Date: July 2025

System Profile for Effective Cyber Threat-based Risk Assessments (SPECTRA) version 1.0

Volume 5: SPECTRA Core Assertions Metamodel, v1.0 – beta 1

OMG Document Number: ptc/25-07-02

Standard Document URL: https://www.omg.org/spec/SPECTRA/

This OMG document replaces the submission document (sysa/25-02-01). It is an OMG Adopted Beta Specification and is currently in the finalization phase. Comments on the content of this document are welcome and should be directed to issues@omg.org by June 2025.

You may view the pending issues for this specification from the OMG revision issues web page https://issues.omg.org/issues/lists.

The FTF Recommendation and Report for this specification will be published in April 2026. If you are reading this after that date, please download the available specification from the OMG Specifications Catalog.

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 9C Medway Road PMB 274, Milford, MA 01757, U.S.A.

TRADEMARKS

CORBA®, CORBA logos®, FIBO®, Financial Industry Business Ontology®, FINANCIAL INSTRUMENT GLOBAL IDENTIFIER®, IIOP®, IMM®, Model Driven Architecture®, MDA®, Object Management Group®, OMG®, OMG Logo®, SoaML®, SOAML®, SysML®, UAF®, Unified Modeling Language®, UML®, UML Cube Logo®, VSIPL®, and XMI® are registered trademarks of the Object Management Group, Inc.

For a complete list of trademarks, see: https://www.omg.org/legal/tm_list.htm. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

OMG's Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process, we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page https://www.omg.org, under Specifications, Report a Bug/Issue.

Table of Contents

List of Figures

Preface

OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable, and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies, and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at https://www.omg.org/.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. All OMG Specifications are available from the OMG website at:
https://www.omg.org/spec

All of OMG's formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters
9C Medway Road, PMB 274
Milford, MA 01757
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: *pubs@omg.org*

Certain OMG specifications are also available as ISO standards. Please consult https://www.iso.org

1 Scope

SPECTRA is a language for describing cyber and cyber-physical systems for the purposes of risk assessments, cybersecurity assessments and vulnerability assessments. System descriptions - including models, consist of many artifacts which are of importance for one or more lifecycle phases. For the purposes of a cybersecurity assessment, certain artifacts are of essence - for example, what are the parts of the system, how these parts are connected to convey information, what information is being conveyed, and what is the nature of the parts. Cybersecurity implies a filter for the level of technical detail, compared to other disciplines involved in the system lifecycle. Effectively extracting only the relevant cybersecurity assertions for a system description is a challenging task. SPECTRA language - a set of conceptual entities and relations, collectively referred to as Core Assertions for cybersecurity - extends Systems Engineering languages with means to identify the core entities and their relationships to support the task of interpreting and postprocessing a system description by automated tools and enabling cybersecurity analytics. SPECTRA facilitates ingesting normalized machine-consumable system descriptions into compliant tools for big data analytics in cybersecurity.

SPECTRA's objective is to provide a standard compliance reference for acquisition contracts soliciting models for various assessments, as well as tools and services for performing such assessments automatically.

This specification defines the SPECTRA Core Assertions Metamodel.

2 Conformance

2.1. Introduction

The SPECTRA for SysML v1 profile specification defines the following four compliance points:

1. Model compliance (XMI)
2. Model compliance (JSON)
3. Analytics Tool (Consumer) compliance (XMI)
4. Analytics Tool (Consumer) compliance (JSON)

2.2. Model Compliance Point (XMI)

An XMI file conforming to the SPECTRA Model compliance point shall be a well-formed SPECTRA CA XMI document where the assertions about the system of interest are made according to the meaning, semantics and constraints described in this specification. It is the responsibility of the producer of the compliant model to choose which meanings to use, based on the need to communicate certain assertions about the system of interest. This involves the mandatory elements defined in the SPECTRA Kernel to communicate the key assertions about the system of interest, as well as any extended meanings and optional elements suggested by this SPECTRA specification.

This compliance point facilitates interchange of SPECTRA models for cyber and cyber-physical systems that can be unambiguously interpreted by the SPECTRA Consumer software (including but not limited to cyber risk assessment tools).

2.2. Model Compliance Point (JSON)

A JSON file conforming to the SPECTRA Model compliance point shall be a well-formed SPECTRA CA JSON document where the assertions about the system of interest are made according to the meaning, semantics and constraints described in this specification. It is the responsibility of the producer of the compliant model to choose which meanings

to use, based on the need to communicate certain assertions about the system of interest. This involves the mandatory elements defined in the SPECTRA Kernel to communicate the key assertions about the system of interest, as well as any extended meanings and optional elements suggested by this SPECTRA specification.

This compliance point facilitates interchange of SPECTRA models for cyber and cyber-physical systems that can be unambiguously interpreted by the SPECTRA Consumer software (including but not limited to cyber risk assessment tools).

2.3. Analytics Tool (Consumer) Compliance Point (XMI)

Software that conforms to the SPECTRA Consumer compliance point shall ingest SPECTRA XMI documents defined according to the Model Compliant Point (XMI). A compliant consumer tool shall be able to ingest all elements described in this specification. This compliance point does not restrict the capabilities of the compliant software. For example, the compliant consumer may choose to ignore some of the extended SPECTRA meanings and focus of the mandatory meanings. At this level of compliance SPECTRA allows compliant tools to have the same interpretation of the input SPECTRA CA model of the SOI, as intended by the modeler.

This compliance point allows various analytics to be performed on the ingested models, included but not limited to the Threat-based Cyber Risk Assessment.

2.3. Analytics Tool (Consumer) Compliance Point (JSON)

Software that conforms to the SPECTRA Consumer compliance point shall ingest SPECTRA JSON documents defined according to the Model Compliant Point (JSON). A compliant consumer tool shall be able to ingest all elements described in this specification. This compliance point does not restrict the capabilities of the compliant software. For example, the compliant consumer may choose to ignore some of the extended SPECTRA meanings and focus of the mandatory meanings. At this level of compliance SPECTRA allows compliant tools to have the same interpretation of the input SPECTRA CA model of the SOI, as intended by the modeler.

This compliance point allows various analytics to be performed on the ingested models, included but not limited to the Threat-based Cyber Risk Assessment.

3 References

3.1 Normative References

- MOF 2.5.1 formal/2019-10-01
- ISO 21778:2017 Information technology — The JSON data interchange syntax
- ISO/IEC IEEE 15288:2015, Systems and software engineering - System life cycle process

3.2 Non-normative References

- NIST SP-800-30
- ISO/IEC 27005
- NIST SP-800-37
- SPDX
- CycloneDX v1.6 Ecma, OWASP, 2024
- NIST SP-800-53
- NIST SP-800-53a
- KDM
- Prolog

4 Terms and Definitions

No additional terms or definitions.

5 Symbols

No additional symbols/abbreviations.

6 Additional Information

6.1 How to read this specification

SPECTRA Core Assertions (CA) Metamodel is organized as a collection of ***packages*** (see section SPECTRA overview), including a **Kernel**. Each package defines ***abstract*** and ***concrete*** classes. Packages and abstract classes provide the backbone of the model organization. There are 16 abstract classes across 16 packages (12 top level, and 4 subpackages of Kernel).

SPECTRA CA defines 26 concrete classes across all packages.

12 concrete classes elements are **essential** to identifying assertions about the SOI. These elements can be found in the Kernel package. The key classes for the purposes of SPECTRA are: ReplaceableUnit, Channel, CarrierInterface, Datatype, Exchange, Capability and Thread.

The remaining classes are **optional** and constitute the Access, Behavior, Mission, Mitigation and GlobalControl packages.

SPECTRA CA uses enumeration to define **an extended vocabulary** relevant for descriptions of cyber and cyber-physical systems. SPECTRA CA uses 28 enumerations. There are some 139 enumeration literals. Most of them can be found in the Characteristics, Artifacts and Traits packages.

The section on **Domain Characteristics** is quite **essential** to the organization and objectives of SPECTRA, so it can be reviewed at the same time as the essential elements in the Kernel Package.

6.2 Semantics of SPECTRA Core Assertions

Semantics of the SPECTRA metadata for SysML v1 is defined in terms of the implied Core Assertions about the SOI and how they can be derived from a SysML v1 model with SPECTRA annotations.

When metadata is added to an element of a SysML model, one or more core assertions are made about the SOI. This process involves several steps.

First, a certain initial claim is considered. The name of the claim corresponds to the name of the metadata. The signature of the claim is determined by the Core Assertion Metamodel. Some parameters of the claim are determined by the tags of the stereotypes, while other parameters are derived from the SysML properties of the element. SPECTRA distinguishes mandatory and optional tags and its attributes, since in most cases the parameters required by SPECTRA Core Assertions are already available in a reasonably complete SysML model.

Second, the parameters of the claim are established. These rules describe how parameters are established from the combination of the information supplied in the tags and the information already available in the SysML model and how it can be referenced from the stereotyped element.

Third, a certain custom rule is applied that uses implication to connect the claim and the specific implied Core Assertions. The claim implies making one or more core assertions. It also implies constraints and well-formedness conditions.

SPECTRA semantic description uses Prolog to describe the claims, the core assertion rules, inference rules and the core assertion infrastructure involved in managing core assertions. Prolog is an adequate approach since it combines a database of assertions and logical inference rules.

The rules for deriving parameters to claims from the combination of metadata attributes and information in the SysML model are described informally.

SPECTRA Core Assertions are outlined in Annex B of this specification.

7 SPECTRA overview

7.1 Organization of SPECTRA packages

SPECTRA Core Assertions Metamodel specifies a MOF model intended to represent the Core Assertions for cyber and cyber-physical systems. The elements are arranged in several packages based on their purpose.

SPECTRA has a Kernel package that defines the mandatory elements representing the key Core Assertion entities and relationships. The Kernel package consists of five subpackages.

The Characteristics, Artifacts and Traits and Data packages define extended stereotypes for cyber and cyber-physical systems.

Access, Behavior and Mitigation define some optional elements.

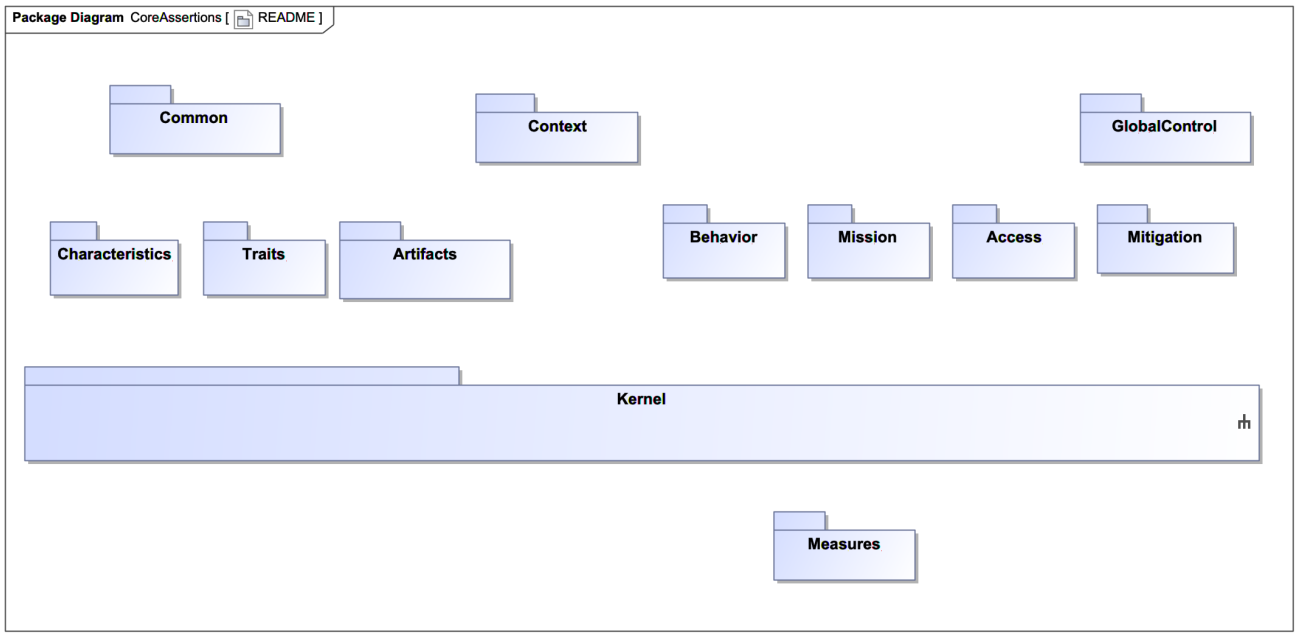

Figure 1 SPECTRA CA packages

Table 1 SPECTRA CA packages

Package	Description
Common	Defines several abstract classes that constitute the top elements of the taxonomy of the Core Assertions.
Context	Defines the container for the Core Assertions instances and the context for the SOI.
Kernel	Defines the mandatory elements for the parts and assemblies, conveying elements and conveyable elements. Also defines key impactful elements and functional threads. The elements in the Kernel package cover the majority of the Core Assertions. Elements in other packages build upon the ones defined in the Kernel.
Characteristics	Characteristics (together with Artifacts and Traits) provide a shared vocabulary to make assertions about the nature of the elements of SOI. Characteristics

	describe the common assertions related to the nature of various elements of the SOI.
Traits	Traits (together with Artifacts and Characteristics) provide a shared vocabulary to make assertions about the nature of the elements of SOI. Traits describe the common assertions related to the nature of the replaceable elements and datatypes of the SOI and their role in the missions and capabilities supported by the SOI.
Artifacts	Artifacts (together with Traits and Characteristics) provide a shared vocabulary to make assertions about the nature of the elements of SOI. Artifacts describe the common assertions related to the nature of the replaceable elements of the SOI, their role in missions and capabilities supported by the SOI, and their place in the supply chain enabling the SOI. Artifacts address assertions related to the unique technologies of the replaceable elements.
Access	The Access package defines some optional Core Assertions related to the agents involved in the operations and their access to the SOI. Access package allows representing Operator, Maintainer and Supplier and some relationships describing their access to the capabilities of the SOI
Behavior	The Behavior package defines some optional Core Assertions related to the custom language of the functions, performed by the SOI (either directly by one of the replaceable units or through collaboration of several units). Such functions are usually related to processing, converting, transforming various items, producing items, disposing of items, performing computations, making decisions, etc.
Mission	The Mission package defines several optional Core Assertions related to the mission context of the SOI.
Mitigation	The Mitigation package defines some optional Core Assertions related to the Security Domain, in particular the stereotypes in the Mitigation package identify mitigation control elements and their position with respect to the mitigatable elements.
GlobalControl	Defines the elements that represent a global catalog of controls which provides the context for mitigations of the SOI. Provides the global context in which mitigation controls are defined
Measures	Defines some basic measures involved in Core Assertions.

8 SPECTRA Common package

8.1 Overview

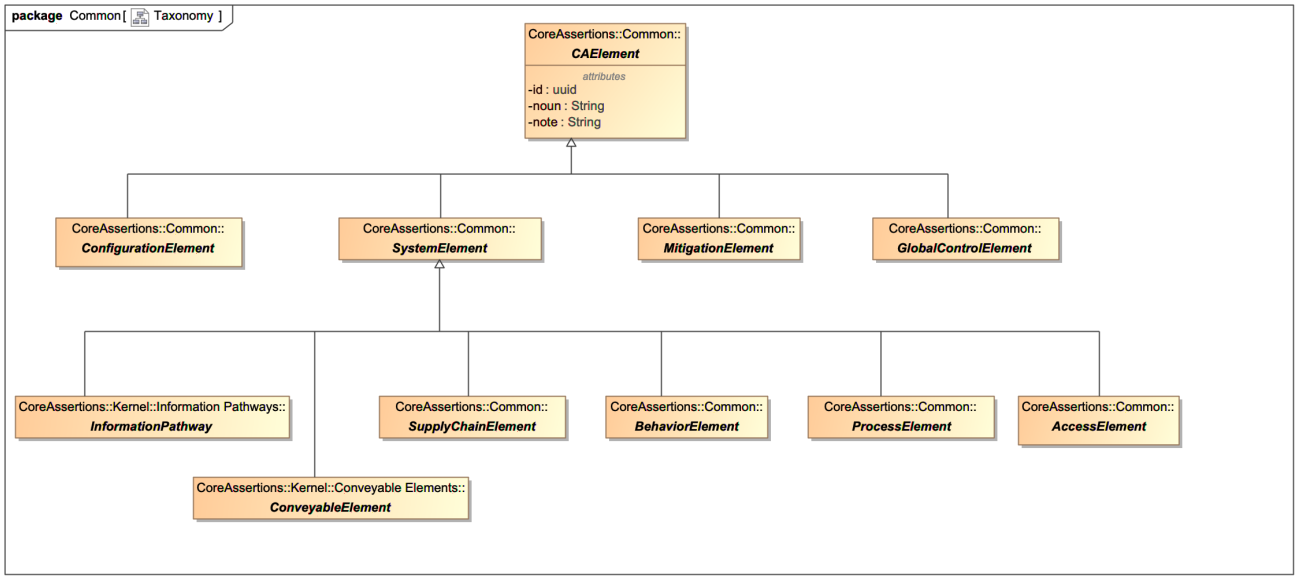

Figure 2 Taxonomy of SPECTRA Core Assertion elements

8.2 CA Elements

8.2.1 CAElement (abstract)

CAElement is the top abstract class that represents a Core Assertion element.

Properties

id:uuid	Unique id of the element.
noun:String	Name of the class for the element
note:String	Notes for the element

Semantics

8.2.2 ConfigurationElement (abstract)

Some Core Assertion elements are used as “scaffolding” to manage the entire collection of the Core Assertion for a specific System Of Interest (SOI).

Generalization

CAElement

8.2.3 SystemElement (abstract)

Most Core Assertion elements represent logical assertions for a specific System Of Interest.

Generalization

CAElement

8.2.4 MitigationElement (abstract)

Some Core Assertion elements represent assertions about mitigation controls in the context a specific System Of Interest.

