The systems architecture ontology (SAO): an ontology-based design method for cyber–physical systems

Diego Camara Sales
Automation, Federal Institute of Education Science and Technology Amazon, Manaus, Brazil
Leandro Buss Becker
Automation and Systems, Federal University of Santa Catarina, Florianopolis, Brazil, and
Cristian Koliver
Information Systems, Federal University of Santa Catarina, Florianopolis, Brazil

Received 14 September 2021
Revised 13 November 2021
16 December 2021
Accepted 26 December 2021

Abstract

Purpose – Managing components' resources plays a critical role in the success of systems' architectures designed for cyber–physical systems (CPS). Performing the selection of candidate components to pursue a specific application's needs also involves identifying the relationships among architectural components, the network and the physical process, as the system characteristics and properties are related.

Design/methodology/approach – Using a Model-Driven Engineering (MDE) approach is a valuable asset therefore. Within this context, the authors present the so-called Systems Architecture Ontology (SAO), which allows the representation of a system architecture (SA), as well as the relationships, characteristics and properties of a CPS application.

Findings – SAO uses a common vocabulary inspired by the Architecture Analysis and Design Language (AADL) standard. To demonstrate SAO's applicability, this paper presents its use as an MDE approach combined with ontology-based modeling through the Ontology Web Language (OWL). From OWL models based on SAO, the authors propose a model transformation tool to extract data related to architectural modeling in AADL code, allowing the creation of a components' library and a property set model. Besides saving design time by automatically generating many lines of code, such code is less error-prone, that is, without inconsistencies.

Originality/value – To illustrate the proposal, the authors present a case study in the aerospace domain with the application of SAO and its transformation tool. As result, a library containing 74 components and a related set of properties are automatically generated to support architectural design and evaluation.

Keywords System architecture, Ontology-based model design, AADL model Transformation

Paper type Research paper

1. Introduction

A systems' architecture (SA) model is devoted to describing the structure, behavior and views of a given system [1], where the architectural description offers a representation of the components, features and properties of the systems, which could represent an SA. However, managing the components within cyber–physical systems (CPS) that make intensive use of sensors and actuators, probably requiring modifications along its life cycle, is a laborious and error-prone task. Tasks such as identifying the relationships between components,

Applied Computing and Informatics
Vol. 21 No. 3/4, 2025
pp. 259-274
Emerald Publishing Limited
e-ISSN: 2210-8327
p-ISSN: 2634-1964
DOI 10.1108/ACI-09-2021-0249

subsystems and other model elements affected by eventual component modifications need to be properly addressed.

Typically, the architectures of CPS are modeled using architecture description languages (ADLs) such as the Systems Modeling Language (SysML) [2] and the Architecture Analysis and Design Language (AADL) [3], which have been used for the specification, analysis and behavior evaluation of complex systems. AADL allows constructing a lower-level system view, combining hardware and software components and covering real-time and other system's properties [4]. SysML provides conceptual representation, systems vision, static and behavioral system representation, as well as requirements and parametric diagrams, but, in contrast with AADL, it does not cover details about software entities, hardware components and detailed properties specification (such as the real-time ones).

AADL is one of the most widespread languages given its capability to model embedded systems and perform analyses from the models. It also includes facilities for developers to create plug-ins for expanding the analyses' capacity. However, it was noticed that, in complex projects, it is difficult to start the development of an SA by writing AADL code without understanding the physical process relationship with the architectural components. Therefore, a design solution using a higher level of abstraction was explored in this work.

In computer science, ontology is a means of highlighting the features, characteristics, attributes and properties or parameters of a subject/area and showing how they are related, by defining a set of concepts and categories [5]. Using an ontology in the initial phase of CPS design helps designers to represent the "system-under-construction" in a more formal manner, thereby allowing the evaluation of semantic (in-)consistencies through the ontological domain properties, such as time constraints, security and performance [6], inference to assess inconsistencies using logical deduction and design patterns [7], viewpoints integration in the CPS development [8].

An ontology dedicated to SA allows the modeling of concepts related to the composition/structure of the architecture, software and hardware components and its features. This brings into light details about the interaction of the SA with the physical process providing a knowledge base that can be used in multiple areas and representing different views of the system. Thereby, computational tools can be used to assess the existence of inconsistencies in the design of the SA, reuse the knowledge base from domain and applications, providing data to modeling CPSs.

However, declaring the ontology entities to represent the SA in such a way that it can be aligned with the Model-Driven Engineering (MDE) model-to-model (M2M) transformation process and common vocabulary is not an easy task. Thereby, it is crucial to solve or mitigate the aforementioned problems, and ADLs can provide a knowledge base about components and relationships. ADLs are a linguistic approach to the formal representation of architectures that is widespread in the scientific and systems engineering community. Therefore, ADLs can be used in a reverse engineering process, providing a knowledge base for the creation of an ontology to represent components within properties and characteristics related to the SA domain.

The main contributions of this paper can be summarized as follows. Firstly, it presents the Systems Architecture Ontology (SAO) to represent hardware, software, system components and their relationships in order to support engineers in detailing the SA and identifying inconsistencies in architectural design and component compatibility. SAO design is based on the AADL, which provides terminologies, vocabulary and concepts that can be reused. A second contribution is a plug-in to the Open Source AADL Tool Environment 2 (OSATE2), which transforms an Ontology Web Language (OWL) model into an AADL model. The OWL model, with an SAO description, is used as a source file in the model transformation process where the data are extracted to create a set of AADL target models. As benefit from such

proposal, it allows saving design time by automatically generating many lines of code, also with the understanding that such code is less error-prone, that is, without inconsistencies.

The remainder of this paper is organized as follows. Section 2 presents relevant related works. Section 3 details the proposed SA ontology, also discussing the applied methodology to build it. Section 4 describes the process to transform an OWL model into an AADL model and the tool set designed for this. Section 5 shows an unmanned aerial vehicle (UAV) domain ontology example of alignment with SAO and the respective model transformations. Section 6 presents our conclusions and perspectives on future work.

