AADL_Data_Model::Data_Representation_Property_Set
        {Data_Representation_Property = Data_Representation::Array;}
        @AADL_Data_Model::Base_Type_Property_Set {Base_Type : Coordinate;}
        @AADL_Data_Model::Dimension_Property_Set {Dimension_Property = 2;}
    }

    attribute def Command{
        attribute Map_Cmd : Map;
        // AADL allows a variable to be named the same as the type name.
        // E.g. "Map: data Map;"
        // SysMLv2 does not. Renamed attribute name to Map_Cmd
    }
```

Figure 7. Proposed equivalent SysML v2 data type definitions.

Another initial area of investigation was modeling the semantic behavior of AADL components within the SysML v2 language, focusing on AADL ports. AADL defines three types of ports: *data*, *event*, and *event data*. Data ports are used to represent sampling ports and contain only the most recent arrival (i.e., no queue). Event ports are data ports without data content and represent discrete event changes like a switch or clock interrupt. Event data ports represent asynchronous communications that may be queued [8]. SysML v2 allows the representation of an action (in this case, the behavior of a port) by using textual representation of a language other than SysML. Using the example UAV model and associated documentation, the behavior of an AADL event data port was described using the CAmkES language [6] and is used in the declaration of event data ports shown above in Figure 3.

```
// Example using the "Opaque" feature of SysML
// to define a textual representation using a language other than SysML
// Following was used from the "AADL Modeling Guidelines for CASE"
// located here: https://github.com/loonwerks/BriefCASE-Tutorial
action def AADL_Event_Data_Port_Action_CAmkES{
    in attribute Event_Data;
    language "CAmkES"
    /*
     *  import <std_connector.camkes>;
     *  import "components/emitter_t_impl/emitter_t_impl.camkes";
     *  import "components/consumer_t_impl/consumer_t_impl.camkes";
     *  assembly {
     *      composition {
     *          component emitter_t_impl src;
     *          component consumer_t_impl dest;
     *          connection seL4Notification
     *              conn1( from src.sb_enq_1_notification,
     *              to dest.sb_deq_notification);
     *          connection seL4SharedData
     *              conn2( from src.sb_enq_queue_1, to dest.sb_deq_queue);
     *      }
     *      configuration {
     *          src.sb_enq_queue_1_access = "W";
     *          dest.sb_deq_queue_access = "R";
     *      }
     *
     *  }
     */
}
```

Figure 8. SysML v2 definition of AADL event data port using CAmkES.

Once the above foundational elements between AADL were mapped to a semantically equivalent representation in SysML v2, the top-level UAV model in AADL (Figure 2) was translated into an equivalent model (based on our class definitions) in SysML v2 (Figure 9).

```
package UAV {

public import MC::*;
public import FC::*;
public import UAS_Busses::*;
public import AADL_Base_Classes::*;

    //Top level Black box definition
    part def UAV specializes AADLSystem{
        port RadioRecv : in_event_data_port;
        port RadioSend : out_event_data_port;
        port RFA : Bus_access;
    }
    //Top level white box definition, includes subcomponents
    part def UAV_implementation specializes UAV {
        part MissionComputer : MC_Implementation;
        part FlightControlComputer : FC_Implementation;

        // Interface Definition Part. Interface is a
        // specialization of a SysML part.
        // This definition includes the corresponding
        // bus access connections 'bac1' and 'bac2' in the AADL model

        //Corresponds to 'SerialBus'' Component in AADL
        interface def SerialBus specializes Serial_Bus{
            end FlightControlComptuer : AADLPort{
                port UartRecv: in_event_data_port;
                port UartSend: out_event_data_port;
            }
            end MissionComputer : AADLPort{
                port UartRecv: in_event_data_port;
                port UartSend: out_event_data_port;
            }
        }
        //Bus Connections
        //Corresponds to connection 'bac3' in AADL reference model
        connection bac3 connect RFA to MissionComputer.RFA;
        //Port Connections
        //Corresponds to connection 'c1' in AADL reference model
        connection c1 connect RadioRecv to MissionComputer.RadioRecv;
        //Corresponds to connection 'c2' in AADL reference model
        connection c2 connect MissionComputer.RadioSend to RadioSend;

        // Bus Interface Connections semantically include the meaning of the
        // "Actual_Connection_Binding" of the AADL syntax connections
        // c3 and c4 to the Serial bus in the AADL reference model.
        // The interface "SerialBus" corresponds to the 'SerialBus'
        // subcomponent of the AADL Reference model.

        //Corresponds to connection 'c3' in AADL reference model
        interface c3 : SerialBus connect MissionComputer.UartSend
        to FlightControlComputer.UartRecv;
        //Corresponds to connection 'c4' in AADL reference model
        interface c4 : SerialBus connect FlightControlComputer.UartSend
        to MissionComputer.UartRecv;
    }

}
```

Figure 9. Top-level SysML v2 UAV model representation.

Conclusions and Future Research

The effectiveness of the SysML v2 model presented above in representing the AADL UAV model largely depends on the interpretation of the SysML model semantics. For example, the *SerialBus* component in the AADL model is a type of *bus*. In the SysML v2 model, a potentially semantically equivalent part, *Serial_Bus* was defined which is of type *interface*. In AADL, the concept of a bus requires a physical hardware bus to support communication protocols across physical components. *Bus access* connections declare the physical connection itself [6]. In the UAV AADL model, this is accomplished with the code block from the top-level AADL model (Figure 10). We interpret this to be semantically equivalent to the SysML code in Figure 11, in which the interface definition *SerialBus* is of type *AADL_Bus* defined within the SysML v2 project. Therefore, *SerialBus* defined in SysML v2 includes the equivalent semantic meaning as the AADL bus access connections "*bac1*" and "*bac2*".