Generalization

CAElement

8.2.5 GlobalControlElement (abstract)

Some Core Assertion elements represent assertions about a global mitigation context independent of a specific System Of Interest (in particular, common to several SOI). These elements are often referred to a Control Catalog. SPECTRA Code Assertion Metamodel provides means to represent such catalogs, so that assertions about mitigation controls for a specific SOI can be interpreted in a proper global context.

Generalization

CAElement

8.2.6 SupplyChainElement (abstract)

Supply Chain Element is an abstract class that represents artifacts in the context of the supply chain arrangements for the SOI. In SPECTRA, supply chain elements are optional, and hence such Core Assertion elements are defined in a separate package outside of the SPECTRA Kernel.

Generalization

CAElement

8.2.7 BehaviorElement (abstract)

Some Core Assertion elements represent assertions about the functions performed by the SOI. In SPECTRA, behavior elements are optional, and hence such Core Assertion elements are defined in a separate package outside of the SPECTRA Kernel.

Generalization

CAElement

8.2.8 ProcessElement (abstract)

Process Elements represent the logical sequences of operations, transformations, or decisions that drive system functionality, independent of any specific structural or technological implementation. They define how functions interact over time to achieve system objectives, encapsulating workflows, behaviors, and system-wide operational dynamics. Some essential process elements are defined in SPECTRA Kernel (Capability and Functional Thread), while other are defined outside of the Kernel (Mission, MissionPhase, Task).

Generalization

CAElement

8.2.9 AccessElement (abstract)

Some Core Assertion elements represent assertions about the access to the SOI. In SPECTRA, access elements are optional, and hence such Core Assertion elements are defined in a separate package outside of the SPECTRA Kernel.

Generalization

CAElement

This page intentionally left blank.

9 SPECTRA Kernel package

9.1 Overview

The Kernel Package defines the mandatory elements representing the parts and assemblies, conveying elements and conveyable elements. It also defines key process elements such as capabilities and functional threads. The elements in the Kernel package cover the majority of the Core Assertions. Elements in other packages build upon the ones defined in the Kernel. The Kernel package consists of four subpackages.

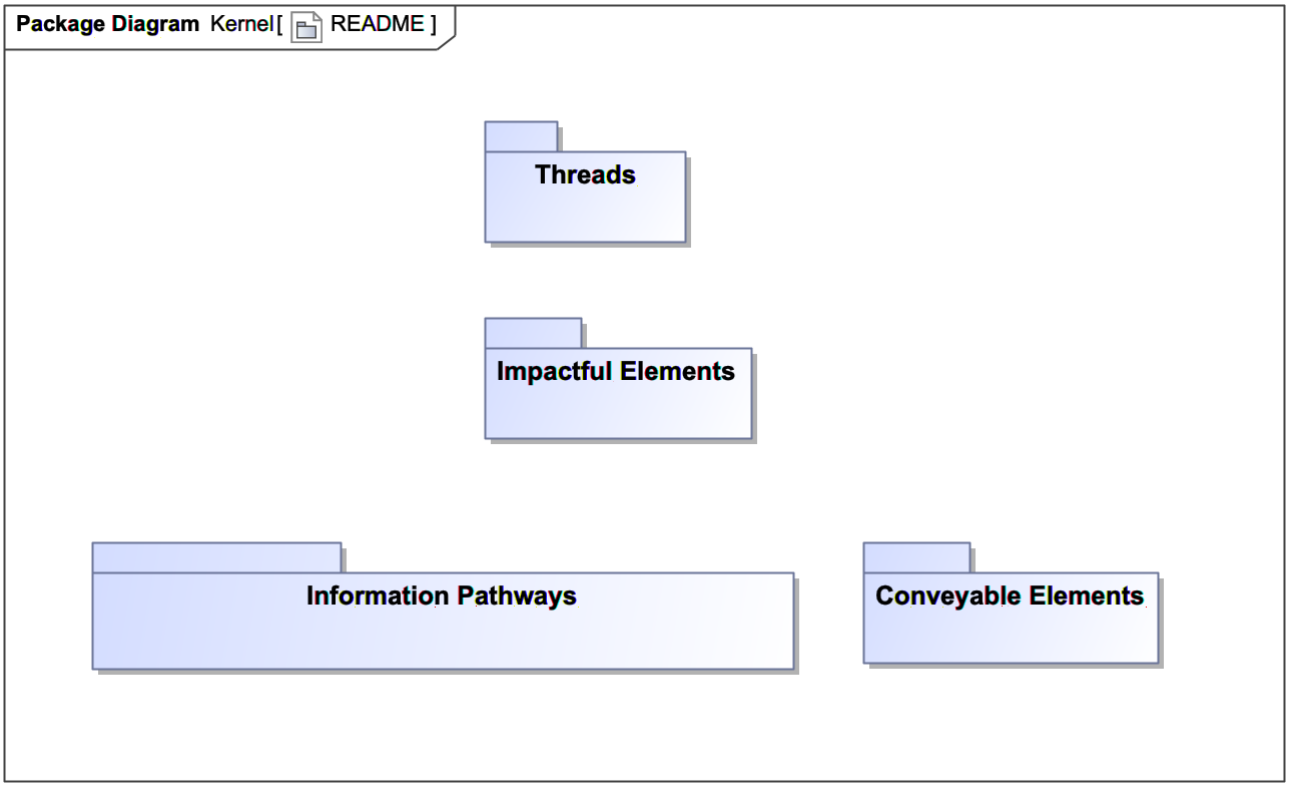

Figure 3 Kernel subpackages

The key package is Information Pathways.

Table 2 Kernel subpackages

Package	Description
Kernel::Information Pathways	Defines the mandatory elements for the parts and assemblies, conveying elements, and several supporting elements that are associated with Digital Information Pathways, such as Exchange, Datastore, and Carrier Interface. The elements in this package address the “WHERE” clauses in the assertions about the SOI.
Kernel::Conveyable Elements	Defines the mandatory elements for the digital information as it is conveyed, stored and processed within the processing element and the communication fabric, defined by the Information Pathways elements. The elements in this

	package address the “WHAT/To WHAT” clauses in the assertions about the SOI.
Kernel::Impactful Elements	Defines mandatory elements for identifying elements of the SOI that are of value to the stakeholders of the SOI. Since such elements are sensitive to the losses in one of the cybersecurity domains, such as Confidentiality, Integrity and Availability, the elements of this package provide the means to optionally add the corresponding measurements to the model. The elements in this package address the “WHAT/Impacting WHAT” clauses in the assertions about the SOI.
Kernel::Threads	Defines mandatory elements for identifying model elements that define functional threads for the SOI. The elements in this package address the “WHEN” clauses in the assertions about the SOI.

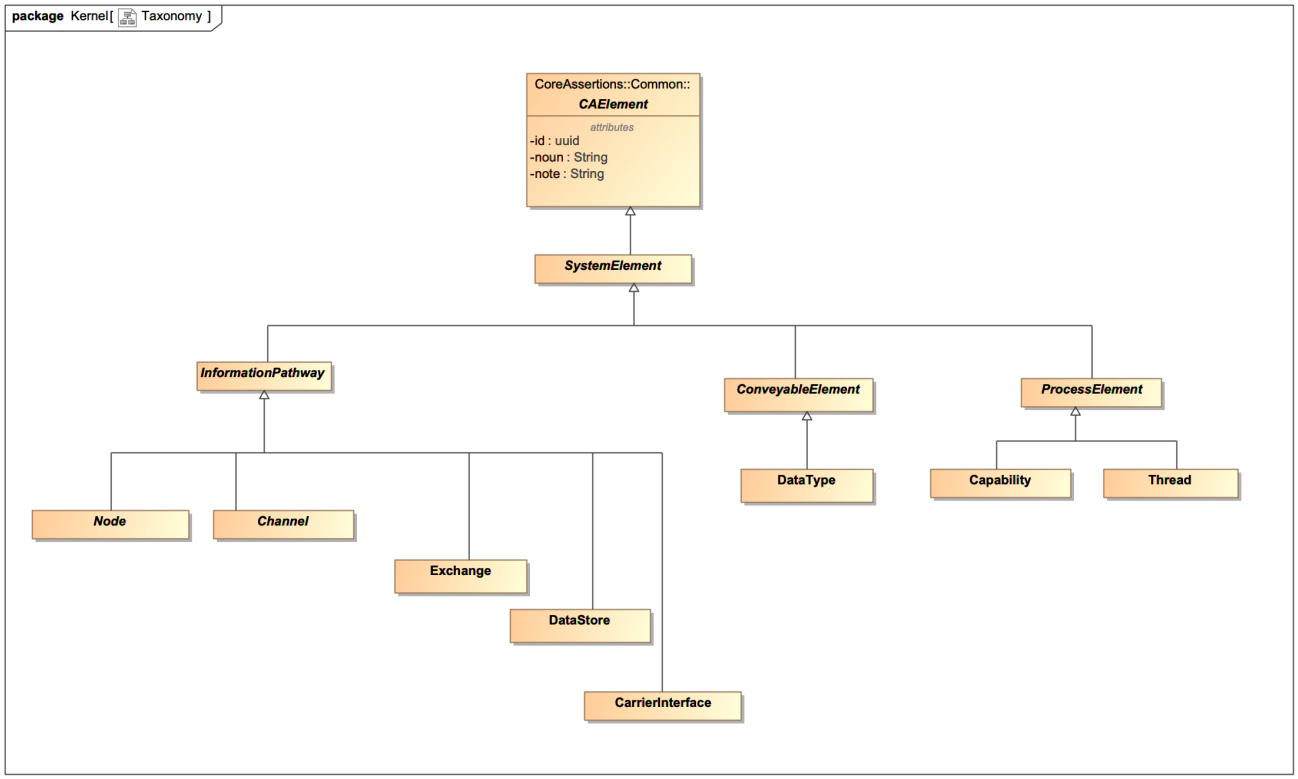

Figure 4 Taxonomy of the Kernel package

9.2 Information Pathways Package

9.2.1 Information Pathway (abstract)

An Information Pathway - is a collection of elements that together describe the "geography" of the SOI in terms of "places" (nodes, parts, assemblies and sub-assemblies - referred to as Replaceable Elements), and connectors between them (channels or edges, circuits, buses - referred to as Conveying Elements) and descriptions of the flows of digital information (and some material items) between these places.

Information Pathways describe the core assertions related to the parts and interconnections between these parts and address the “WHERE?” question/clause in the assertions related to the SOI. For example, “Failure of the VHF radio causes loss of the capability to communicate”. Here “VHF Radio” is the “WHERE?” clause. Same “Capability to communicate” (which is the “WHAT/To WHAT” clause) can be also disrupted/denied at other places in the SOI.

Generalization

SystemElement

Semantics

An element identified as an Information Pathway represents an identifiable “place” in the SOI. Specific meanings are provided by the subclasses of Information Pathway. In particular, SPECTRA distinguishes two types of “places” in cyber systems: Nodes (Processing Elements) and Conveying Elements (communications fabric of the system).

“Places” is a description of a cyber system are usually hierarchically arranged. SPECTRA distinguishes two hierarchies: “indented list of equipment” (single master configuration) and multiple independent configurations or assemblies. The “indented list of equipment” is related to the process of integrating the SOI from components and subcomponents.

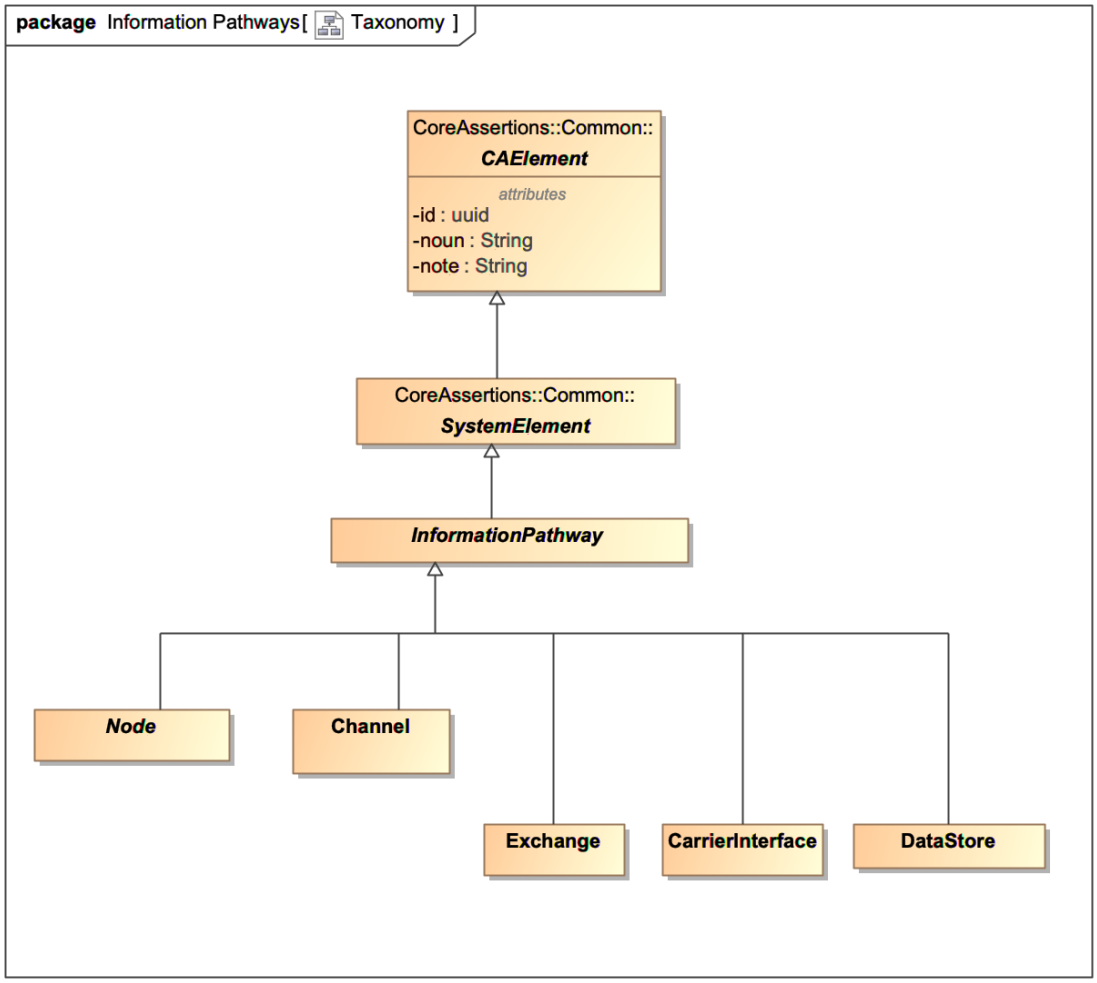

Figure 5 Taxonomy of the Information Pathways package

9.2.2 Node (abstract)

A Node (can also be called a Processing Element, but SPECTRA reserves this term of a specific Application Trait) - is a block or a part that describes a "place" in the SOI. This “place” can be a set of tangible parts and connections in the actual SOI, rather than one “tangible” place. Some "places" in the SOI are tangible (e.g. a piece of equipment, or a facility), while others are intangible, virtual arrangements of other units (e.g. an Electrical Subsystem).

A Node can be a bearer of information, a (possibly aggregated) producer and/or consumer of information flows, a performer of a function, a scope for a collaboration, a unit of integration, supply and maintenance, a unit of configuration, and a unit of defense by way of allocated controls.

Generalization

InformationPathway

Properties

name:String	Specifies the identifying name of the element
purpose:String	Provides the description of the element
isInternal:Boolean	Specifies whether the element is internal or external in the context of SOI
domain:DomainEnum	Specifies the domain of the element
characteristic:CharacteristicEnum[0..*]	Specifies a set of unique characteristics of the element

Associations

/node:Node[0..*]	(derived, optional) specifies the child nodes of the element, if the node is composite. This property provides a generic interface, and specific subclasses redefine it to define specific constraints on the parent-child relationships

Semantics

SPECTRA has a two-fold complementary perspective on the SOI: 1) a set of parts (replaceable units, processing elements), plugged into unique places within the communication fabric, and 2) communication fabric (conveying elements) that defines the “placeholders” for the replaceable units. Replaceable Units define the “technical surface” of the SOI. isInternal property of Nodes identify the exact set of parts under assessment, and help identify the end-to-end dataflows through the system, especially ones originating outside of the SOI and threading through the parts of the SOI. Or dataflows originating inside of the SOI and threading outside of the SOI. From the CRA perspective, this information is essential to understand the **technical surface** of the SOI and to identify possible attack paths for the SOI.

Both Replaceable Units and Conveying Elements constitute the “places” in the SOI. Other “places” are aggregates of some replaceable units - they are “above” the technical surface of the SOI. Other “places” are internal parts of Replaceable Units - they are “below” the technical surface of the SOI.