2. Related works

Different authors and organizations have proposed ontologies to represent areas and applications that somehow integrate the components of a system's architecture. For example, the IoT-Lite ontology represents Internet of Things (IoT) resources and services [9], covering platform representation and communication. The IEEE-RAS ontology standard was proposed to represent autonomous robots (ROA) [10]. The World Wide Web Consortium (W3C) proposed two ontologies related to sensor networks: the Semantic Sensor Network (SSN) and the Sensor, Observation, Sample and Actuator (SOSA) [11].

However, the aforementioned ontologies do not cover component-related topics in totality, lacking information regarding their implementation, software connection, bus communication and resource allocation, from software to hardware, which is necessary to represent an SA design, highlighting the need to create a new ontology. There are many methodologies devoted to building an ontology [12–15], and many encourage the reuse of modules and ontologies seeking to maintain a standard, common vocabulary and other benefits cited by W3C [11].

Regarding the practical application of ontologies in the design of CPS, some works evaluate consistency during integration, formalizing the interrelationships between the different views. In [6, 16], it is tackled inconsistency in the context of different design processes. Such issue is also covered in [17], which provides reasoning techniques that allow achieving greater understanding of CPS.

Regarding CPS architecture design, MDE approaches and ADLs are used during activities or steps aiming to represent SA. ADL helps to promote mutual communication and allow the early analysis and feasibility testing of architectural design decisions [18]. Thus, developing an ontology for SA development based on ADL helps to represent common concepts, aligning vocabularies and knowledge.

Given the limited expression power of the existing languages to model ontologies, the W3C released, in 2004, a new language called Web Ontology Language (OWL). OWL is a semantic web language to represent relationships and interactions among entities and groups of entities based on a Resource Description Framework (RDF) that describes the conceptual structure of OWL ontologies. OWL 2 extends OWL with a small but useful set of features that have been requested by users, for which effective reasoning algorithms are now available, and which OWL tool developers are willing to support [19]. Furthermore, such a structural specification of OWL 2 provides the foundation for the implementation of tools such as "reasoners," so that other facts can be inferred that are implicitly contained in the ontology.

OWL 2 is a general-purpose modeling language not tied to a specific domain, such as UML and SysML. Usually, in the scientific community, system requirements modeled on UML and SysML sources are used in an OWL transformation model [20–23]. This is due to the similarity of the structure of languages that are based on RDF metamodels and that can be used in the creation of rules, mappings and extensions or additional elements to become OWL ontology.

Differently from OWL, UML and SysML, domain-specific modeling languages (DSMLs) cover a range of abstraction levels for a particular domain and support conservation and reuse. DSMLs (e.g. AADL) often support automatic source code generation from DSML models.

For instance, AADL is a DSML from Society of Automotive Engineers (SAE) [24] that is intended to offer a unified framework for model-based software systems engineering. AADL can capture static structure and dynamics in a single architecture model and annotates it with information that is relevant to the analyses of characteristics.

AADL models represent SA as a hierarchy of interacting components. In total, AADL provides ten *component categories* to define software, hardware and composite system components. Software component categories are data, subprogram, thread, thread group and process. Hardware component categories are memory, processor, bus and device. Lastly, the system component is composed of hardware, software or both hardware and software components. In addition, AADL supports the early prediction and analyses of capabilities and operational quality attributes (such as performance, reliability and security) [3] in SA projects.

Behjati *et al.* [25] provide an overview of an AADL model, representing its main concepts and their relationships, based on the AADL reference manual [3]. These core concepts were elaborated as metamodel abstractions describing a system in terms of their components, interfaces and connectors between the interfaces. Metamodels support the model transformation process, aiding the exchange of OWL data and representation of SA through AADL.

Other MDE-based designs proposed the transformation of models, such as UML-MARTE [26] SysML [4] and SIMULINK [27], to AADL with a focus on data having the same representation in architecture and consistency. In the present work, we show an approach that serves to transform OWL models into AADL models. The transformation process follows the principles of MDE and requires metamodels of source and target models.

3. Adopted design methodology

This section describes the SAO design, detailing the adopted steps that supported its development and defining the entities that make up our proposal. Such a proposal was inspired by the ontology design methodology guide presented in [15]. It consists of seven steps used in the development of SAO, as further described.

Step 1 focuses on determining the domain and scope of the ontology. Considering the knowledge base provided by ADLs that provide the structure and detailed description of the components and relationships, we define the main nomenclature, concepts and vocabulary used in SAO design. Thereby, starting with a model representing the SA in the AADL language, we define the initial concepts from SA hardware components (devices, processors, memory and buses), software (process and threads) and their interaction (connections and binding).

Step 2 targets the reuse of existing SA ontologies to build a wider SA ontology. Reuse enables object-oriented design, providing the flexibility to structure the ontology in a target domain by adapting the desired module. In this case, we reused the SOSA and SSN with System Capability Module (SCM) ontologies to extend S&A representation with Quantity, Unit, Dimensions and data Types ontology (QUDT), which can be applied to the properties of SA [28].

In Step 3, we integrated the selected ontologies by aligning the modules, entities, terminologies and vocabulary to the outline of the target ontology. Considering that the SA ontology has reference to AADL, its components compose the ontology classes that guide the mapping and alignment.

Afterward, in Step 4, we declared the SAO classes based on the AADL models' structure using the AS5506C standard [24] and reusing imported classes. SAO classes represent the hardware, software and system components of an SA. An important premise when designing SAO is to reuse ontologies aligned with the SAO's domain and its components. In Figure 1, SAO classes are presented. The sao:Sections class represents the contents of sao:ComponentType and sao:ComponentImplementation and includes a set of sao:Features and sao:Properties descriptions. The sao:Features class represents the interface of components. The sao:Properties class represents the characteristics of components and has a relationship with the reuse sao:SystemCapabilityModule class. Another class is sao:Connections, which represents the interconnection between components. sao:Annex, sao:Modes, sao:Flows, sao:Prototypes and sao:extends are classes that support the representation of the content of component types and implementation, which will be presented later. The sao:SOSA class represents the reuse of sensor- and actuator-imported classes that need to be aligned.