```
subcomponents
    MissionComputer: system MC::MissionComputer.Impl;
    FlightControlComputer: system FC::FlightController.Impl;
    SerialBus: bus UAS_Buses::Serial_Bus.Impl;
connections
    bac1: bus access SerialBus <-> MissionComputer.UARTA;
    bac2: bus access SerialBus <-> FlightControlComputer.UARTA;
```

Figure 10. AADL bus connections.

```
    // Equivalent to an AADL 'Bus'
    abstract interface def AADL_Bus{
        end Start: AADLPort[*]{
        }
        end End: AADLPort[*]{
        }
    }
}
part def Serial_Bus specializes AADL_Bus{
}
//Corresponds to 'SerialBus'' Component in AADL
interface def SerialBus specializes Serial_Bus{
    end FlightControlComptuer : AADLPort{
        port UartRecv: in_event_data_port;
        port UartSend: out_event_data_port;
    }
    end MissionComputer : AADLPort{
        port UartRecv: in_event_data_port;
        port UartSend: out_event_data_port;
    }
}
```

Figure 11. Proposed SysMLv2 bus connections.

Finally, the logical port connections, which describe data being sent and received, are defined, and "bound", that is, explicitly mapped to a hardware bus (in this case *SerialBus)* using properties. We therefore propose interpreting the AADL and SysML code in Figure 12 and Figure 13 as semantically equivalent.

```
    c3: port MissionComputer.UartSend ->
        FlightControlComputer.UartRecv;
    c4: port FlightControlComputer.UartSend ->
        MissionComputer.UartRecv;
properties
    Actual_Connection_Binding =>
        (reference (SerialBus)) applies to c3, c4;
```

Figure 12. AADL connections.

```
//Corresponds to connection 'c3' in AADL reference model
interface c3 : SerialBus connect MissionComputer.UartSend
to FlightControlComputer.UartRecv;
//Corresponds to connection 'c4' in AADL reference model
interface c4 : SerialBus connect FlightControlComputer.UartSend
to MissionComputer.UartRecv;
```

Figure 13. Proposed semantically equivalent SysML v2.

A syntactically correct SysML v2 model was developed without syntax errors in an Eclipse-based environment. However, achieving semantic alignment with AADL remains a challenge, as SysML v2 does not inherently encompass AADL semantics, except for specific elements like the event data port defined in CAmkES within SysML. To facilitate model sharing across organizations, a consensus on a common AADL library within SysML v2 projects is crucial, along with the development of automated tooling for AADL-equivalent architecture analysis within SysML v2 environments. This research focused on a limited subset of AADL, successfully mapping AADL classifiers and components in a UAV model to SysML v2. However, it did not address many AADL constructs such as flows, modes, virtual components, and subprograms. Future research is necessary to determine whether SysML v2, in its current state, can fully express the entirety of AADL semantics. While effective in this study for representing the UAV AADL model, SysML v2's capacity to natively express all AADL aspects remains to be verified. This study represents an initial step in evaluating the syntactic and semantic expressiveness of SysML v2 and KerML, underscoring the need for continued research and collaboration between the academic and

industrial sectors. Such efforts are essential to develop a comprehensive metamodel that facilitates interoperability between SysML v2 and other system modeling languages, including AADL, thus creating a Lingua Franca for broader model interoperability.

Contact Information

Kyle Litwin
e-mail: Kyle.litwin@collins.com

Acknowledgments

This work was inspired in part by the independent efforts of Jérôme Hugues (Software Engineering Institute at Carnegie Mellon University) and John Hatcliff (Kansas State University) to develop an AADL library for SysML v2.

References

1. Object Management Group. n.d. *2023 Sysml V2.0 Overview*. https://www.omg.org/events/2023Q2/special-events/SysML-Session.htm.
2. Object Management Group (OMG). n.d. "Kernel Modeling Language." https://www.omg.org/spec/KerML.
3. Model Driven Solutions, Inc. Release 2023-10. "Introduction to the SysML v2 Language Textual Notation." https://github.com/Systems-Modeling/SysML-v2-Release/blob/master/doc/Intro%20to%20the%20SysML%20v2%20Language-Textual%20Notation.pdf. Licensed under Creative Commons Attribution 4. 0 International License http://creativecommons.org/licenses/by/4.0/
4. Object Management Group, Inc. (OMG). 2023. "OMG Systems Modeling Language (SysML)." no. Release 2023-02. March 2023. https://github.com/Systems-Modeling/SysML-v2-Release/blob/c4d8aebd12929bd258051341baefd95e60e93e56/doc/2-OMG_Systems_Modeling_Language.pdf.
5. Galois. n.d. *CAMET Tools.* https://camet-library.com/camet-tools.
6. Collins Aerospace. 2022. "BriefCASE-Tutorial." October 3. https://github.com/loonwerks/BriefCASE-Tutorial/tree/main/Initial.
7. DARPA. n.d. *Cyber Assured Systems Engineering (CASE).* https://www.darpa.mil/program/cyber-assured-systems-engineering.
8. Feiler, Peter, and David Gluch. 2013. *Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analsys & Design Language.* Upper Saddle River, New Jersey: Pearson Education Inc