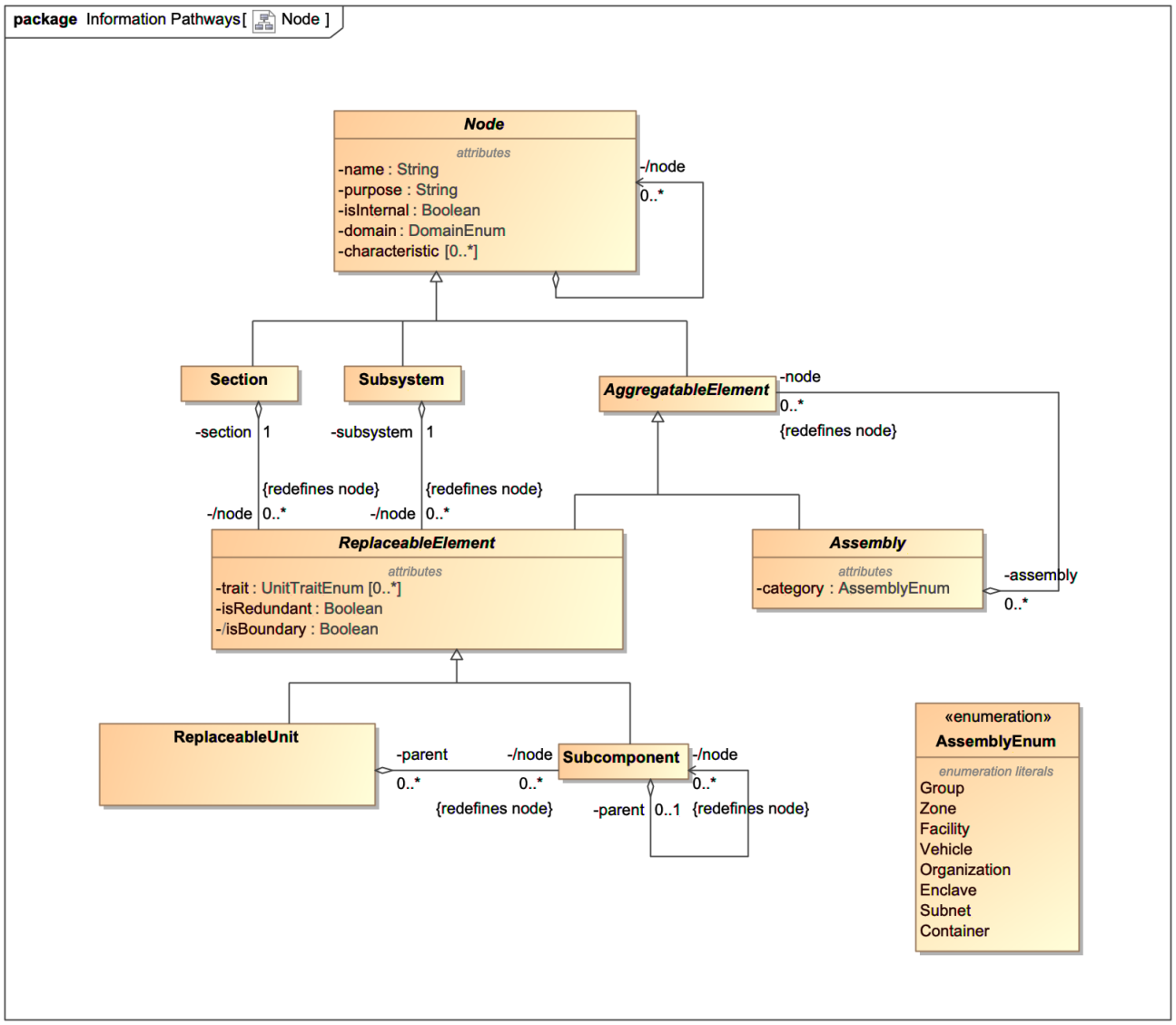

Figure 6 Node

9.2.3 Section

Section – a highest level of structural decomposition of the SOI into independent or autonomous areas (possibly each under a distinct authority), e.g. spacecraft, ground station, launch vehicle. Sections often participate in distinct mission phases, and therefore in distinct mission (and/or capability) configurations. Therefore, sections are significant for the purposes of risk assessment.

System of Interest – one or more section where isInternal=true represent the System Of Interest (SOI). All assertions related to SOI are owned by a single element called the Context (see the Context package).

Generalization

Node

Associations

node:Node[0..*]	(redefines node in Node)

Constraints

- A section can only have ReplaceableElement instances as child nodes.

9.2.4 Subsystem

Subsystem - a highest level of structural decomposition of the SOI into functional configurations where replaceable units collaborate to provide related functions and/or capabilities, e.g. communications subsystems or thermal subsystems. Every subsystem is a parent of replaceable units or possibly subcomponents that each constitute a further decomposition of replaceable units. Subsystems are significant for the purposes of cyber risk assessment, because of their alignment to system functions and/or capabilities, and thus the related operational impacts.

Generalization

Node

Associations

node:Node[0..*]	(redefines node in Node)

Constraints

- A subsystem can only have ReplaceableElement instances as child nodes.

9.2.5 AggregatableElement (abstract)

This abstract class is defined as a common class of elements that can be parts of various node configurations called assemblies. AggregatableElement subclasses are constrained to ReplaceableElements and Assemblies, and exclude Sections and Subsystems.

Generalization

Node

Associations

assembly:Assembly[0..*]	Assemblies of which the element is a member

9.2.6 Replaceable Element (abstract)

A Replaceable Element represents a tangible node in the SOI that plugs into a unique place in a web of buses and links, and thus produces or consumes a unique set of flows and performs a unique set of functions. This is an abstract element; its concrete subclasses are "replaceable unit" and "subcomponent". Replaceable units represent a certain level of the structural decomposition of the SOI that is commensurate with the tangible conveying elements (communications fabric of the system).

A Replaceable Element is important for understanding the technical surface of the SOI as they provides unique opportunities and the context for attacks on the SOI. A replaceable unit may represent a set of interdependent attack opportunities (in terms of suppliable artifacts) (e.g. a custom software image running on a specific type of hardware and a specific type of operating system). Usually, a replaceable unit is pre-integrated, pre-configured, and then supplied to be integrated into the SOI. Since a replaceable unit is pre-integrated, the specific knowledge of how it can be attacked is obtained from studying it as a whole.

The model may choose to further describe subcomponents of a replaceable unit and define their collaborations, and how each subcomponent represents unique traits (see package Traits) and thus unique opportunities for attacks (in terms of suppliable artifacts). It is most beneficial for the purposes of a cyber risk assessment to consider a replaceable unit

together with its (abstracted) functions, suppliable artifacts, and the flows that it serves separately from its subcomponents and their interactions. Thus, the complexity of the attack paths throughout the entire SOI end-to-end is reduced.

The interactions of subcomponents of a replaceable unit for a cyber system are quite complex and the corresponding functions are usually emergent. Therefore, it is more practical to treat behavior of Replaceable Units as irreducible and not try to derive anything from the more elementary functions of subcomponents. This also reduces the complexity of understanding attack paths during the risk assessment.

The level of decomposition called "replaceable unit" of the SOI is relative to the scope of the risk assessment (e.g. mission context, system-of-systems, individual system and its subcomponents, as independent systems would all have different levels of what is considered a replaceable element.

Generalization

AggregatableElement

Properties

trait:UnitTraitEnum [0..*]	Specifies the unique traits of the element
isRedundant:Boolean	Specifies whether the element has redundant units
/isBoundary:Boolean	(derived) specifies whether the element is part of the boundary of the SOI (either from the external side or from the internal side)

Associations

section:Section	Section of the ReplaceableElement
subsystem:Subsystem	Subsystem of the ReplaceableElement

Analytics

- The value of isBoundary is derived. A ReplaceableElement is considered boundary (isBoundary=true) if there exist at least one Exchange for which that ReplaceableElement is either producer or consumer, such that the isInternal properties of the produce and the consumer of that Exchange have opposite values (i.e. the Exchange crosses the boundary between SOI and its environment).

9.2.7 Replaceable Unit

A Replaceable Unit represents a tangible node in SOI that plugs into a unique place in the web of networks, buses and links and thus produces and consumes a unique set of exchanges and performs a unique set of functions. A replaceable unit fits into a unique place defined by its connections to networks, buses and links in SOI.

A replaceable unit may represent a set of interdependent attack opportunities (in terms of suppliable artifacts) (e.g. a custom software image running on a specific type of hardware and a specific type of operating system). A replaceable unit is usually pre-integrated and pre-configured and then supplied to be integrated into the SOI.

From the risk assessment perspective, it is often preferable to consider the functions of a replaceable unit as irreducible, rather than as collaborations between subcomponents, providing a level of separation between replaceable units and any of its subcomponents in the end-to-end information pathways involving other replaceable units and channels.

A replaceable unit is involved in defining functional threads (various replaceable units are the “places” through which the thread “traverses”).

A set of replaceable units is the scope of collaboration.

Generalization

ReplaceableElement

Associations

node:Node[0..*] (redefines node in Node)

Constraints

- A ReplaceableUnit can only have Subcomponent instances as child nodes.

Semantics

When stereotype ‘ReplaceableUnit’ is added to a block or part, the following claim is made about the SOI:

replaceableUnit(id, name, isInternal, note)

Derived parameters:

id- id of the connector in the SysML model
name - is derived from the SysML name of the element
note - is derived from the notes associated with the element
isInternal - if the element has stereotype “internal” then ‘true’, if the element has stereotype “external” then ‘false’, default ‘true’.

Implied Core Assertions:

replaceableUnit(Oid, Name, IsInternal, Note) :- newid()=Id, assertz(node(Id, ‘ReplaceableUnit’, Name)), assertz(originalRef(Id,Oid)), assertz(internal(Id)), setBoundaryCrossing(Id), assertz(note(newid(), Id, Note).

9.2.8 Subcomponent

Subcomponent - any block that is part of a replaceable unit. A model of a SOI may choose to describes more levels of structural decomposition for certain replaceable units, with internal channels and flows and more elementary functions, collaborating to produce the functions of the entire replaceable units as emergent behaviors; and specific suppliable artifacts representing unique attack opportunities; for cyber and cyber-physical systems such collaborations may be quite complex, and the model may or may not have a high fidelity description of such behaviors; for the purposes of cyber risk assessment, subcomponents are ignored and their "boundary" flows are abstracted to the corresponding replaceable unit, their emergent functions abstracted to the replaceable unit, and all their artifacts aggregated and assigned to the replaceable unit. Since the corresponding replaceable unit is pre-integrated prior to its deployment into the SOI, this allows normalization of the information pathways and reduction in their complexities. The functions and flows of each replaceable unit are treated as axioms regardless of subcomponents. Marking an element as a subcomponent (of some replaceable unit) hides this element from information pathways.

A subcomponent corresponds to a certain level in the "master" multi-level structural decomposition of the SOI, known as the BOM or the "indented list of equipment".

Generalization

ReplaceableElement

Associations

node:Node[0..*] (redefines node in Node)

Constraints

- A Subcomponent can only have Subcomponent instances as child nodes.

9.2.9 Assembly

An Assembly (also can be referred to as Functional Configuration) is a collection of replaceable units of the SOI, that collaborate to perform some system function and/or implement a capability. This is a concrete element, and the specific semantics is further provided by the AssemblyCatagoryEnum.

Generalization

AggregatableElement

Properties

category:AssemblyEnum	Specifies an incoming Flow

Associations

node:Node[0..*]	(redefines node in Node)

Constraints

- An Assembly can only have ReplaceableElement and other Assemblies (with same of different category) as child nodes.

Semantics

A ReplaceableUnit can be part of exactly one Section, and exactly one Subsystem, but may be a member of one or more Assembly.

9.2.10 AssemblyCategoryEnum (Enumeration)

Table 3 Enumeration Literals for Assembly Category

Enum Value	Definition
Group	Group - generic arrangement of replaceable nodes
Organization	Organization - arrangement of replaceable nodes that has common organizational controls
Enclave	Enclave - arrangement of replaceable units that has common organizational and technical controls. An enclave is defined as a collection of information systems connected by one or more internal networks under the control of a single authority and security policy.
Subnet	Subnet - arrangement of replaceable units around a common subnet (possibly in an enclave) that has common security policy and other constraints, common configuration, as well as common organization and technical controls, such as a firewall, etc.
Facility	Facility - arrangement of replaceable units that has common physical controls, such as a building
Vehicle	An arrangement of replaceable units that is mobile platform, e.g. an aircraft or a truck.
Zone	An (often spacial) arrangement of replaceable units, important for physical security
Container	An (often spacial) arrangement of replaceable units, such as a locked cabinet.

9.2.11 Channel

Channel (can be also called a conveying element) - any connector or an association between replaceable units that allows a flow of information or other items; channel has at least one producer and at least one consumer; a special case is a "local link" where one replaceable element is both the producer and the consumer; an information channel supports a certain protocol stack that may involve multiple levels; channel is usually attackable, defendable and configurable. Together with replaceable elements channels describe the information pathways in the SOI (tangible/attackable connectors between tangible/attackable places).

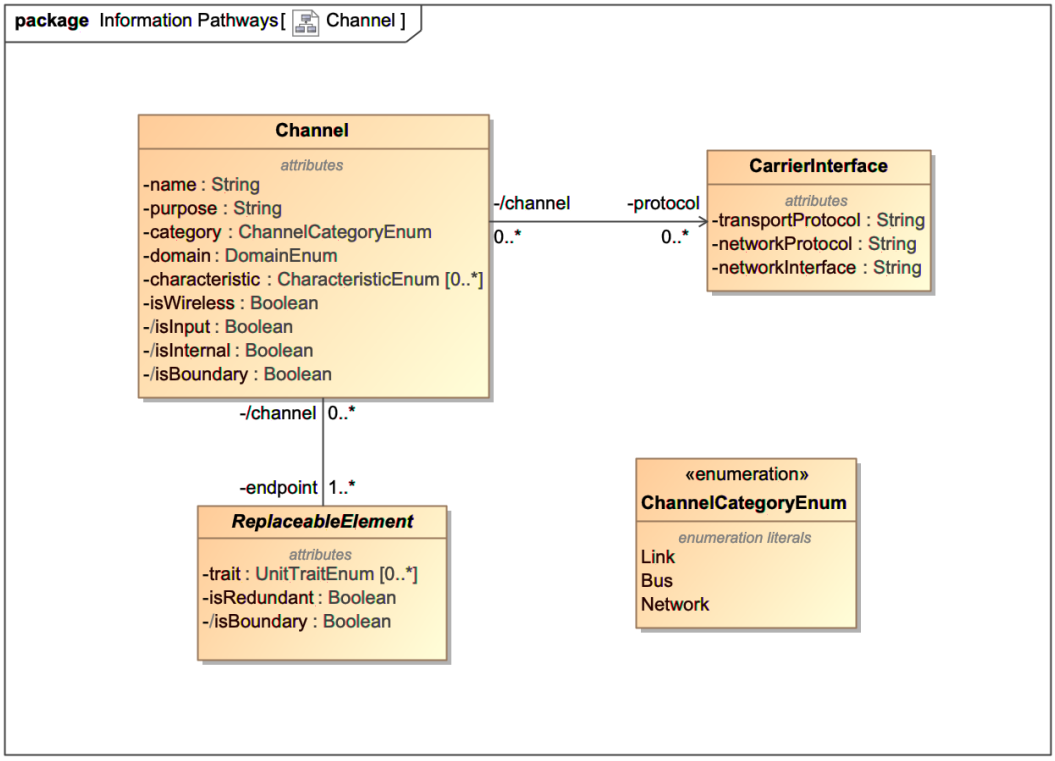

Figure 7 Channel

Generalization

InformationPathway

Properties

name:String	Specifies the identifying name of the element
purpose:String	Provides the description of the element
category:ChannelCategoryEnum	
domain:DomainEnum	
characteristic:CharacteristicEnum[0..*]	
isWireless:Boolean	
/isInput:Boolean	(derived)
/isInternal:Boolean	(derived)
/isBoundary:Boolean	(derived)

Association

protocol:Carrier Interface [0..*]	specifies the Carrier Interface for this Conveying Element
endpoint:ReplaceableUnit[1..*]	

Constraints

- Link (category=Link) shall have exactly two endpoint

Semantics

SPECTRA has a two-fold complementary perspective on the SOI: 1) a set of parts (replaceable units, processing elements), plugged into unique places within the communication fabric, and 2) a communication fabric (constituted by Conveying Elements) that defines the “sockets” for the replaceable units. Conveying Elements and Replaceable Units define the “technical surface” of the SOI. This technical surface identifies the exact set of parts under assessment, helps identify the end-to-end data flows through the system, especially those that originate outside of the SOI and thread through the parts of the SOI and those that originate inside of the SOI and thread outside of the SOI. From the CRA perspective, this information is essential to identify possible attack paths for the SOI.

A more specific meaning of a conveying element is given by one of the concrete subclasses.

9.2.12 ChannelCategoryEnum (Enumeration)

Table 4 Enumeration Literals for Channel Category

Enum Value	Definition
Link	Link - a channel between exactly one producing and one consuming Replaceable Element.
Bus	Bus - a channel among one or more producing and/or one or more consuming Replaceable Elements. For example, a bus may have a single producer and multiple consumers, or multiple producers and a single consumer, or multiple producers and multiple consumers. One Replaceable Element can be both a producer and a consumer for the same bus.
Network	A Network (as defined in NIST-800-53) is a system implemented with a collection of connected components. Such components may include routers, hubs, cabling, telecommunications controllers, key distribution centers, and technical control devices. A Network involves a communication medium connecting one or more computing devices and involving common communication protocols over digital interconnections using telecommunications technologies based on physically wired, optical or wireless radio-frequency methods and arranged in a variety of network topologies. For example, an Ethernet local area network is a Network. One Replaceable Element can be both a producer and a consumer for the same network.

9.2.13 Exchange

Exchange - a representation of a type of item that flows between a single producer and a single consumer, involving an information type or some other (physical) kind of a Conveyable Element, a Channel and a specific Carrier Interface supported by the Channel. While cyber systems involve "information flows", in a cyber physical system one may find a broader range of "item flows" where items conveyed from the producer to the consumer may be physical. An exchange is described as a combination of a specific channel (network, bus or link), a unique (logical) information type and a unique (physical) protocol. A channel may support multiple communication protocols.

The Conveyable Element of the Exchange shall has domain=Application. Models of cyber and cyber-physical systems shall avoid using Carrier Data as Conveyable Element in Exchanges. The Carrier Interface for the Channel (channel) shall be used instead.