With ssn-scm:SystemCapabilityModule, classes are imported from the SSN ontology to represent a set of properties that describe operations, conditions and system properties that have a relationship with the sao:Properties class. Extending the SCM, we include the *ComponentProperty* subclass, which describes additional characteristics from S&A that cover hardware, software and architecture design presented in a previous work (omitted due blind review process).

In Step 5, we defined the objects' properties. From the SAO classes and AADL metamodel structure of relationships, we defined a set of object properties to represent the relationship

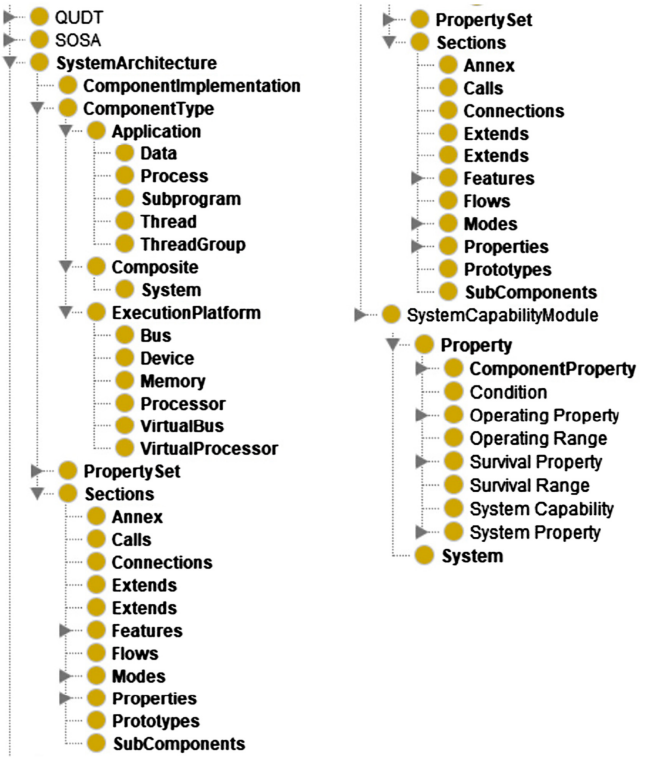

Figure 1.
SAO classes, with reuse, modeled in Protegé tool

between the classes of component, implementation and description of content, as shown in Table 1.

According to the ADL model, the components of the architecture can be defined as types and modeled in a structure of sections that can be aggregated as needed by the designer. Each section seeks to detail the specific characteristics and properties reserved for the modeled component. Here, AADL indicates that the component type can be represented by seven sections, where the classes sao:Application, sao:Composite and sao:ExecutionPlatform and subclasses can be represented with this structure. These sections are shown in the order in which they are declared. Features, in which interfaces of a component are declared, such as ports, and require bus access, reflect data traffic. Flow allows the representation of the flow through a component without exposing the component's implementation. The sao:Modes section allows operational modes of the component. The sao:Prototypes section allows the use of the top parameters the component type as a template. sao:Extends refines the component type template, and the sao:Properties section is where values are declared for properties associated with a component.

The component implementation has sections that are intended to declare how its structure, interactions and interconnections are organized. Thus, sao:Sections, similar to component types such as properties and attachments, has a scope aimed at describing details at the level of implementation and individual of the sao:Subcomponents that comprise it.

In Step 6, we defined the data and value type used in the ontology to represent some kind of data. In this step, the data properties assigned to the data representation classes (domain) and range (type of data) are declared. Usually, data types and values are modeled using a schema definition language to support a wide variety of data representation (i.e. XML Schema Definition language – XSD).

Data and data types are declared following a proposed structure based on AADL. With this structure, there is less effort when aligning the OWL and AADL languages, supporting the model transformation process; an example is shown in Figure 2. The data structure is as follows: declaration of the name of the data property – in this case, the mass measurement unit qudt:Mass; declaration of the type of class to which it belongs, for example, mass is a subclass of qudt: Weight; declaration of the measurement unit, reusing the QUDT ontology that provides the unit that we will use in this measurement represented with the terminology (g); declaration of the maximum (ValueMax) and minimum (ValueMin) allowed values and data type – considering the application, the components must have a mass less than 2000 g and are represented in integer data type:xsd:int. In total, 18 types of data were declared, which can be added in the future depending on the application of the SA and its components.

Lastly, in Step 7, we created a set of individuals to represent the materialization of the components of a CPS architecture in a specific application, which can be used to provide data for models in life cycle projects.

Individuals can represent a generic or precise description of classes. Generic individuals represent some description not related to components but related to the capability, comments or descriptions of the system (e.g. colors, unit of measurement and words). Precise descriptions represent a component type, component implementation, property set, capability and sections, which detail technical specifications. For components and implementation types, we have a set of related classes for representation. Thus, the individual can contain, through object properties assertion, a set of individuals from sections and capabilities types to describe their features and properties.

By completing the steps, the ontology is consolidated and provides a knowledge base for the semantic evaluation of an individual architecture, a physical process that represents the environment and control plant, measurement units and the network used in a specific domain.

It is important to highlight that SAO can be reused in other domains to support the representation of the architecture. For example, the aerospace domain has an area-specific

SA object property					
hasDataType	hasBussAccess	hasDeviceProperties	hasCalls	hasFeatures	
belongsToCategory	Binding_Component	hasComponentProperty	hasConnections	isComponentType	
ActualConnectionBinding	hasApplicationSwProperties	hasDeviceComponent	hasBussAccessConnection	hasAnnex	
hasCompositeProperties	hasBuss	isDeviceComponent	hasProcessAllocation	hasPrototypes	
hasConnection	hasData	requireBus	hasExtends	isSectionOf	
hasDevice	hasExecutionPlatform	isSubComponent	hasFlows	hasProperties	
hasExecutionPlatformProperties	hasPort	isPartOf	hasModes	hasThreadGroup	
hasProcess	hasProcessImplementation	isImplementedBy	isImplementationOf	hasThread	
hasProcessor	hasProperty	Implements	hasBusAllocation	hasSubComponents	

Table 1.
SAO object properties

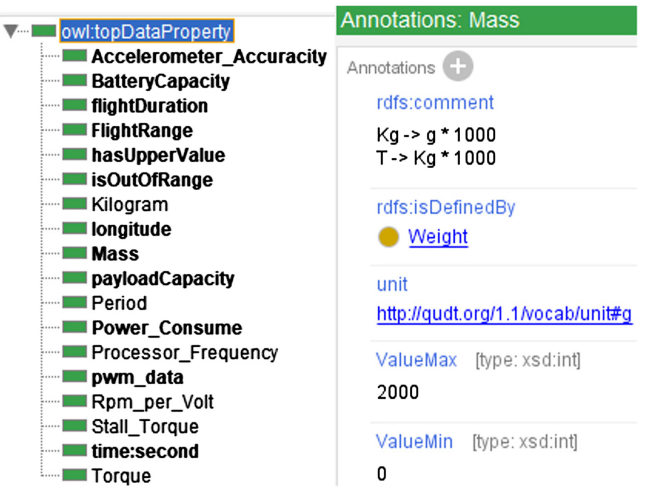

Figure 2.
SA data property example

vocabulary to represent UAVs and can use SAO to represent SA. Similarly, automotive application ontologies can benefit from SAO due to AADL-based component representation that is aligned with the SAE standard.

Ontologies related to SA, that is, representing components of the architecture, can be aligned to SAO in order to expand the representation of the system. Because SAO is based on AADL, OWL models can be used as input in the model transformation process to generate complementary architectural models in AADL to analyze the system's capability.

4. OWL model transformation to AADL

From the OWL domain model used as the source in the transformation engine, a set of activities are performed for identifying the architecture-related individuals, which lead in the generation of an AADL model package from OWL entities, data and properties that are related to the SA. This requires the mapping and creation of a set of rules and declarations that support the transformation engine presented in this section.

The generated AADL package contains three files, so that it is in accordance with the good practices to create AADL models [29], supporting a modular architecture design. Such files aim to represent: (1) the set of properties that compose measurements and units used by SA; (2) the architecture of the implemented system; and (3) the components library.

The OWL2AADL plugin was developed within this work to support this process. It is available on GitHub [1].

The transformation process requires the *OWL Model* source to conform to *SAO and reuse structure* and RDF/XML syntax. Based on the MDE method of M2M transformation, the OWL and the AADL metamodels are used to provide data structure and to create a set of specifications and rules.

In Figure 3, we present the transformation engine flow. The *Transformation Specification* block defines the mapping and relationships from OWL entities using *SAO and reuse structure* and the *AADL structure model* to process a set of *Transformation Rules* to generate an AADL model as output.

In the *Transformation Specification* block, OWL entities are defined based on the OWL2 metamodel [19], mapping classes, object properties and data related to the structure of the AADL model and its elements. The *Transformation Rules* block uses the data provided by

the transformation specification block to create a set of rules to process the OWL model input and create an AADL model output file.

Transformation Rules process OWL entities with relationships with AADL components through mapping, as shown in Table 2. The OWL Class is a component related to SA, sections or system capability. Individuals represent component features and properties, depending on the type of object property defined. ObjectProperty is used to composite AADL components, SA implementation and the property set. It also describes the relationship between the classes of the architecture components and the knowledge base.

DataProperties is dedicated to the model measurement objective and units of SAO and has a relationship with the AADL property set file. However, the property set has a distinct declaration structure, which is covered in more detail in the next table. *NamedIndividual* is related to individual components and properties. In this case, it is only dedicated to individual components as the *DataProperties* cover properties in property set modeling. From *DataRange*, we defined the list of components that applies a set of properties. Lastly, *AnnotationProperty* provides information to the AADL property set, such as metric unit conversion.

AADL models enable architectural assessments and analyses based on the declaration of properties. These properties are modeled in a separate file called the *PropertySet*, which has a defined structure composed of property type declaration, property constant declaration and property declaration.

Property type defines a type for the values that are acceptable to a property, establishing a name and the set of legal values for a property of this type through a type definition. Property definition provides the name and type of properties used by AADL elements. Property constants are property values that are known by a symbolic name and can be used wherever the value itself is allowed [3].

Thereby, to support the fulfillment of properties in AADL model declaration, we defined a set of seven DataProperties that should be followed as a standard in modeling the data property to SAO property assertions, as shown in Table 3. The type indicates the name of the property set, to support the creation of different property set files. Units define the

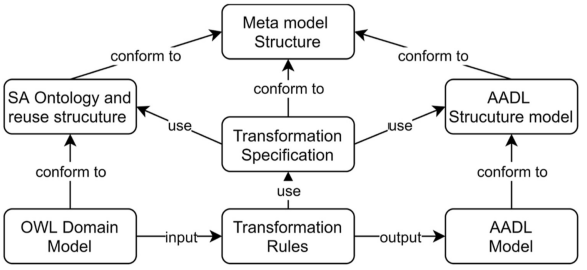

Figure 3.
OWL to AADL transformation engine process

OWL entities	AADL element
Class	Components related to SA, sections or capability
Individuals	Components and features used
DataProperties	Property set from SA components
ObjectProperty	Define composite components
DataRange	List of components to which specific property belongs
AnnotationProperty	A comment that provides information to AADL property set

Table 2.
Relationships among OWL entities and AADL components

measurement units of data, which can be declared by QUDT measurement and the unit's ontology; constant supports the constant numerical value of a measurement unit; Valuemin and Valuemax are used to define numerical values to support range value declaration and Range defines, which components of the architecture can use a created property.