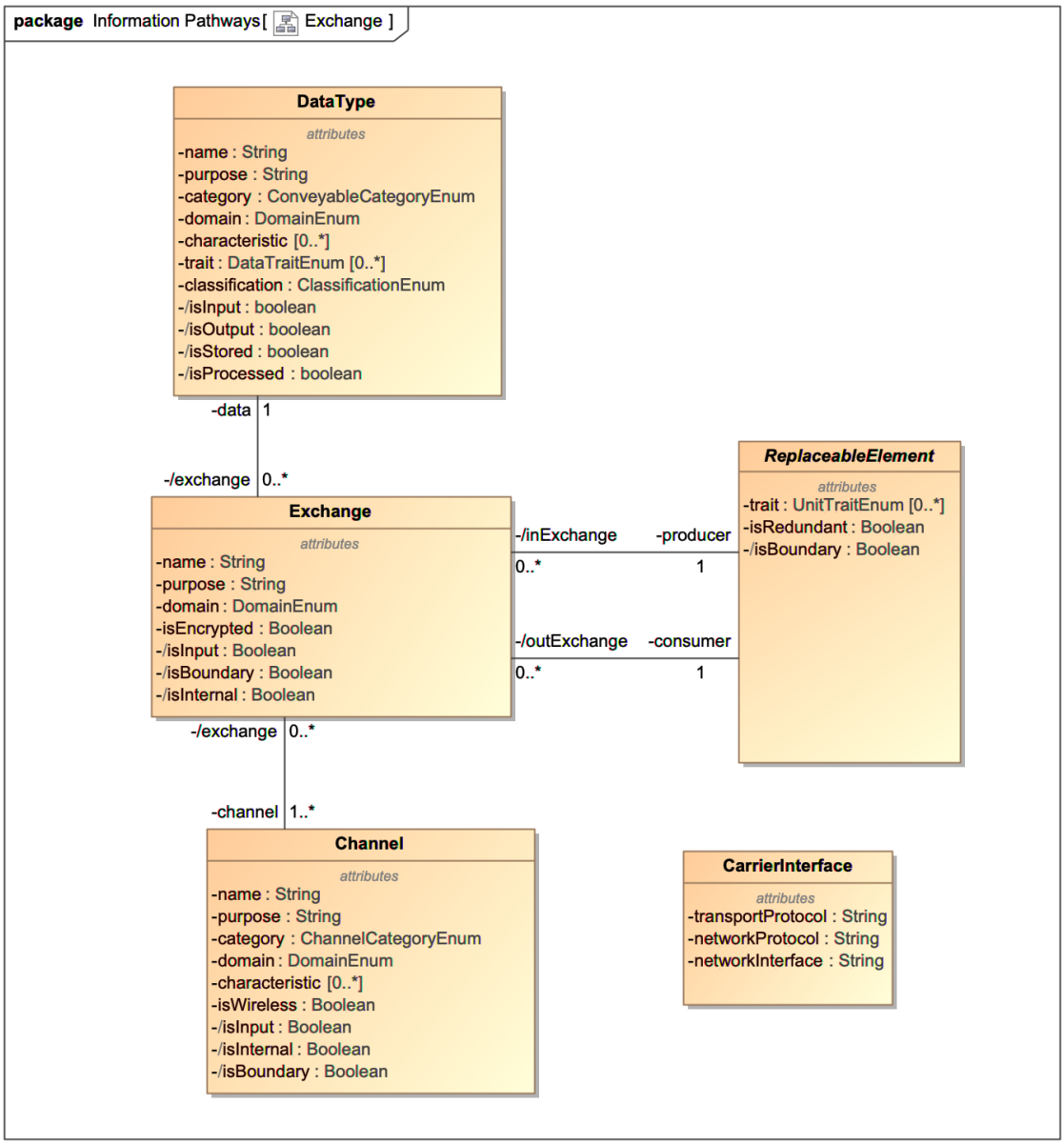

Figure 8 Exchange

Generalization

InformationPathway

Properties

name:String	Specifies the identifying name of the element
purpose:String	Provides the description of the element
domain:DomainEnum	Specifies the domain of the element
isEncrypted:Boolean	Specifies whether the Exchange is encrypted
/isInput:Boolean	(derived)

/isInternal:Boolean	(derived)
/isBoundary:Boolean	(derived)

Associations

data:DataType[1]	Specifies an Conveyable Element conveyed by this Flow (e.g Operational Data or a Resource)
channel:Channel[1]	Specifies a Conveying Element through which the Flow is conveyed
producer:ReplaceableElement[1]	Specifies the producer of the Exchange
consumer:ReplaceableElement[1]	Specifies the consumer of the Exchange

Analytics

- An exchange is considered internal (isInternal=true) if both the produce and the consumer units are internal (isInternal=true). The value in the isInternal column shall match the value derived for the definitions of the units.
- An exchange is considered boundary (isBoundary=true) if producer and consumer units have opposite isInternal values. The value in the isBoundary column shall match the value derived for the definitions of the units.
- An exchange is considered input (isInput=true) if both the producer unit is external (isInternal=false) and the consumer unit is internal (isInternal=true). The value in the isInput column shall match the value derived for the definitions of the units.

Semantics

Encryption is the process of encoding information in a way that, ideally, only authorized parties can decode. This process converts the original representation of the information, known as plaintext, into an alternative form known as ciphertext. Applies to Flow, Datastore (however encryption is achieved by the producer and consumer, not by the channel), but one of the protocols of the carrier interface may support encryption, which also entails exchange of crypto keys).

9.2.14 Datastore

A Datastore is a repository for persistently storing and managing collections of data, which include not just repositories such as databases, but also simpler store types such as simple files, emails, logs, etc. As part of the Core Assertions, A Datastore represents “data at rest”.

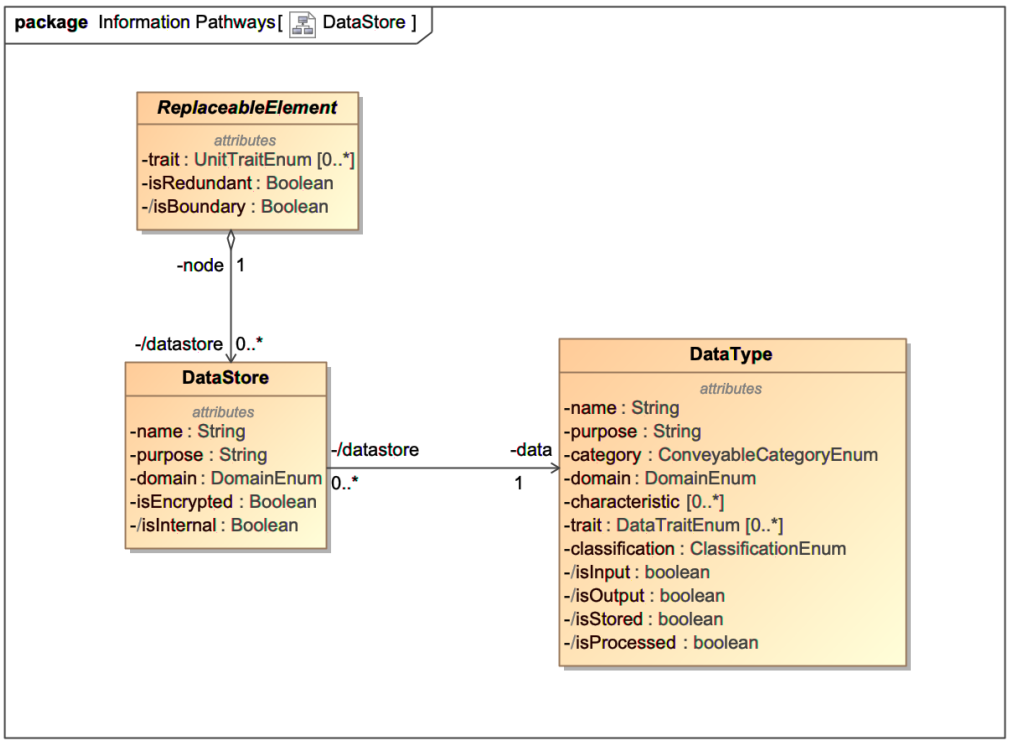

Figure 9 Datastore

Generalization

InformationPathway

Properties

name:String	Specifies the identifying name of the element
purpose:String	Provides the description of the element
domain:DomainEnum	Specifies the domain of the element
isEncrypted:Boolean	Specifies whether the datastore is encrypted
/isInternal:Boolean	(derived)

Associations

data:DataType[1..*]	Specifies Operational Data stored in the Datastore.
node:ReplaceableElement	Specifies a ReplaceableElement that owns the datatore

Analytics

- A datastore is considered internal (isInternal=true) if the unit that owns the datastore is internal (isInternal=true). The value in the isInternal column shall match the value derived for the definitions of the units.

Semantics

Encryption is the process of encoding information in a way that, ideally, only authorized parties can decode. This process converts the original representation of the information, known as plaintext, into an alternative form known as ciphertext. Applies to Flow, Datastore (however encryption is achieved by the producer and consumer, not by the channel), but one of the protocols of the carrier interface may support encryption, which also entails exchange of crypto keys).

9.2.15 Carrier Interface

A Carrier Interface is a representation of the essential protocols supported by a Channel (Conveying Element). A protocol is a system of rules that allows two or more entities to transmit information via any variation of a physical quantity. There are two reference frameworks for describing network protocols: the OSI Reference Model and the TCP/IP Conceptual Layers. The Carrier Interface follows some industry best practices, and focuses at the Transport, Network and Network Interface layers of the TCP/IP stack, which are aligned with the levels 4,3 and a combination of layers 2 and 1 in the OSI Reference Model, respectively.

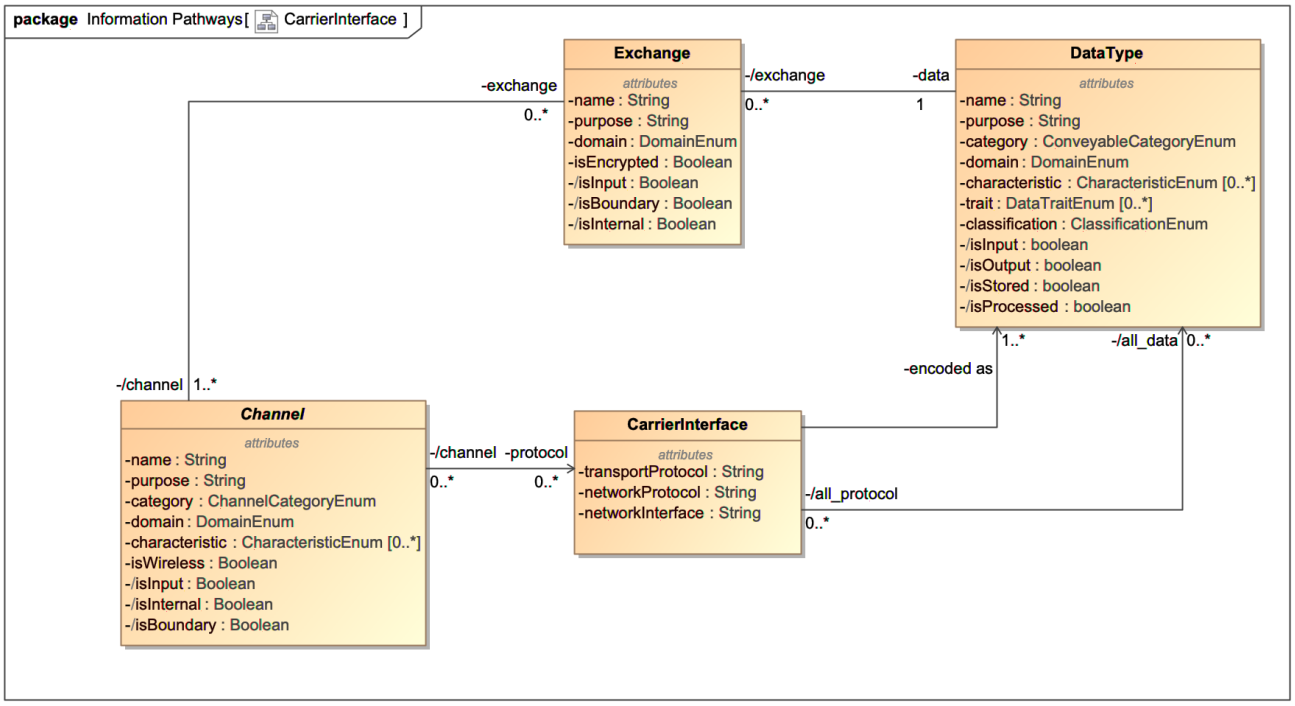

Figure 10 Carrier Interface

Generalization

InformationPathway

Properties

transport protocol:String	Specifies the transport protocols involved in the Carrier
network protocol:String	Specifies the network protocol involved in the Carrier
network interface:String	Specifies the network interface involved in the Carrier

Associations

encoded as:DataType[1..*]	Specifies Carrier Data involved in the Carrier (Conveying Element).

Semantics

SPECTRA is aligned with some industry best practices for representing protocols in models. SPECTRA does not provide a formal list of Carrier Traits (although it provides a formal list of Security and Maintenance Traits, and some generic Application Traits common to many a specific application domain where cyber and cyber-physical systems are developed).
A transport protocol can be identified as e.g. “TCP”, “UDP”,
A network protocol can be identified as e.g. “MIL-STD-188-164A”, “CANopen”, “TCP/IP”, “PWM”, “IPV4”, “IPv6”, “IPv4/IPv6”, “RTP”, “MIL-STD-1553B”, “RS-232”, “NMEA 0183”, “ARINC 429”
A network interface can be identified as e.g. “RF for SATCOM”, “CAN Bus”, “Ethernet”, “MIL-STD-1553”, “SONET/SDH”, “IEEE 1394”, “Wi-Fi”

Compliant tools shall enforce protocol identification in Carrier Interface elements.

The information conveyed by Exchanges shall be identified as Operational Data (defined as the union of Application, Maintenance and Security Domains) or a Resource. Carrier Data shall be associated with Carrier Interfaces (and through them - to Conveying Elements that support Flows), if needed.

Proper identification of the elements that relate to the Application Domain vs Carrier Domain (and also vs Security Domain vs Maintenance Domain) is essential for the proper interpretations of the model for the purposes of a cybersecurity risk assessment of the corresponding SOI. Current industry best practices are often insufficient to make such distinction. See also section on Domain Characteristics.

9.2.16 Collaborative Element (abstract)

Collaborative Element represents an element that is defined through a set of dependencies to other elements.

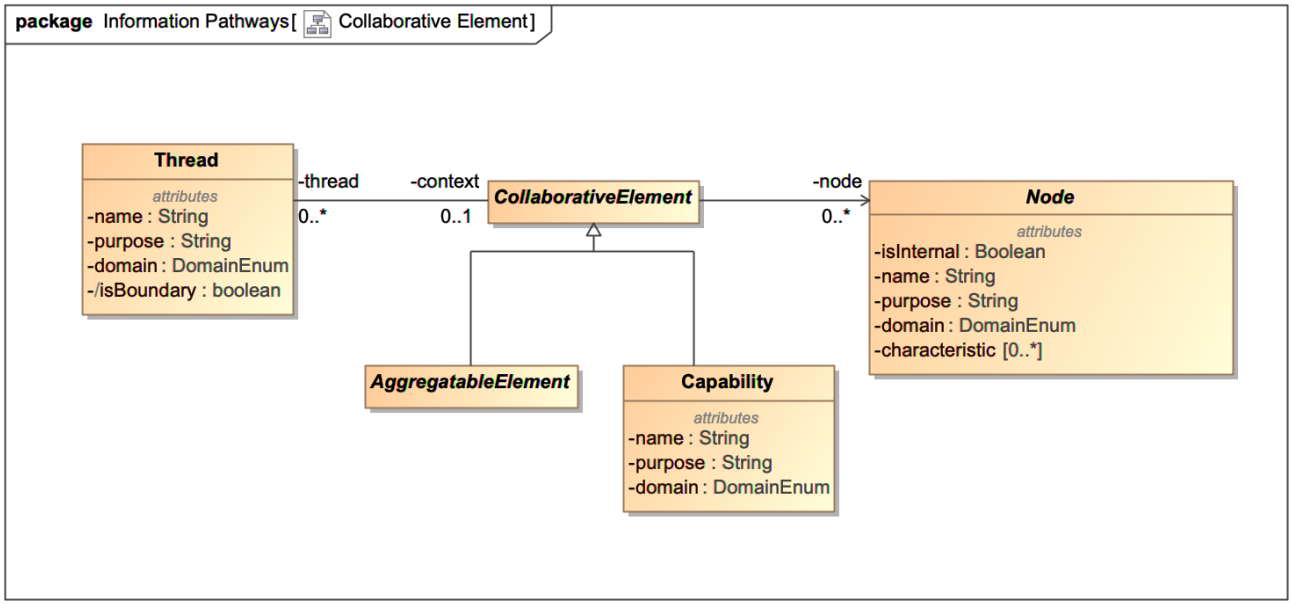

Figure 11 Collaborative Element

Properties

thread: Thread[0..*]	Specifies zero or more threads that define the collaboration
node:Node[0..*]	Zero or one node configuration that includes the replaceable units involved in the collaboration

9.3 Conveyable Elements Package

While the Information Pathways package describes the Core Assertions related to the information bearers of the SOI, the Conveyable Elements addresses the elements of Digital Information (as well as some physical items) involved in the missions and capabilities supported by the SOI.

This package addresses the “WHAT/To WHAT” clauses in assertions about the SOI. For example, “Failure of the Encrypted VHF radio causes exposure of the position information”. Here “position information” is the “WHAT/To WHAT” clause, the “Encrypted VHF Radio” is the “WHERE” clause, and the “Failure of…” is the “Impacting WHAT” clause. “exposure of …” is a “consequence cause”.

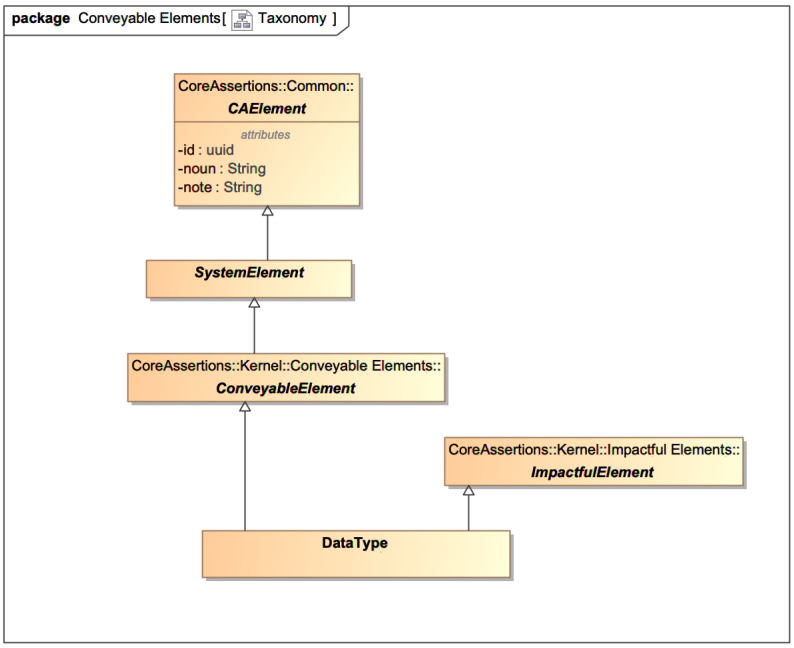

Figure 12 Taxonomy of the Conveyable Element package

9.3.1 Conveyable Element (abstract)

Conveyable Element - any type of thing that is conveyed between replaceable units that allows a flow of information or other items.

9.3.2 DataType

Information Type – a digital information item that can flow between replaceable units and/or be processed by replaceable units and/or be stored at one or more replaceable units. For the purpose of representing cyber-physical system, the DataType class of the Core Assertions metamodel can also represent analog information items, as well as non-information items (physical resources).

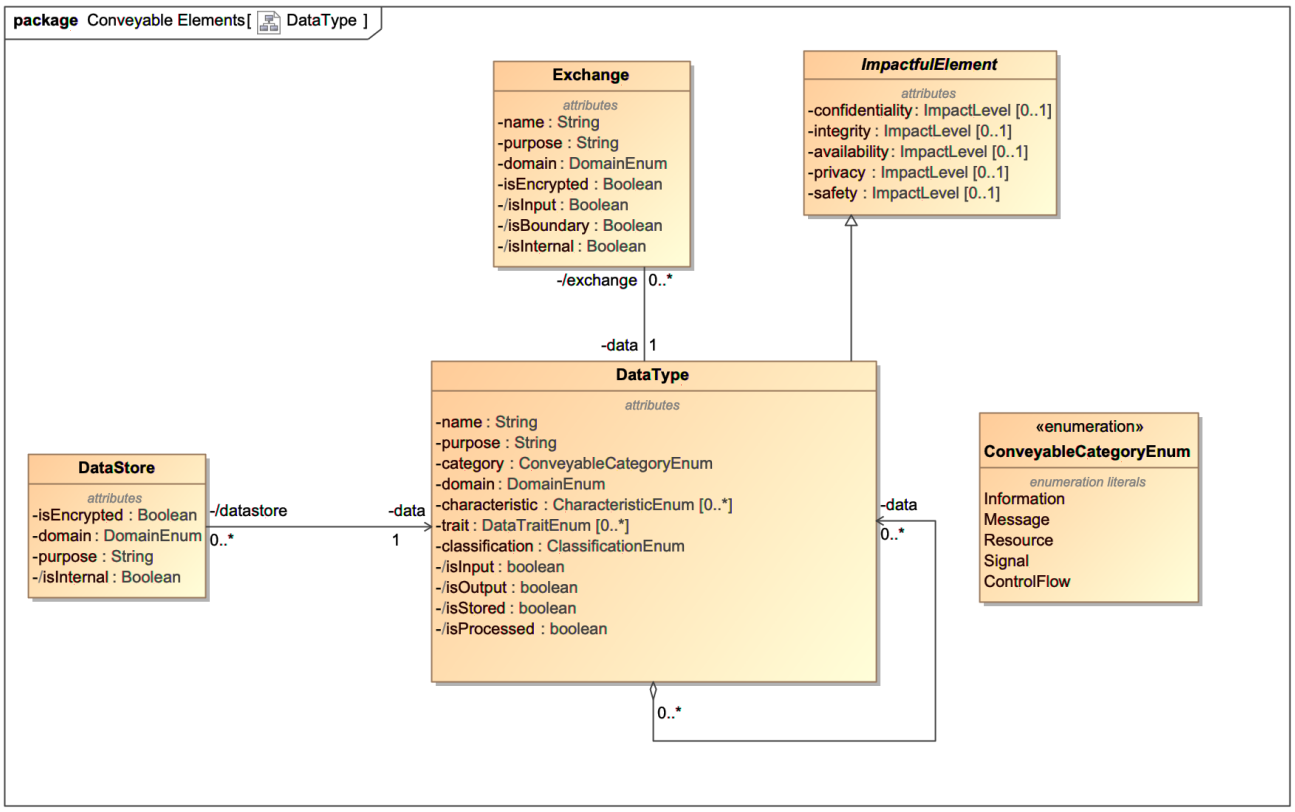

Figure 13 DataType

Generalization

ConveyableElement

Properties

name:String	Specifies the identifying name of the element
purpose:String	Provides the description of the element
category:ConveyeableCategoryEum	Specifies the meaning of the element
domain:DomainEnum	Specifies the domain of the element
characteristic:ChracteristicEnum[0..*]	Specifies the unique characteristics of the element
trait:DataTraitEnum[0..*]	Specifies the unique traits of the element as DataTrainEnum
classification:ClassificationEnum	Specifies the classification markup of the element
isInput:Boolean	(optional) Specifies whether this item is used in the inbound, boundary-crossing flows .
isOutput:Boolean	(optional) Specifies whether this item is used in the outbound, boundary-crossing flows .
isStored:Boolean	(optional) Specifies whether this item is stored in any datastores
isProcessed:Boolean	(optional) Specifies whether this item is processed or transformed by Function Units (other than the ones performing basic send, receive, store or retrieve behavior).

Associations

data:DataType[0..*]	Specifies child datatype for a composite item

Analytics

- The value of isInput is derived. A datatype is considered input (isInput=true) if there exists at least one exchange where producer is external and consumer is internal and the data of the exchange matches the name of the datatype. The value in the isInput column shall match the value derived for the rest of the assertions, based on the isInternal values of the units.
- The value of isOutput is derived. A datatype is considered output (isOutput=true) if there exists at least one exchange where consumer is external and producer is internal and the data of the exchange matches the name of the datatype. The value in the isOutput column shall match the value derived for the rest of the assertions, based on the isInternal values of the units.
- The value of isProcessed is derived. A datatype is considered processed (isProcessed=true) if there exists at least one flow (of a functional thread) where isProcessed=true and the data of the exchange matches the name of the datatype. The value in the isProcessed column shall match the value derived for the rest of the assertions, based on the definitions of the functional threads.
- The value of isStored is derived. A datatype is considered output (isStored=true) if there exists at least one datastore where the data of the datastore matches the name of the datatype. The value in the isStored column shall match the value derived for the rest of the assertions, based on the definitions of the datastores.

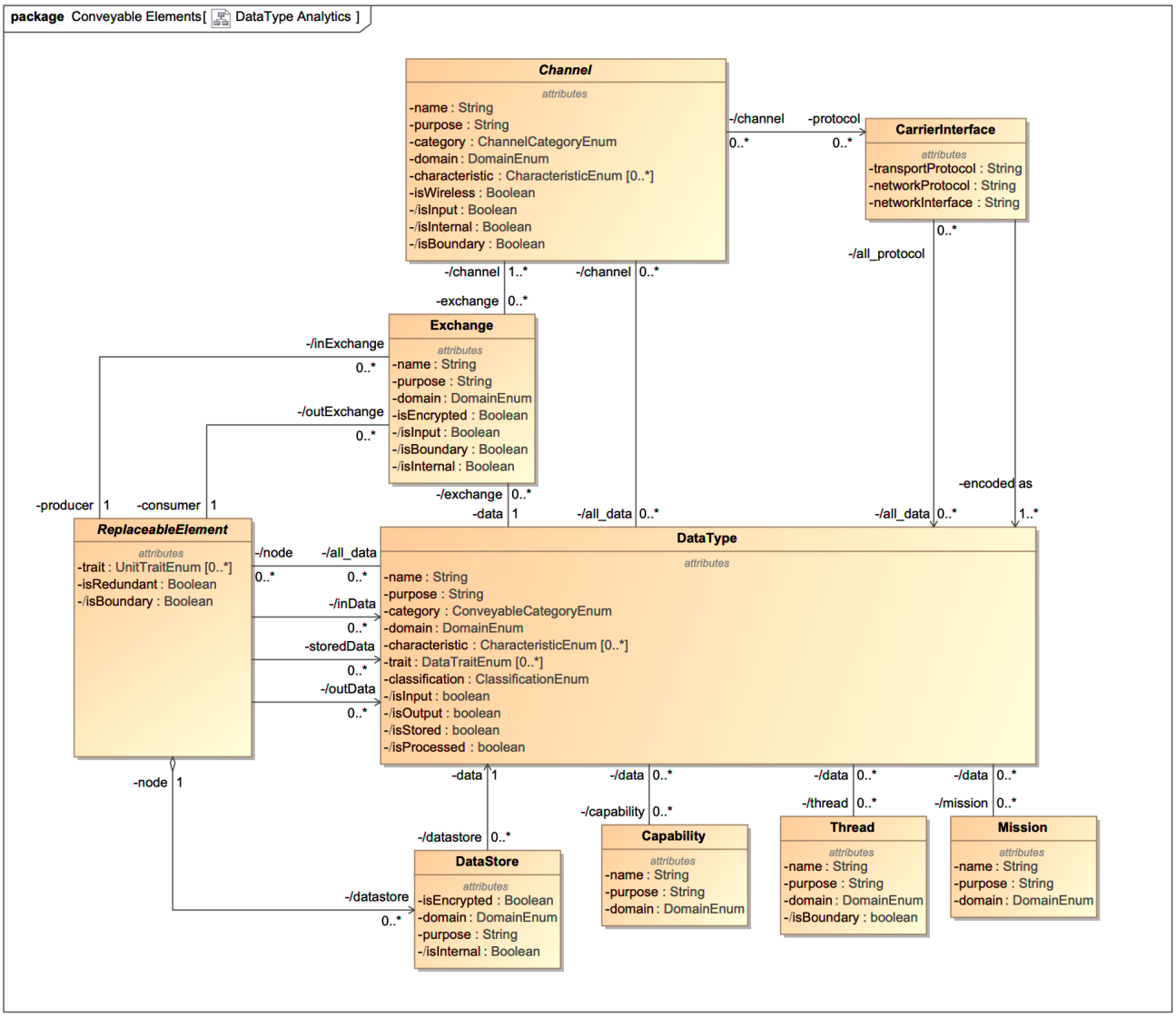

Figure 14 DataType Analytics

Semantics

When an DataType element specified isInput=true, the extracted facts shall include at least one Exchange that conveys the element, and where the Exchange is defined as inbound and border-crossing.
When an DataType element specified isOutput=true, the extracted facts shall include at least one Exchange that conveys the element, and where the Exchange is defined as outbound and border-crossing.
When an DataType element specified isStored=true, the extracted facts shall include at least one Datastore that stores the element.
When an DataType element specified isProcessed=true, the extracted facts shall include at least one Function that either consumes the element, or retrieves the element from a Datastore, and where the Function is defined as implementing a processing or transforming behavior (other than an implicit send, receive, store or retrieve).

In terms of the CoreAssertions, these attributes imply on of the following additional assertions to the ones implied by the specific subclass:
isInput(Id)
isOutput(Id)
isStored(Id)

isProcessed(Id)

When the DataType element has an additional stereotype “internal”, the following assertion is added:
isInternal(Id,’true’)
When the DataType element has an additional stereotype “external”, the following assertion is added:
isInternal(Id,’false’)

9.4 Impactful Elements Package

While the Information Pathways package describes the Core Assertions related to the information bearers of the SOI, and the Conveyable Elements addresses the elements of Digital Information (as well as some physical items) conveyed by the parts of the SOI through Conveying Elements, this package addresses the Core Assertions related to missions and capabilities supported by the SOI.

This package addresses the “WHAT/Impacting WHAT” clauses in assertions about the SOI. For example, “Failure of the VHF radio causes loss of the capability to communicate”. Here “capability to communicate” is the “impacting what?” clause, the “VHF Radio” is the “WHERE” clause, and the “Failure of…” is the “WHAT/Impacting WHAT” clause. “loss of …” is a “consequence cause”. Threat-based analytics will consider various attacks that can cause a certain “impact” situation.

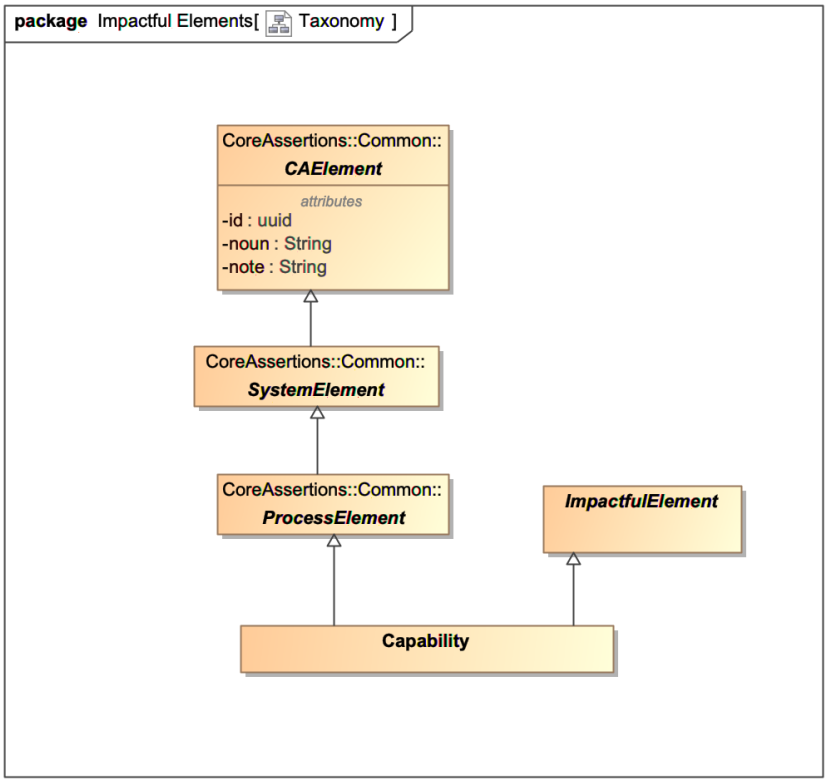

Figure 15 Taxonomy of the Impactful Element package

9.4.1 Impactful Element (abstract)

Certain elements of the SOI may be sensitive to situations where there is a “loss” of the Confidentiality, Integrity, Availability, Privacy, or Safety cybersecurity objectives. An optional property measuring the Impact Level to the SOI’s stakeholders represents each of these objectives.

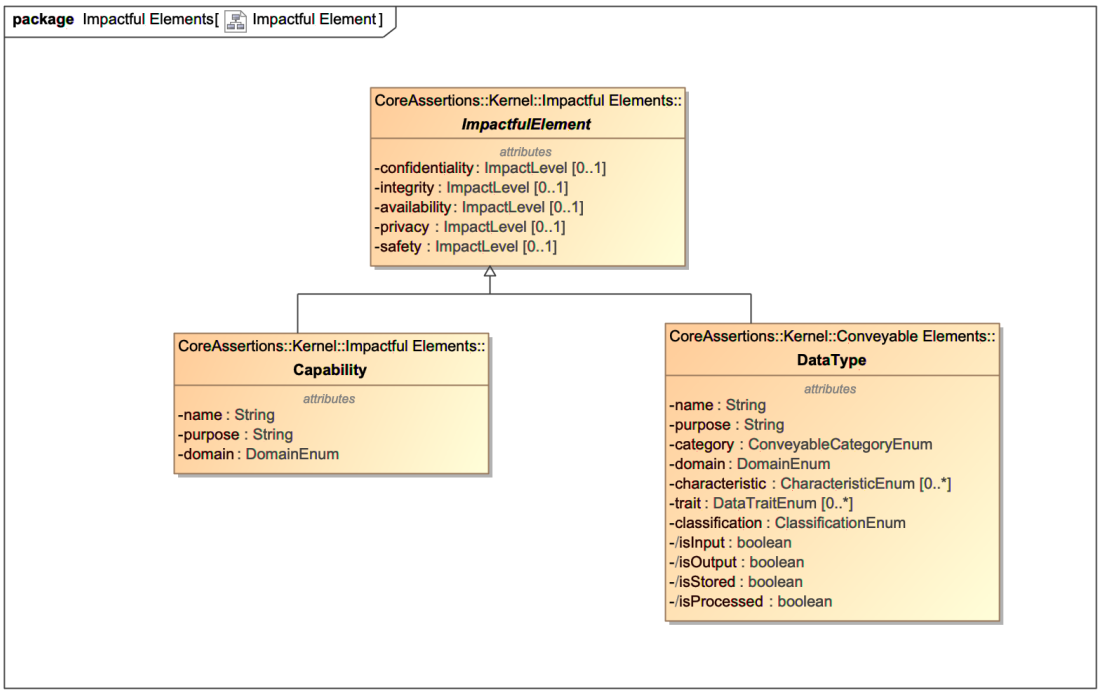

Figure 16 Impactful Element

Properties

confidentiality:Impact Level[0..1]	Represents sensitivity to the loss of confidentiality (if applicable).
integrity:Impact Level[0..1]	Represents sensitivity to the loss of integrity (if applicable).
availability:Impact Level[0..1]	Represents sensitivity to the loss of availability (if applicable).
privacy:Impact Level[0..1]	Represents sensitivity to the loss of private and personal identifiable information (if applicable).
safety:Impact Level[0..1]	Represents sensitivity to the loss of safety (if applicable).

9.4.2 Impact Level (enumeration)

The Impact Level measure is aligned with NIST SP-800-30 (Impact, Table H-3). This can be mapped to a 3-range measure {high, moderate, low} used in FIPS-199 by considering {very high, high} as FIPS high, and {very low, low} as FIPS low. The definitions below follow NIST-800-30.