In Table 4, we summarize the alignment between the OWL data type and basic property type constructors provided by the AADL standard. In this case, OWL data type declarations are defined to support the fulfillment of the model transformation process, following a set of variables to support the construction of the AADL property set and enabling the AADL plugin analyses in the OSATE2 tool.

The *transformation rules* block is composed of two sequential activities. The first is to transform the OWL file in Java objects related to the OWL structure. In the second activity, these objects were transformed to Java objects related to the AADL model structure through *transformation Specification*, where objects related to AADL are used as the basis for creating AADL files.

In the first activity, we used a library to process and transform the OWL file into Java objects. Because it is based on XML, the tool uses the standard Java library (available in the org.w3c.dom package) widely used in the development of tools. This library returns a Java object containing the entire contents of the file.

The OWL2AADL tool searches for the node Class; in the sequence DataProperties and ObjectProperty; and, finally, Individual. It is important to follow this order because of the references defined by the OWL2 structure model. For example, an Individual node contains references to one or more owl nodes: Class, DataProperties and ObjectProperty.

Finding a defined node, the tool starts an individualized treatment: it checks the subnodes and their attributes. This scan looks for components as defined in the knowledge base and SAO. From this scan, the data are separated to be used in the conversion. Entities that do not

Table 3.
OWL data properties' definitions for AADL property set

OWL data properties	AADL property set
<owl:DataProperties rdf:about="http...#type"/>	Property type declaration
<owl:DataProperties rdf:about="http...#constant"/>	Property constant declaration
<owl:DataProperties rdf:about="http...#unit"/>	Property declaration
<owl:DataProperties rdf:about="http...#Valuemin"/>	
<owl:DataProperties rdf:about="http...#Valuemax"/>	
<rdfs:domain rdf:resource="http...#AADLComponent"/>	
<rdfs:isDefinedBy rdf:resource="http...#Measurement"/>	
<owl:DataProperties rdf:about="http...#Range"/>	

Table 4.
OWL and AADL data type alignment declarations

OWL data types	AADL property types
xsd:boolean	aadlboolean
xsd:string	aadlstring
sao:enumeration	enumeration
sao:units	units
owl:real	aadlreal
xsd:int	aadlinteger
sao:range	range of
sao:classifier	classifier
sao:comment	reference
sao:record	record

follow the defined rules are ignored and may be included in the future to represent other domains.

The second activity occurs in the conversion of OWL-related objects to AADL-related objects. In this activity, the tool creates a set of files, including a library component, AADL package and property set. After conversion at the Java object level, AADL-related objects are used to guide the writing of AADL files.

5. Case study

In order to illustrate the SAO application, including the model transformation from OWL to AADL, a case study related to the development of a UAV is presented in this section. Such a UAV design is, in fact, part of a research effort named ProVant. It involves two Brazilian research institutions, the Federal University of Minas Gerais (UFMG) and the Federal University of Santa Catarina (UFSC), in collaboration with the University of Seville (Spain). Since the beginning of ProVant, four aircraft prototypes have been designed. Besides enhancing the flight properties, each new version aims to enhance the problems encountered in the predecessor project. The fourth-generation aircraft is currently under construction, and this study collaborated to refine the draft design (version 4.0) to the one that is in fact being constructed (version 4.1).

This section is organized in two parts. Section 5.1 relates to the application of SAO to develop the OWL model of the Provant 4.0 aircraft. Next, in 5.2, we show the transformation of the OWL model into AADL through the developed OWL2AADL tool.

5.1 ProVANT 4.0 OWL model

In order to present more details regarding the representation of the aircraft (mechanical, structural parts), an ontology dedicated to UAVs called drone ontology [30] was reused, which is a knowledge base in the aerospace domain. In this way, it was possible to detail the physical process of CPS, such as the type of mission and application, and the aircraft components (i.e. fuselage, payload, landing and lift system, engine type).

The components that make up the SA were provided by the engineering team and are presented in Table 5. Based on SAO entities, these components were modeled in OWL, detailing their characteristics, properties and relationships with the physical process of the system. Each component has a specific set of properties, which includes maximum and minimum values, and the related units of measurement, which need to be properly defined.

Each component type has a set of technical characteristics provided by their manufacturers (available on data sheet and manuals), which represent its capabilities. These data properties will be fulfilled with the numerical values of each capability of the components of the architecture to be modeled, thus enabling semantic evaluations to be carried out.

After declaring the data properties of the ProVANT 4.0 components, the individual modeling of the components and the system begins. Each created individual contains three definitions field-modeled based on SAO declarations: types, object properties assertions and data properties assertions, as presented in Figure 4. Types define the classes that the components belong to. In this case, ProVANT 4.0 has a list of types to describe generic UAV characteristics, by *drone ontology*, such as fuselage, UAV category and hardware components (sensors and actuator). Object properties assertions follow the individuals' relationships. The modeling of the individual that represents the ProVANT 4.0 architecture and its components is carried out with the declaration of the object properties assertions of SAO, where each component was previously modeled, also as an individual, and declared in the architecture.