Enumeration Literals

very high	The loss of confidentiality, integrity, availability, privacy, or safety could be expected to have **multiple severe or catastrophic** adverse effects on organizational operations, organizational assets, or individuals.
high	The loss of confidentiality, integrity, availability, privacy or safety could be expected to have a **severe or catastrophic** adverse effect on organizational operations, organizational assets, or individuals.
moderate	The loss of confidentiality, integrity, availability, privacy or safety could be expected to have a **serious** adverse effect on organizational operations, organizational assets, or individuals.
low	The loss of confidentiality, integrity, availability, privacy or safety could be expected to have a **limited** adverse effect on organizational operations, organizational assets, or individuals.
very low	The loss of confidentiality, integrity, availability, privacy or safety could be expected to have a **negligible** adverse effect on organizational operations, organizational assets, or individuals.

9.4.3 Capability

A Capability is a strategic element that refers to an enterprise's ability to achieve a desired effect realized through a combination of ways and means (e.g. capability configuration that involves a set of collaborating performers and channels) along with specified measures. In a SysML model, a capability can be represented as an activity, a block or a use case. A capability may depend on other capabilities, and usually involves collaboration and flows among multiple replaceable units

Generalization

ProcessElement

Properties

name:String	Specifies the identifying name of the element
purpose:String	Provides the description of the element
domain:DomainEnum	Specifies the domain of the element

Associations

capability:Capability[0..*]	Specifies child capability for a composite item
node:ReplaceableElement[0..*]	Nodes that are involved in this capability
dependsOn:Capability[0..*]	Specify capabilities that the current one depends on

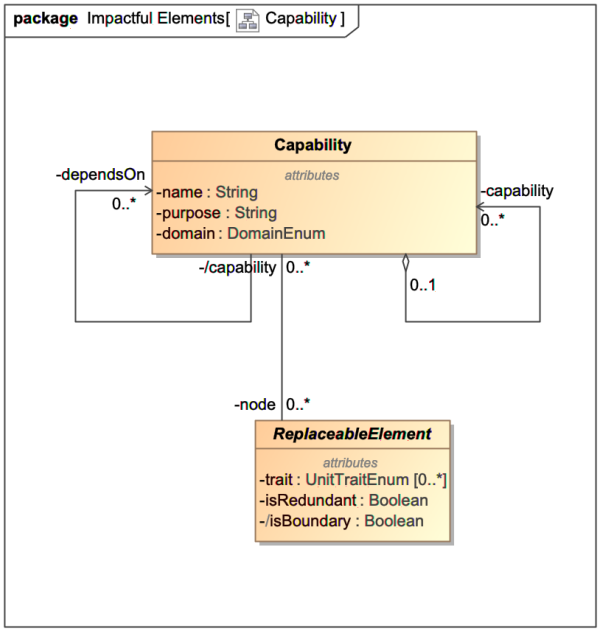

Figure 17 Capability

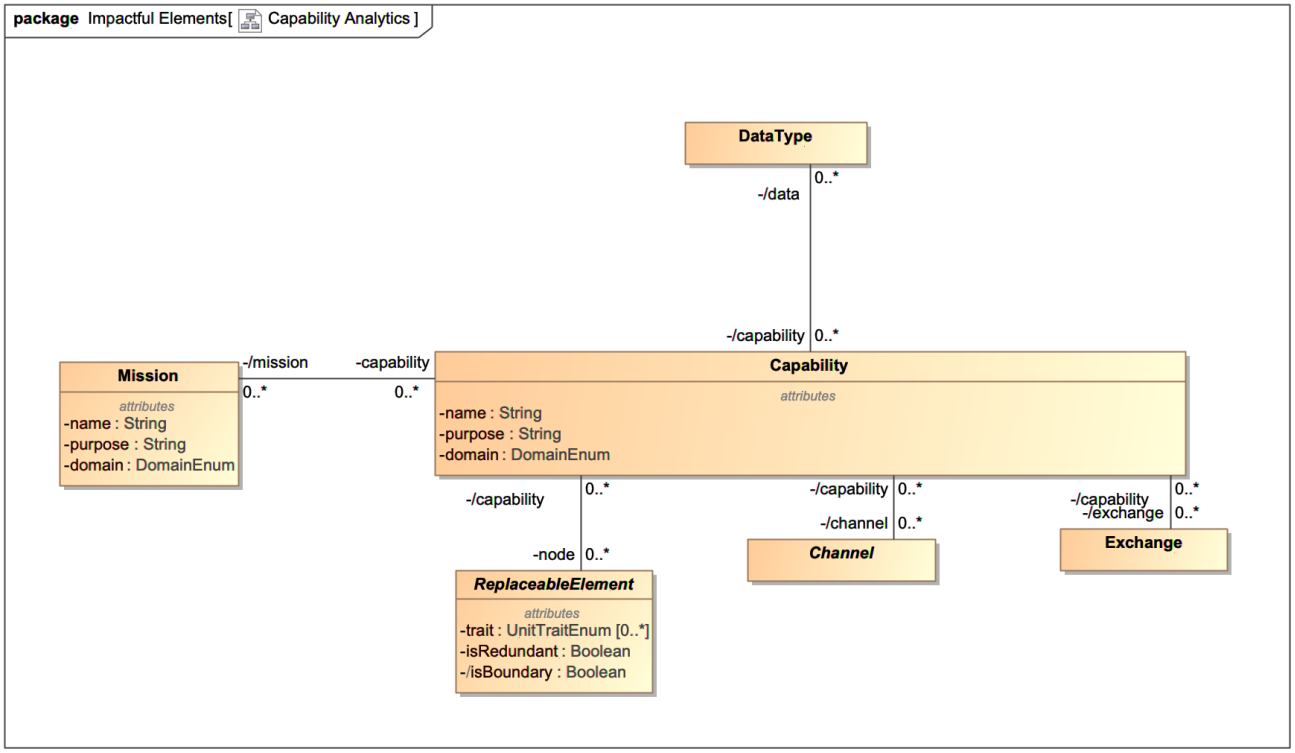

Figure 18 Capability Analytics

9.5. Threads Package

While the Information Pathways package describes the Core Assertions related to the information bearers of the SOI, and the Conveyable Elements addresses the elements of Digital Information (as well as some physical items) conveyed by the parts of the SOI through Conveying Elements, this packages the relative timing of activities performed by the SOI as it supports missions and capabilities. This package identifies the Core Assertions in terms of functional threads through the system, as collections of Flows ordered by “happens before”.

This package addresses the “WHEN” clause.

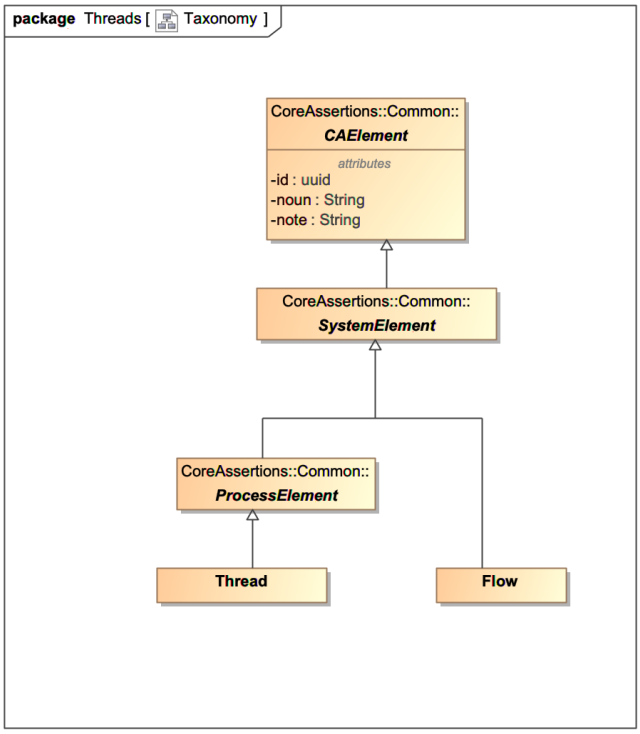

Figure 19 Taxonomy of the Threads package

9.5.1 Functional Thread

Functional Thread - end-to-end ordered collection of flows usually corresponding to an emergent function or a capability.

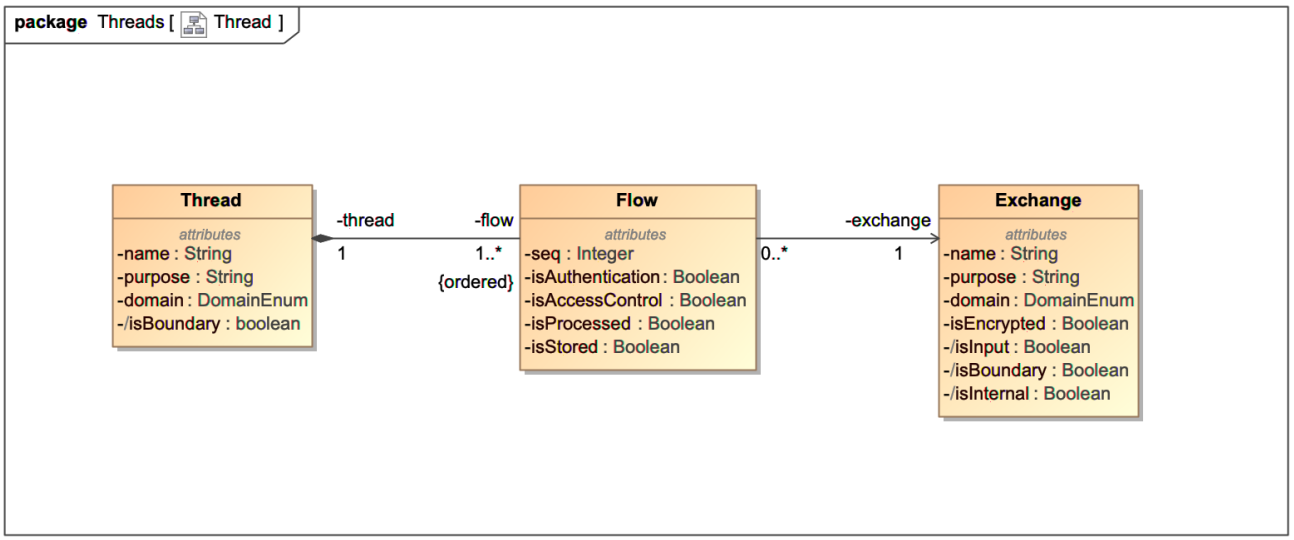

Figure 20 Functional Thread

Generalization

ProcessElement

Properties

seq:Integer	Specifies the position of the flow in the sequence of exchanges in relation to the flows that happened before, after or concurrently with the current element
isAuthentication:Boolean	Specifies whether the flow involves authentication
isAccessControl:Boolean	Specifies whether the flow involves access control
isProcessed:Boolean	Specifies whether the flow involves processing
isStored:Boolean	Specifies whether the flow involves a datastore (read or write)

Properties

flow:Flow[1..*] {ordered}	One or more ordered Flows involved in the thread

Semantics

The ordering of the Flow elements in a Functional Thread I defined by the seq property.

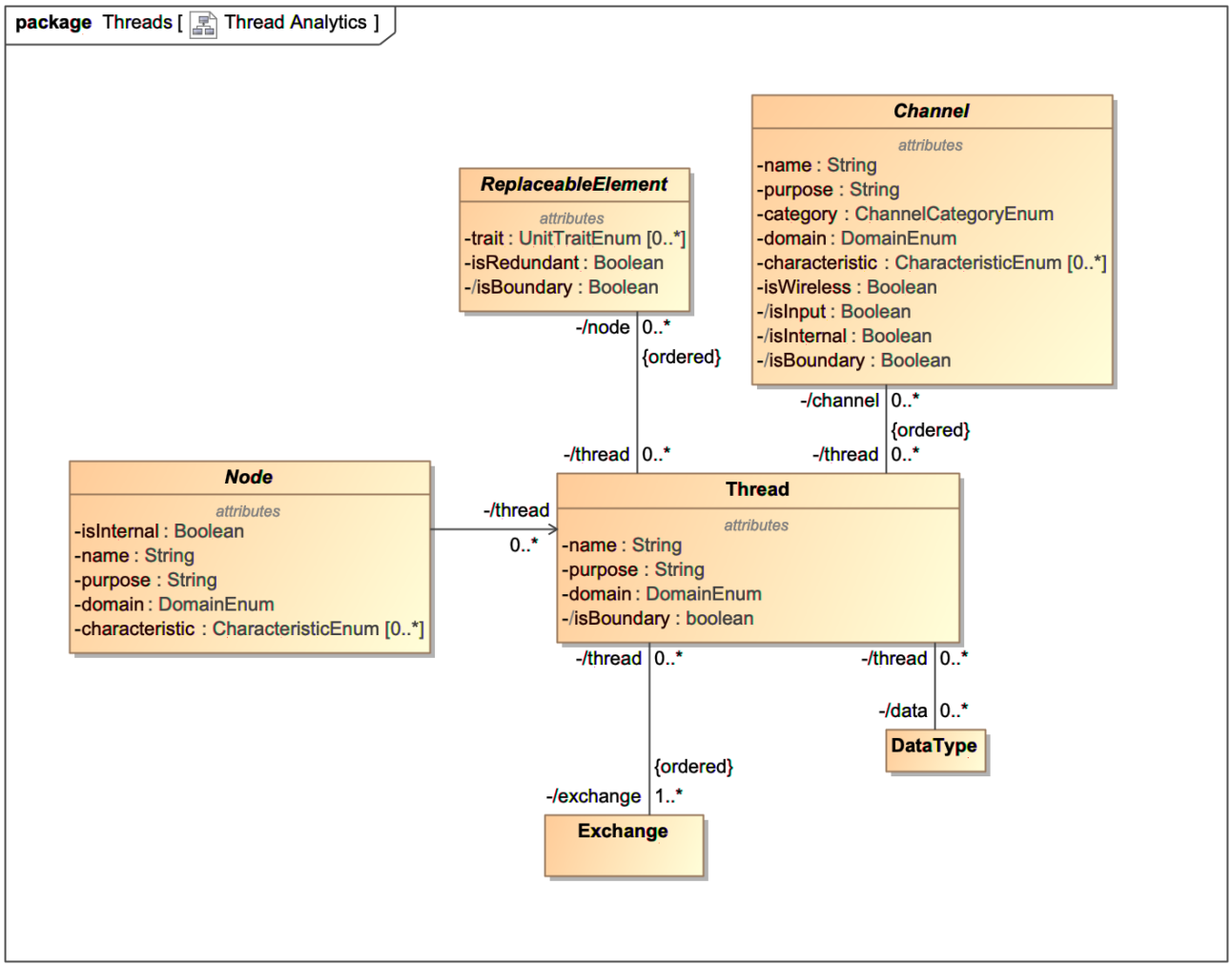

Figure 21 Thread Analytics

This page intentionally left blank.

10 SPECTRA Artifacts package

10.1 Overview

Artifacts (together with Traits and Characteristics) provide a shared vocabulary to make assertions about the nature of the elements of SOI. Artifacts describe the common assertions related to the nature of the replaceable elements of the SOI, their role in missions and capabilities supported by the SOI, and their place in the supply chain enabling the SOI. Artifacts address assertions related to the unique technologies of the replaceable elements.

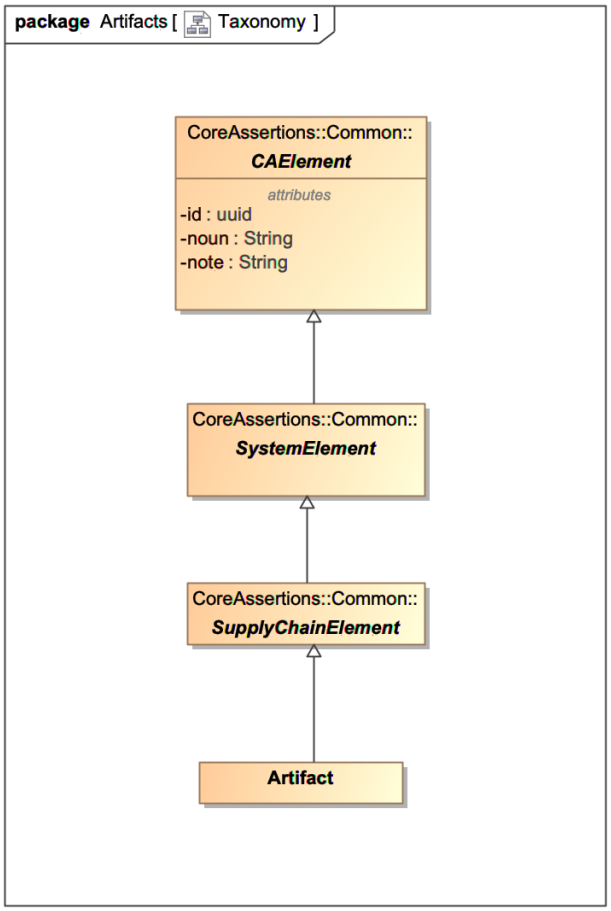

Figure 22 Taxonomy of the Artifacts package

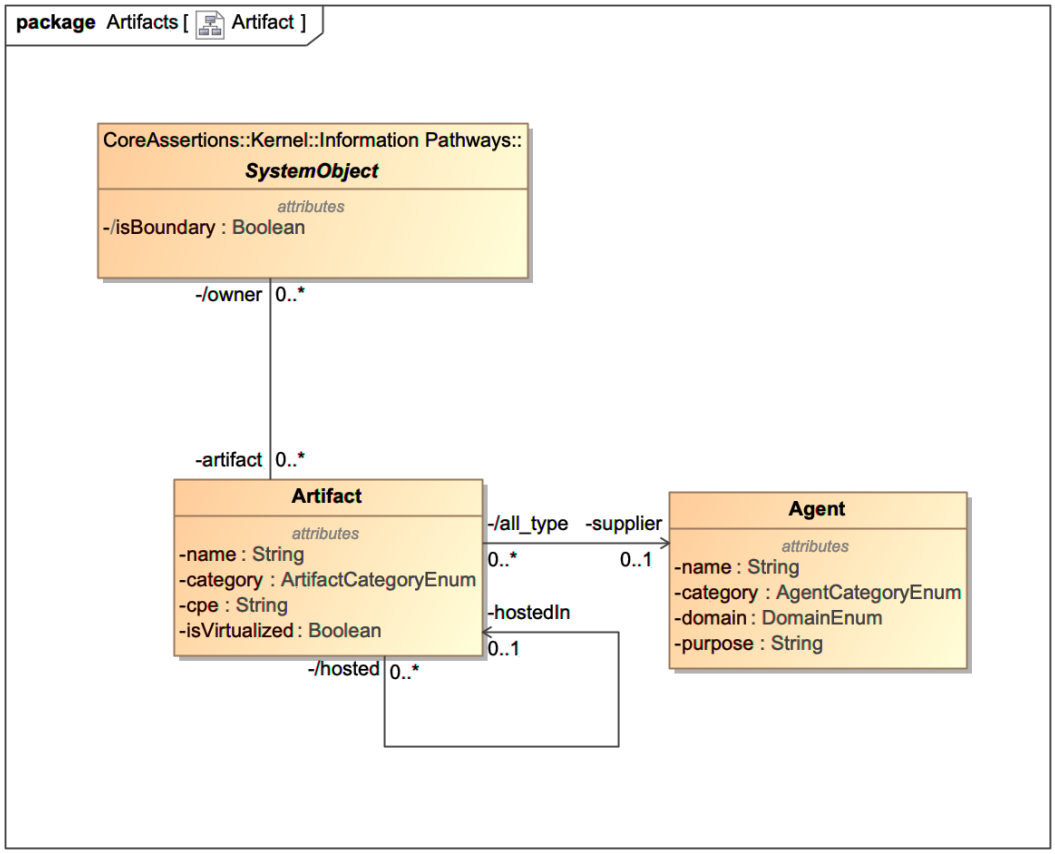

Figure 23 Artifacts

10.2 Artifacts

10.2.1 Artifact

Artifact - Artifact represents a unique attack opportunity and is defined as a combination of a characteristic and a reference to a unique suppliable artifact type called "Artifact image". Characteristics are defined as concrete subclasses of Artifact. From the supply chain perspective (BOM/SBOM), an artifact is something owned by an attackable unit (hardware, firmware, custom software, operating system, media, etc.). Artifact is associated with a uniquely identified suppliable element called "artifact image". An artifact image can be associated with multiple replaceable units. An artifact image is further associated with a supplier and a CPE code. Note that several other important elements associated with a replaceable unit can be derived from the model, such as facility, operational procedures, personnel, policy, controls, etc. Specific artifact subclasses represent individual attack opportunities as well as frequently used common suppliable element categories.

Generalization

SupplyChainElement

Properties

name:String	Specifies the identifying name of the element
category:ArtifactCategoryEnum	Specifies the meaning of the element by a reference of a specific enumeration literal

cpe:String	(optional) Specifies the Common Platform Enumeration code of the artifact
isVirtualized:Boolean	true when the infrastructure element is virtualized

Associations

hostedIn:Artifact[0..1]	Specifies Cloud Infrastructure element that hosts the infrastructure element
supplier:Agent[0..1]	(optional) specifies supplier for this artifact

10.2.2 ArtifactCategoryEnum (Enumeration)

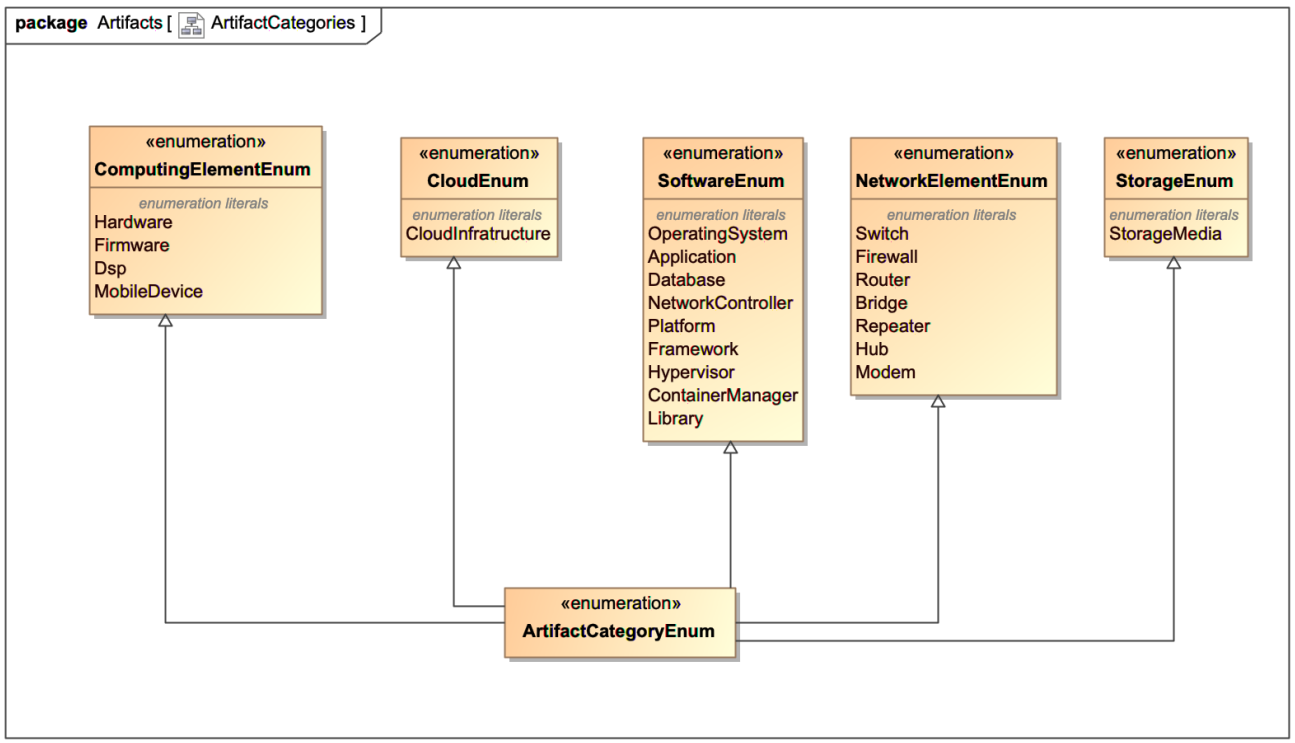

Figure 24 Artifact Categories

Artifacts (together with Traits and Characteristics) provide a shared vocabulary to make assertions about the nature of the elements of SOI. Artifacts describe the common assertions related to the nature of the replaceable elements of the SOI, their role in missions and capabilities supported by the SOI, and their place in the supply chain enabling the SOI. Artifacts address assertions related to the unique technologies of the replaceable elements.

Computing Element

Computing element is an abstract element that represents computing hardware. Detailed semantics is provided by concrete subclasses.

Table 5 Enumeration Literals for Computing Elements

Enum Value	Definition
DSP	Dsp - digital signal processing unit. A digital signal processor (DSP) is a specialized microprocessor chip, with its architecture optimized for the operational needs of digital signal processing. The digital signals processed in this manner are a sequence of numbers that represent samples of a continuous variable in a domain such as time, space, or frequency.
Hardware	Hardware - Computer hardware includes the physical parts of a computer, such as the central processing unit (CPU), random access memory (RAM), motherboard, computer data storage, graphics card, sound card, and computer case. It includes external devices such as a monitor, mouse, keyboard, and speakers. Aligned with Cyclone DX component type “device”
Firmware	Firmware - firmware is software that provides low-level control of computing device hardware. For a relatively simple device, firmware may perform all control, monitoring and data manipulation functionality. For a more complex device, firmware may provide relatively low-level control as well as hardware abstraction services to higher-level software such as an operating system. Firmware is found in a wide range of computing devices including personal computers, phones, home appliances, vehicles, computer peripherals and in many of the digital chips inside each of these larger systems. Firmware is stored in non-volatile memory – either read-only memory (ROM) or programmable memory such as EPROM, EEPROM, or flash. Changing a device's firmware stored in ROM requires physically replacing the memory chip – although some chips are not designed to be removed after manufacture. Programmable firmware memory can be reprogrammed via a procedure sometimes called flashing. Aligned with Cyclone DX component type “firmware”
MobileDevice	Mobile Device (NIST-800-53) is a portable computing device that has a small form factor such that it can easily be carried by a single individual; is designed to operate without a physical connection (e.g., wirelessly transmit or receive information); possesses local, non-removable data storage; and is powered on for extended periods of time with a self-contained power source. Mobile devices may also include voice communication capabilities, on-board sensors that allow the device to capture (e.g., photograph, video, record, or determine location) information, and/or built-in features for synchronizing local data with remote locations. Examples include smartphones, tablets, and e-readers. Mobile devices often emphasize wireless networking to both the local area networks and Internet and to other devices in their vicinity.

Cloud Infrastructure

Table 6 Enumeration Literals for Cloud Infrastructure

Enum Value	Definition
CloudInfrastructure	Cloud Infrastructure is an element that represents the hosting environment for virtualized infrastructures.

Software

Software is an abstract element that represents computer programs that instruct the execution of a computing device. Detailed semantics relevant to cyber and cyber-physical systems are provided by the subclasses.

Table 7 Enumeration Literals for Software

Enum Value	Definition
OperatingSystem	Operating system - An operating system (OS) is system software that manages computer hardware and software resources and provides common services for custom computer programs and other system software. Aligned with Cyclone DX component type “operating-system”
Application	Application- application/custom software. artifact comprising only instructions for computer hardware, excluding operating system software, database and other common utility systems. Aligned with Cyclone DX component type “application”
Database	Database - database, SQL or no-SQL relational, key-valued, graph, etc. e.g. Oracle, Postgres, Redis, etc.
NetworkController	Network controller - telecommunication network technologies based on physically wired, optical, and wireless radio-frequency methods that may be arranged in a variety of network topologies. The transmission media (often referred to in the literature as the physical medium) used to link devices to form a computer network include electrical cable, optical fiber, and free space. A network interface controller (NIC) is computer hardware that connects the computer to the network media and has the ability to process low-level network information. For example, the NIC may have a connector for plugging in a cable, or an aerial for wireless transmission and reception, and the associated circuitry.
Framework	Framework - A software framework. Aligned with Cyclone DX component type "framework"
Platform	Platform - A runtime environment which interprets or executes software. This may include runtimes such as those that execute bytecode or low-code/no-code application platforms. Aligned with Cyclone DX component type "platform"
Hypervisor	Hypervisor element represents a type of computer software that creates and runs virtual machines.
ContainerManager	Container manager - containerization is operating system-level virtualization or application-level virtualization over multiple network resources so that software applications can run in isolated user spaces called containers in any cloud or non-cloud environment, regardless of type or vendor. containerization technology has been widely adopted by cloud computing platforms like Amazon Web Services, Microsoft Azure, Google Cloud Platform, and IBM Cloud. Container orchestration or container management is mostly used in the context of application containers. Implementations providing such orchestration include Kubernetes and Docker swarm.
Library	Library – A software library. Aligned with Cyclone DX component type "library"

Network Element

The nodes of a computer network can include personal computers, servers, networking hardware, or other specialized or general-purpose hosts. Computer networks may be classified by many criteria, including the transmission medium used to carry signals, bandwidth, communications protocols to organize network traffic, the network size, the topology, traffic control mechanisms, and organizational intent.

Table 8 Enumeration Literals for Network Elements

Enum Value	Definition
Switch	Network bridges and network switches are distinct from a hub in that they only forward frames to the ports involved in the communication whereas a hub forwards to all ports. Bridges only have two ports,Abut a switch can be thought of as a multi-port bridge. Switches normally have numerous ports, facilitating a star topology for devices, and for cascading additional switches.
Firewall	A firewall is a network device or software for controlling network security and access rules. Firewalls are inserted in connections between secure internal networks and potentially insecure external networks such as the Internet. Firewalls are typically configured to reject access requests from unrecognized sources while allowing actions from recognized ones.
Router	A router is an internetworking device that forwards packets between networks by processing the addressing or routing information included in the packet. The routing information is often processed in conjunction with the routing table. A router uses its routing table to determine where to forward packets and does not require broadcasting packets which is inefficient for very big networks.
Bridge	Network bridges and network switches are distinct from a hub in that they only forward frames to the ports involved in the communication whereas a hub forwards to all ports. Bridges only have two ports but a switch can be thought of as a multi-port bridge. Switches normally have numerous ports, facilitating a star topology for devices, and for cascading additional switches.
Repeater	A repeater is an electronic device that receives a network signal, cleans it of unnecessary noise and regenerates it. The signal is retransmitted at a higher power level, or to the other side of obstruction so that the signal can cover longer distances without degradation.
Hub	An Ethernet repeater with multiple ports is known as an Ethernet hub. In addition to reconditioning and distributing network signals, a repeater hub assists with collision detection and fault isolation for the network. Hubs and repeaters in LANs have been largely obsoleted by modern network switches.
Modem	Modems (modulator-demodulator) are used to connect network nodes via wire not originally designed for digital network traffic, or for wireless. To do accomplish this, one or more carrier signals are modulated by the digital signal to produce an analog signal that can be tailored to give the required properties for transmission.

Storage Media

Table 9 Enumeration Literals for storage Media

Enum Value	Definition
StorageMedia	Storage Media - Computer data non-volatile secondary storage refers to a technology consisting of computer components and recording media that are used to retain digital data. Secondary storage (also known as external memory or auxiliary storage) differs from primary storage in that it is not directly accessible by the CPU. The computer usually uses its input/output channels to access secondary storage and transfer the desired data to primary storage (part of hardware). Secondary storage is non-volatile (retaining data when its power is shut off). In modern computers, hard disk drives (HDDs) or solid-state drives (SSDs), rotating optical storage devices, such as CD and DVD drives are usually used as secondary storage. Other examples of secondary storage technologies include USB flash drives, floppy disks, magnetic tape, paper tape, punched cards, and RAM disks.

11 SPECTRA Access package

11.1 Overview

The Access package addresses the “BY WHOM?” clauses related to SOI (in addition to “WHERE” and “To WHAT” clauses introduced in the Kernel Package). Assertions related to access are important for understanding the trust zones of the SOI, its internal attack surface as well as the external attack surface in the context of advanced persistent threats. Stereotype is the Access package that allows identifying certain blocks as the “agents” who are trusted to access some capabilities of the system. Access assertions are made in the form of agent handles data, agent accesses unit, or agent performs function. In addition, this package also identifies assertions related to the supply chain of the SOI in the form supplier supplies artifact.

When modeling cyber and cyber-physical systems, some elements may already represent technology and human agents. Human elements are represented as ReplaceableUnits with a characteristic Human. Such elements will be part of the exchange fabric and may perform some functions. However, it is often the case that a model will imply more agents that can interact with the SOI, including operators, maintainers, suppliers, as well as attackers impersonating legitimate users or as unique agents. SPECTRA Access package defines some elements that can capture these agents (including a unique vocabulary of names that the stakeholders are using to describe them). Access relationships focus on what capabilities of the SOI may be available to these agents.

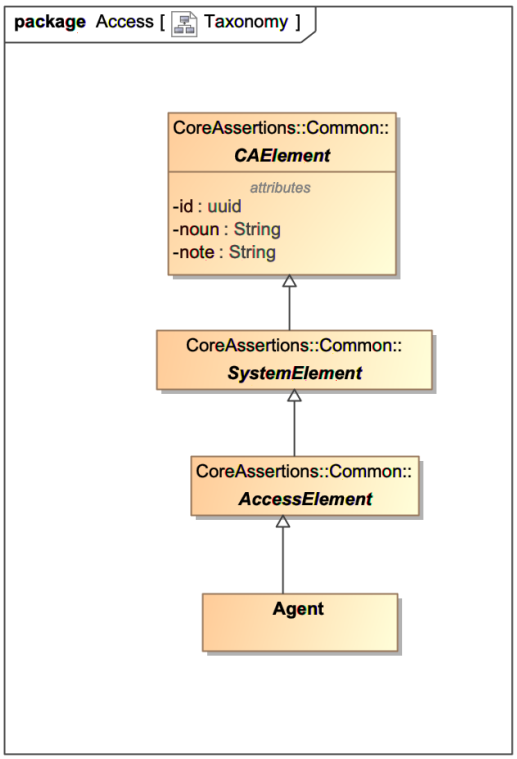

Figure 25 Taxonomy of the Access Package

11.2 Agents and their Access

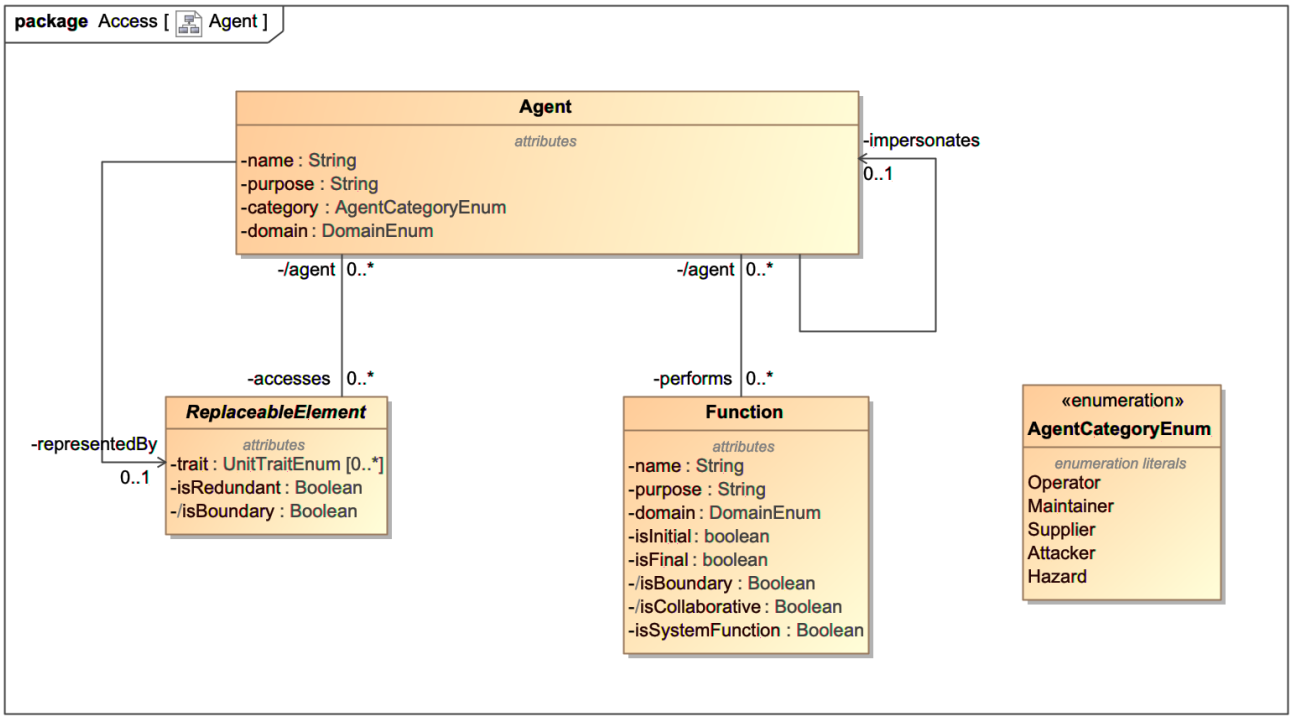

Figure 26 Agent

11.2.1 Agent

Agent - block or part that represents a human being (as a part in a complex information processing enterprise); e.g. an operator; a human can be vulnerable to subversion, deceit, can be clueless, careless or malicious; or can be vulnerable to some health and safety hazard.