An example of modeling the AXI 2418 brushless motor is shown in Figure 5. This component is classified as an electric motor in drone ontology and a device in SAO. Among

Qty	Component	Type	Description/functionality
2	STM32F767ZI	Development board	Development board with STM32F767ZI microcontroller
1	JetsonTx2	Development board	Development board with artificial intelligence application
1	Navio 2	Autopilot HAT	Autopilot Hw composed of 2 x IMU, 1 x GPS, 1 x barometer
1	ADIS164890	IMU	Principal, precise, and accurate inertial measurement unit to provide data to a system
1	3DR Pixhawk Airspeed	Pitot tube	A wind speed sensor
1	MB2530 IRXL-MaxSonar-CS3	Sonar	Sonar sensor to provide a distance data by reflection measurement
1	UBLOX NEO-M8T	GPS	Global positioning system, a sensor that provides position data to system
2	Flat Maxon motor brushless EC 45	Brushless motor	Brushless motor used to provide tilt VTOL capability
2	Maxon controller ESCON 36/3	ESC	*n* electronic speed control to brushless motors for VTOL capability
2	Maxon sensor Encoder MILE, 512	Encoder	Type of mechanical motion sensor that creates a digital signal from motion
6	Hitec D145SW Digital HV	Servo-motor	Servo motor(device) used to control UAV wing movements
2	AXI 2830/10 V2 LONG	Brushless motor	Brushless motor used for propulsion engines of UAV.
2	Mezon 160 ESC	ESC	Electronic speed control for brushless motor of propulsion engines
1	Turnigy iA6C PPm/ SBUS	Radio receiver	Radio transceiver device used to receive and transmit data from UAV and control base on the ground
1	Battery management system (BMS)	Energy control system	System used to control the current consumption and charging battery by solar panels in UAV
1	Max3232 RS232/UART	UART converter	Device converter of RS232 bus communication used to interface compatibility between components
2	KNACRO RS422 to TTL UART	UART converter	Device converter of RS422 bus communication used to interface compatibility between components
4	LIFE 1500mah 25C 3S	Power supply	Set of components that provide power source to the system (battery pack)

Table 5.
ProVANT 4.0 hardware components

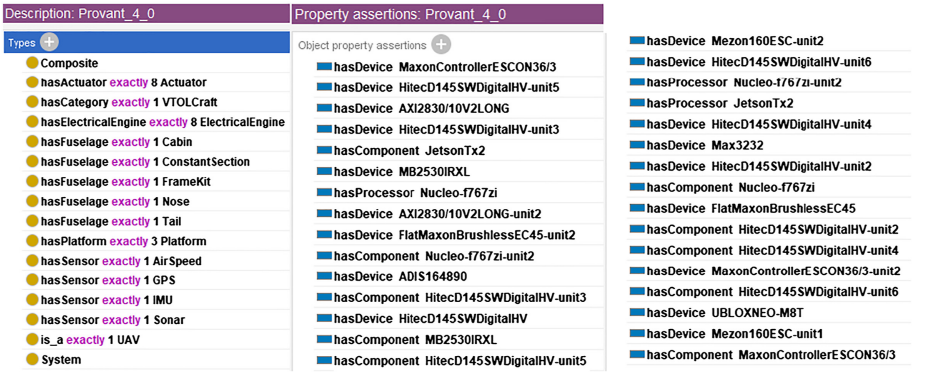

Figure 4.
OWL individual of Provant 4.0 system

the object properties, hardware characteristics were defined, such as PWM communication port, power consumption properties, period and mass. The data assigned to the properties are presented in the data property assertions.

In total, the ontology has 1219 axioms, 696 logical axiom, 129 classes, 133 object property, 34 data property, 86 individuals and 47 annotation property. From the entities of the OWL model, reasoners pallet of the Protegé tool were applied to analyze inconsistencies and inferences between the components of the architecture, physical process and quality attributes such as performance, behavior, traceability and support to designers for identifying impacts in architectural components change.

5.2 Model transformation: OWL to AADL

After modeling, in OWL, the data properties and individual that represent the ProVant systems and their components, the designer uses the OWL2AADL transformation tool developed in this work.

The tool is an OSATE2 plugin that performs the process of transforming OWL models to AADL, as presented in Section 4. The process is based on MDE concepts, where, from the metamodels of the input (OWL) and output (AADL), a set of transformation rules proposed in the transformation engine are performed. The tool reads the OWL file and processes the relationship with OWL entities to AADL elements, generating two AADL model files containing the property set and the architectural model. These two generated files contain the minimum AADL structure necessary for modeling the system and carrying out evaluations and analysis in OSATE2.

The generated *SAO* AADL property set file is based on OWL data properties declared with the definitions presented in Table 3 from the transformation engine process. The *SAO* AADL property set file follows the AADL structure declaration, which defines the data type, unit of measure, maximum and minimum values and capability name. A part of the SAO property set file is shown in Listing 1. In total, 42 lines of code were automatically generated, enabling the modeling of properties of AADL components of the ProVant SA.

The second file generated by the tool is the OWL2AADL package, which contains the AADL components of the architecture, resulting from the transformation process of the OWL individuals. OWL model input provide 83 individuals, including the ProVANT architecture, component features and library candidate input. In total, the OWL2AADL output file creates 74 AADL components, represented in 409 automatically generated command lines.

The number of AADL components generated in the transformation was smaller than the number of OWL individuals because there were individuals not directly related to the types of components. In total, nine individuals representing the section type were used to build the component features. Regarding outstanding space limitations, in Listing 2, we show some AADL components that make up the Provant 4.0 SA, generated using the OWL2AADL tool.

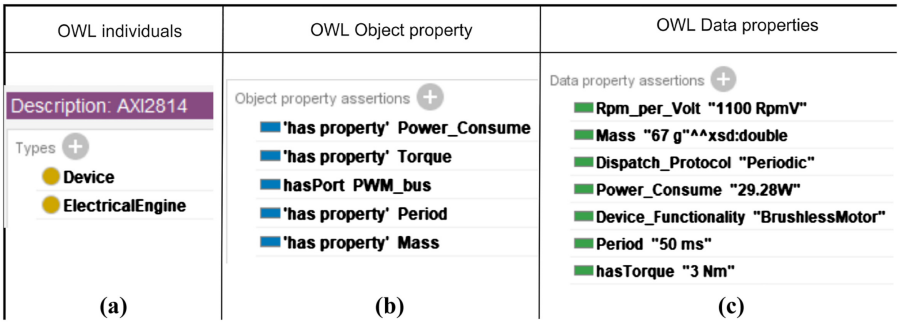

Figure 5. OWL individual AXI2814 brushless motor

```
property set SAO is
BatteryCapacity_Units: type units( Watt);
BatteryCapacity_Min : constant aadlinteger units SAO:BatteryCapacity_Units => 0;
BatteryCapacity_Max : constant aadlinteger units SAO:BatteryCapacity_Units => 1000;
BatteryCapacity_Range : type aadlinteger SAO:BatteryCapacity_Min .. SAO:BatteryCapacity_Max units SAO:
    BatteryCapacity_Units;
BatteryCapacity : BatteryCapacity_Range applies to (bus, device, memory, processor);
Mass_Units: type units( g);
Mass_Min : constant aadlreal units SAO:Mass_Units => 0;
Mass_Max : constant aadlreal units SAO:Mass_Units => 2000;
Mass_Range : type aadlreal SAO:Mass_Min .. SAO:Mass_Max units SAO:Mass_Units;
Mass : Mass_Range applies to (device, memory, processor, system);
Power_Consume_Units: type units( Watt);
Power_Consume_Min : constant aadlinteger units SAO:Power_Consume_Units => 0;
Power_Consume_Max : constant aadlinteger units SAO:Power_Consume_Units => 10000;
Power_Consume_Range : type aadlinteger SAO:Power_Consume_Min .. SAO:Power_Consume_Max units SAO:Power_Consume_Units;
Power_Consume : Power_Consume_Range applies to (device, memory, processor, system);
...
Accelerometer_Accuracity_Units: type units( mg/sqrHz);
Accelerometer_Accuracity_Min : constant aadlinteger units SAO:Accelerometer_Accuracity_Units => 0;
Accelerometer_Accuracity_Max : constant aadlinteger units SAO:Accelerometer_Accuracity_Units => 100;
Accelerometer_Accuracity_Range : type aadlinteger SAO:Accelerometer_Accuracity_Min .. SAO:Accelerometer_Accuracity_Max
    units SAO:Accelerometer_Accuracity_Units;
Accelerometer_Accuracity : Accelerometer_Accuracity_Range applies to (device, system);
```

Listing 1.
AADL model generated by OWL2AADL plugin

```
package OWL2AADL
public
system implementation ProVANT_4_0
  subcomponents
    MaxonControllerESCON363 : device MaxonControllerESCON363;
    MaxonControllerESCON363-unit2 : device MaxonControllerESCON363-unit2;
    MEZON160ESC-unit2 : device MEZON160ESC-unit2;
    MEZON160ESC : device MEZON160ESC;
    Nucleo-f767zi-unit2 : processor Nucleo-f767zi-unit2;
    Nucleo-f767zi : processor Nucleo-f767zi;
    JetsonTx2 : processor JetsonTx2;
    HitecD145SWDigitalHV-unit6 : device HitecD145SWDigitalHV-unit6;
    ...
    HitecD145SWDigitalHV : device HitecD145SWDigitalHV;
    AXI2830/10V2LONG  device AXI2830/10V2LONG;
    AXI2830/10V2LONG-unit2 device AXI2830/10V2LONG-unit2;
    Max232 : device Max232;
    MB2530RXL : device MB2530RXL;
    FlatMaxonBrushlessEC45-unit2 : device FlatMaxonBrushlessEC45-unit2;
    FlatMaxonBrushlessEC45 : device FlatMaxonBrushlessEC45;
    KNACRORS422-unit2 : device KNACRORS422-unit2;
    3DRPixhawkAirSpeed : device 3DRPixhawkAirSpeed;
    UBLOXNEO-M8T : device UBLOXNEO-M8T;
  properties
    SAO:BatteryCapacity => 3000 mah;
    SAO:flightDuration => 40 minutes;
    SAO:MaxFlightRange => 20 km;
    SAO:payloadCapacity => 6 Kg;
end ProVANT_4_0;
end OWL2AADL;
```

Listing 2.
AADL model generated by OWL2AADL plugin

After the OWL2AADL tool generates the AADL models and property set from individuals' representation, it is possible to open these files in OSATE2 and thus perform instances and analyses of the architecture through the development of plugins. This allows us to carry out multiple analyses related to the impact of quality attributes and characteristics.

6. Conclusions and future work

In this work, we proposed an SAO, aimed at representing semantically the SA of CPS, providing terminologies, vocabulary and concepts and identifying the relationships between hardware, software and system components, with a focus on implementation. SAO can be reused in other domain ontologies to represent the details of the SA. We conducted a study of

previous works that have addressed this issue, with the intention of using ontology-based benefits (high abstraction modeling) and AADL design to represent SA models.

We also developed a tool for transforming OWL models into AADL in order to use ontologies in a specific domain to generate an architectural model of the system. We presented an example of application in the aerospace domain with OWL modeling to demonstrate the capacity of details about description, objects and data properties that support the transformation process to an AADL model with a property set and architecture file generation.

For future work, as a wide variety of related ontologies use SA components, a study of semantic rules will be performed, seeking alignment with the W3C standard module. Ontology-based design provides a rich knowledge base, including the physical process, network and computational parts of CPS, which can be used to support SA design by a transformation model to create a large set of library components, a property set and an architectural model in order to evaluate the system's capability and implementation.

Note

1. https://github.com/diegosales/OWL2AADL

References

1. Jaakkola H., Thalheim B. Architecture-driven modelling methodologies. In: Information Modelling and Knowledge Bases XXII. IOS Press; 2011. p. 97-116.
2. Hause M., *et al.* The SysML modelling language. In: Fifteenth European Systems Engineering Conference, 9; 2006. p. 1-12.
3. Feiler P.H., Lewis B.A., Vestal S. The SAE Architecture Analysis & Design Language (AADL) a standard for engineering performance critical systems. IEEE; 2006. p. 1206-211.
4. de Saqui-Sannes P., Hugues J. Combining SysML and AADL for the design, validation and implementation of critical systems. ERTS2. 2012: 117.
5. Gruber T.R., *et al.* A translation approach to portable ontology specifications. Knowl Acquisit. 1993; 5(2): 199-220.
6. Vanherpen K., Denil J., David I., De Meulenaere P., Mosterman P.J., Torngren M., Qamar A., Vangheluwe H. Ontological reasoning for consistency in the design of cyber-physical systems. In: 2016 1st International Workshop on Cyber-Physical Production Systems (CPPS). IEEE; 2016. p. 1-8.
7. Herzig S.J., Qamar A., Paredis C.J.. An approach to identifying inconsistencies in model-based systems engineering. Proc Comp Sci. 2014; 28: 354-62.
8. Törngren M., Qamar A., Biehl M., Loiret F., El-Khoury J. Integrating viewpoints in the development of mechatronic products. Mechatronics. 2014; 24(7): 745-62.
9. Bermudez-Edo M., Elsaleh T., Barnaghi P., Taylor K. IoT-Lite: a lightweight semantic model for the Internet of Things. In: 2016 INTL IEEE conferences on ubiquitous intelligence & computing, advanced and trusted computing, scalable computing and communications, cloud and big data computing, internet of people, and smart world congress. IEEE; 2016. p. 90-7. available at: uic/atc/scalcom/cbdcom/iop/smartworld.
10. Olszewska J.I., Barreto M., Bermejo-Alonso J., Carbonera J., Chibani A., Fiorini S., Goncalves P., Habib M., Khamis A., Olivares A., *et al.* Ontology for autonomous robotics. In: 2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN). IEEE; 2017. p. 189-94.
11. Janowicz K., Haller A., Cox, *et al.* SOSA: A lightweight ontology for sensors, observations, samples, and actuators. J Web Sem. 2019; 56: 1-10.
12. Uschold M., Grüninger M. Ontologies: Principles, methods and applications. Knowl Eng Rev. 1996; 11.

13. Fernandez M., Gomez-Perez A., Juristo N. Methontology: from ontological art towards ontological engineering. In: proceedings of the AAAI97 spring symposium series on ontological engineering, Stanford, USA; 1997. p. 33-40.
14. Gómez-Pérez A., Fernández M., De Vicente A. Towards a method to conceptualize domain ontologies. In: ECAI96 Workshop on Ontological Engineering. Budapest; 1996. p. 41-51.
15. Noy N.F., McGuinness D.L., *et al.* Ontology development 101: A guide to creating your first ontology. Stanford knowledge systems laboratory technical report KSL-01-05 and . . .; 2001.
16. Feldmann S., Herzig S.J., Kernschmidt K., Wolfenstetter T., Kammerl D., Qamar A., Lindemann U., Krcmar H., Paredis C.J., Vogel-Heuser B.. Towards effective management of inconsistencies in model-based engineering of automated production systems. IFAC-PapersOnLine. 2015; 48(3): 916-23.
17. Balduccini M., Griffor E., Huth M., Vishik C., Burns M., Wollman D. Ontology-based reasoning about the trustworthiness of cyber-physical systems; 2018.
18. Clements P.C. A survey of architecture description languages. In: Proceedings of the 8th international workshop on software specification and design. IEEE; 1996. p. 16-25.
19. Parsia B., Patel-Schneider P., Motik B. OWL 2 web ontology language structural specification and functional-style syntax. W3C, W3C Recommendation; 2012.
20. Na H.-S., Choi O.-H., Lim J.-E. A Method for Building Domain Ontologies based on the Transformation of UML Models. In: Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06); 2006. p. 332-38. doi: 10.1109/SERA.2006.4.
21. Trinkunas J., Vasilecas O. Ontology transformation: From requirements to conceptual model. Scientific Papers. Comp Sci Inform Technol. 2009; 751: 52-64. University of Latvia.
22. Novacek J., Viehl A., Bringmann O., Rosenstiel W. Ontology-based Requirements Transformation. In: 2019 International Symposium on Systems Engineering (ISSE); 2019. p. 1-8. doi: 10.1109/ISSE46696.2019.8984265.
23. Wardhana H., Ashari A., Sari A. Transformation of SysML Requirement Diagram into OWL Ontologies. Int J Adv Comp Sci Appl. 2020; 11. doi: 10.14569/IJACSA.2020.0110415.
24. S. AS5506. Architecture analysis and design language (aadl). Embedded Computing Systems Committee. 2004. SAE.
25. Behjati R., Yue T., Nejati S., Briand L., Selic B. Extending SysML with AADL concepts for comprehensive system architecture modeling. In: European Conference on Modelling Foundations and Applications. Springer; 2011. p. 236-52.
26. Brun M., Faugère M., Delatour J., Vergnaud T. From UML to AADL: a Need for an Explicit Execution Semantics Modeling with MARTE. In: Embedded Real Time Software and Systems (ERTS2008); 2008.
27. Goncalves F.. Integrated Method For Designing Complex Cyber-Physical Systems. PhD thesis. Universidade Federal de Santa Catarina; 2018.
28. Bailin S.C., Hodgson R., Keller P.J. Large-Scale Knowledge Sharing for NASA Exploration Systems.
29. Feiler P. The Open Source AADL Tool Environment (OSATE). Technical Report. Carnegie Mellon University Software Engineering Institute Pittsburgh United. . .; 2019.
30. Martín-Lammerding D., Córdoba A., Astrain J.J., Medrano P., Villadangos J. An Ontology-Based System to Collect WSN-UAS Data Effectively. IEEE IoT J. 2021; 8(5): 3636-52. doi: 10.1109/JIOT.2020.3023168.

Corresponding author
Diego Camara Sales can be contacted at: diegocsales@gmail.com