Generalization

AccessElement

Properties

name:String	Specifies the identifying name of the element
purpose:String	Provides the description of the element
category:AgentCategoryEnum	Specifies the meaning of the element by reference to a specific enumeration literal
domain:DomainEnum	Specifies the domain of the element

Properties

performs:Functional Element[0..*]	Specifies the functions that the agent can perform
accesses:Replaceable Unit[0..*]	Specifies the Replaceable Units to which the agent has access
impersonates:Agent[0..1]	(optional) for specifying attackers
representedBy:ReplaceableElement[0..1]	(optional) for specifying attackers

11.2.2 Agent Category Enumeration

Table 10 Enumeration Literals for Agent Category

Enum Value	Definition
Operator	An operator is a human can be vulnerable to subversion, deceit, can be clueless, careless or malicious; or can be vulnerable to some a health and safety hazard
Maintainer	An agent performing maintenance. Maintenance operations can be performed at various mission phases, such as during the operational phases (hot patches, dynamic reconfiguration), during dedicated maintenance phases (planned patching or an upgrade), or during the major evolutionary upgrade phases.
Supplier	Supplier - Representation of a supplier of an artifact type (optional, can be uniquely derived in the form of "supplier of an artifact type xxx"), but if a model provides this information, it is important to annotate accordingly, so that this element is not mis-interpreted, and so that fidelity is not compromised
Attacker	Attacker – a representation of an attacker on the SOI, capable and motivated to do harm to the stakeholders of the SOI
Hazard	Hazard - A hazard is a potential source of harm. Substances, events, or circumstances can constitute hazards when their nature would potentially allow them to cause damage to health, life, property, or any other interest of value. In physics terms, a common theme across many forms of hazards is the presence of energy that can cause damage, as it can happen with chemical energy, mechanical energy or thermal energy.

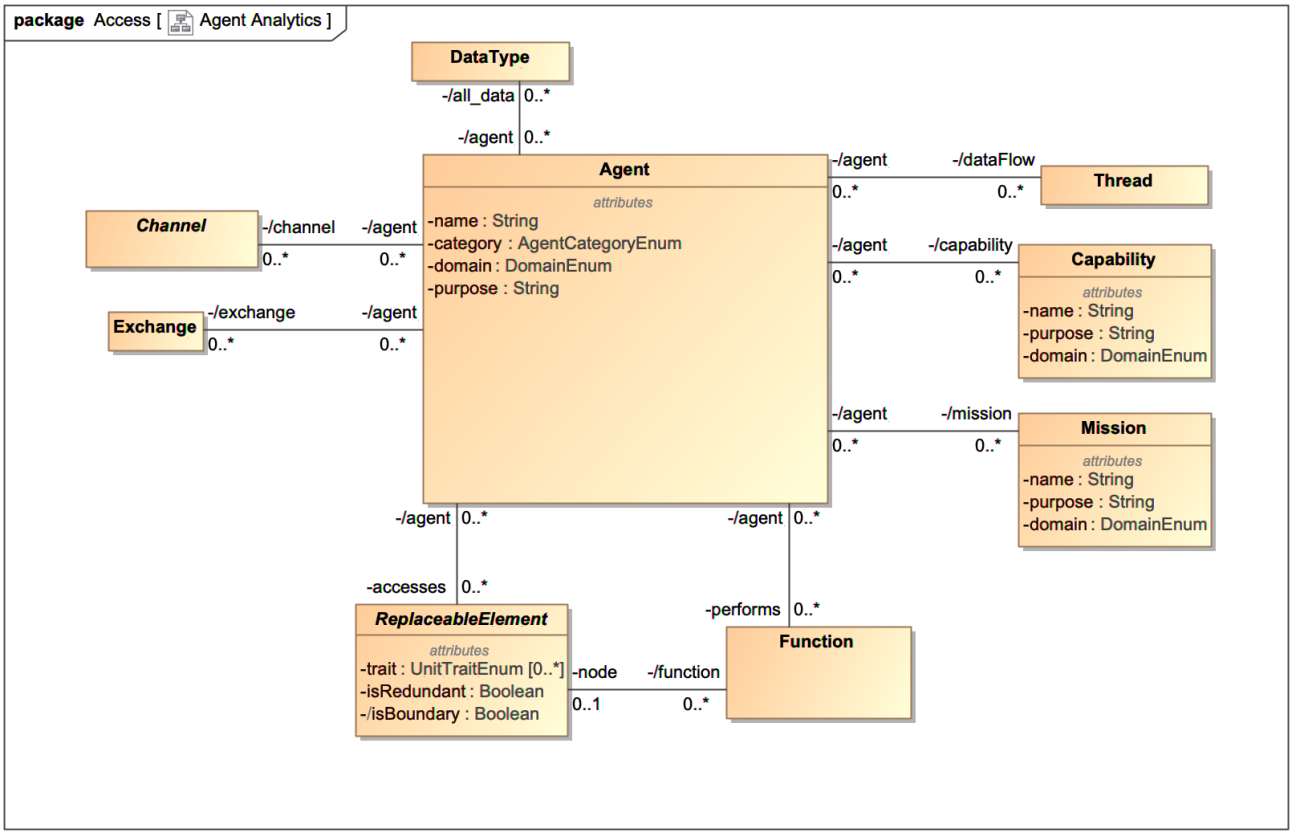

Figure 27 Agent Analytics

This page intentionally left blank.

12 SPECTRA Behavior package

12.1 Overview

Information Pathways package through its emphasis on how digital information types (and some material items) are conveyed between the parts of the SOI, already implies certain “activities” performed by the replaceable units of the SOI namely:

- send an information item (or a material item)
- receive an information item (or a material item)
- store an information item
- retrieve an information item

Material items imply some kind of storage, once a material item is transferred to a replaceable unit, it remains stored in that unit.

Behavior package identifies Core Assertion related to the custom language of other “activities”, performed by the SOI (either directly by one of the replaceable units or through collaboration of several units). Such activities are usually related to processing, converting, transforming various items, producing items, disposing of items, performing computations, making decisions, etc.

The behavior viewpoint of SPECTRA is optional; attacks can be adequately predicted at the purely structural viewpoint offered by information pathways alone; SPECTRA allows identification of functions (and emergent collaborative functions) as refinement of the replaceable units. SPECTRA also allows identification of Mission Tasks (as they are introduced by the Mission Engineering language and are distinct from the Functions and SystemFunctions of the SOI.

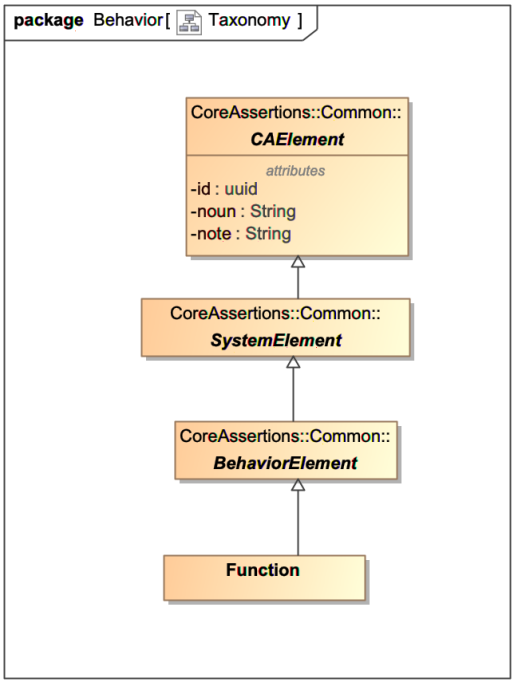

Figure 28 Taxonomy of the Behavior package

Behavior package does not introduce any new clauses; however the identification of Functions refines the “WHERE” and “WHEN” clauses, in comparison to a situation when they are defined only by Replaceable Elements.

12.2 Functions

12.2.1 Function

SysML activity, action or block that represent something that an individual replaceable unit does; function happens in a certain place (some functions are performed by an individual replaceable unit. Function may be denied by an attack, may fail to perform, or may perform at a wrong time, or perform incorrectly, e.g. fire a missile at a wrong target, or may be degraded. A function is the most specific producer/consumer of a flow, and an agent that stores/retrieves data from datastores.

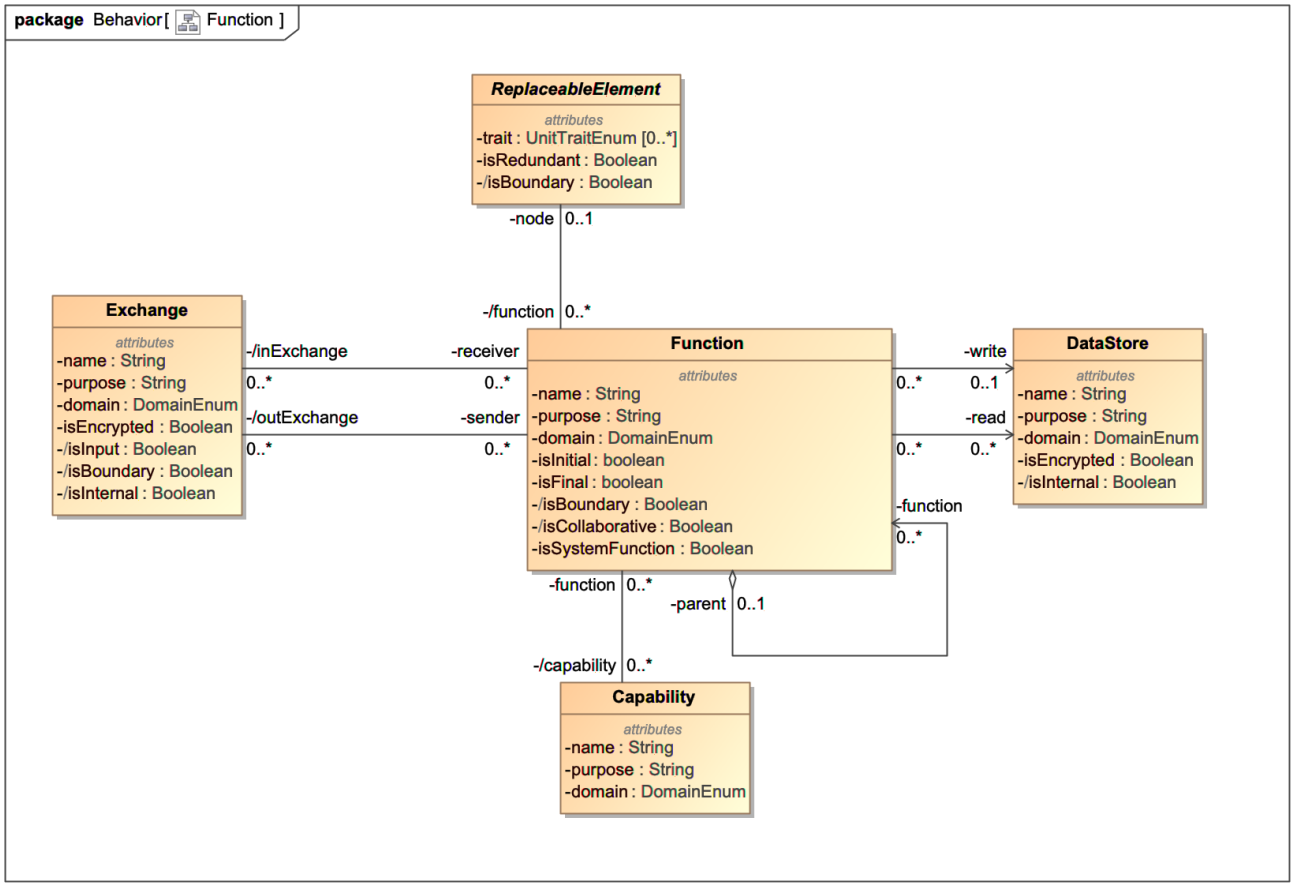

Figure 29 Function

Properties

name:String	Specifies the identifying name of the element
purpose:String	Provides the description of the element
domain:DomainEnum	Specifies the domain of the element
isInitial:Boolean	Specifies whether the function can initiate any functional threads
isFInal:Boolean	Specifies whether the function can terminate any functional threads
/isBoundary:Boolean	(defined) specifies whether the function is part of the technical surface of the SOI

/isCollaborative:Boolean	(derived) specifies whether the function involves collaboration of more than one replaceable unit
isSystemFunction:Boolean	Specifies whether this is a system function

Associations

function:Function[0..*]	Specifies functions into which the current function is decomposed
node:ReplaceableElement[0..1]	Specifies Replaceable Element that performs the function
/inExchange:Exchange[0..*]	Specifies Exchanges that send data to be consumed by the function
/outExchange:Exchange[0..*]	Specifies Exchanges that receive data produced by the function
write:Datastore[0..*]	Specifies the datastores to which the function stores data and/or resources
read:Datastore[0..*]	Specifies the datastores from which the function retries data and/or resources

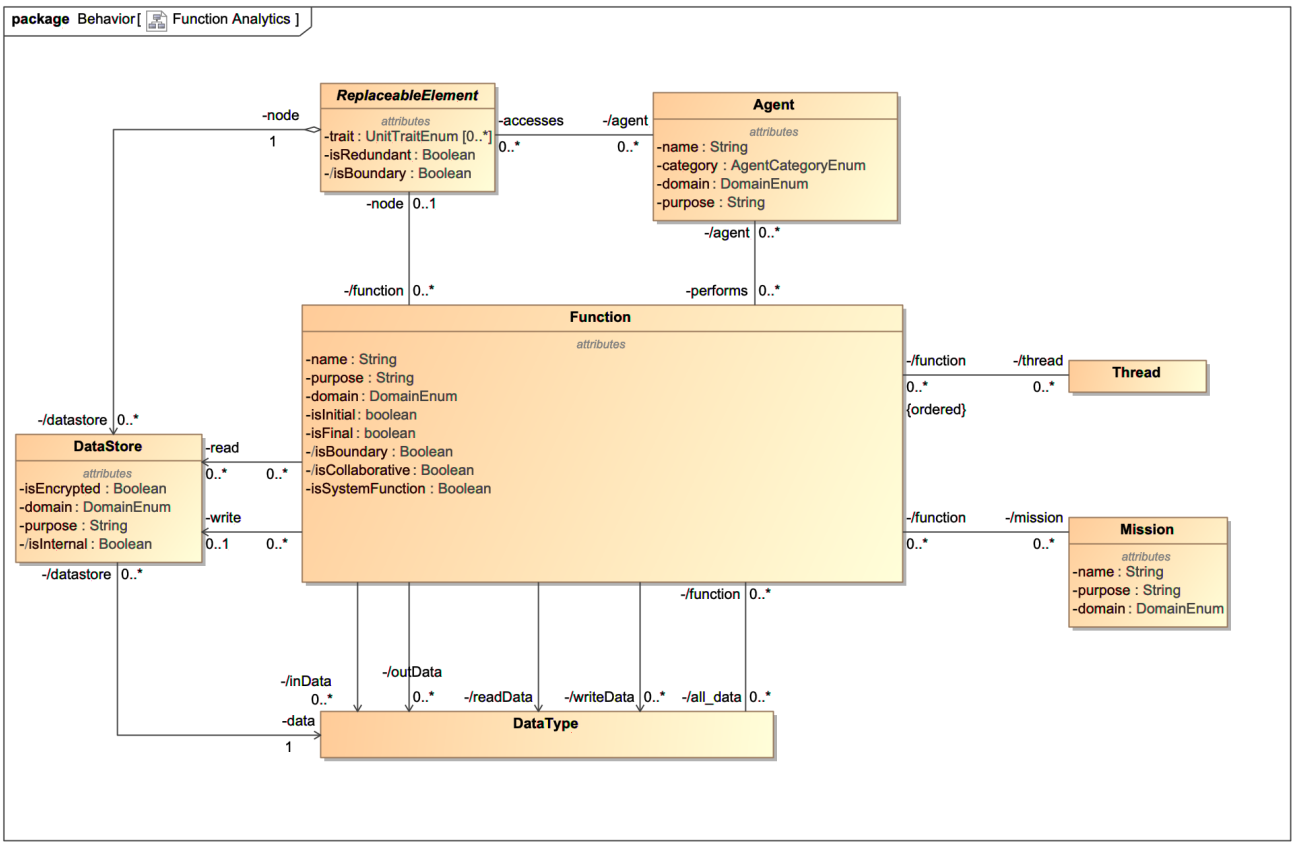

Figure 30 Function Analytics

This page intentionally left blank.

13 SPECTRA Mission package

13.1 Overview

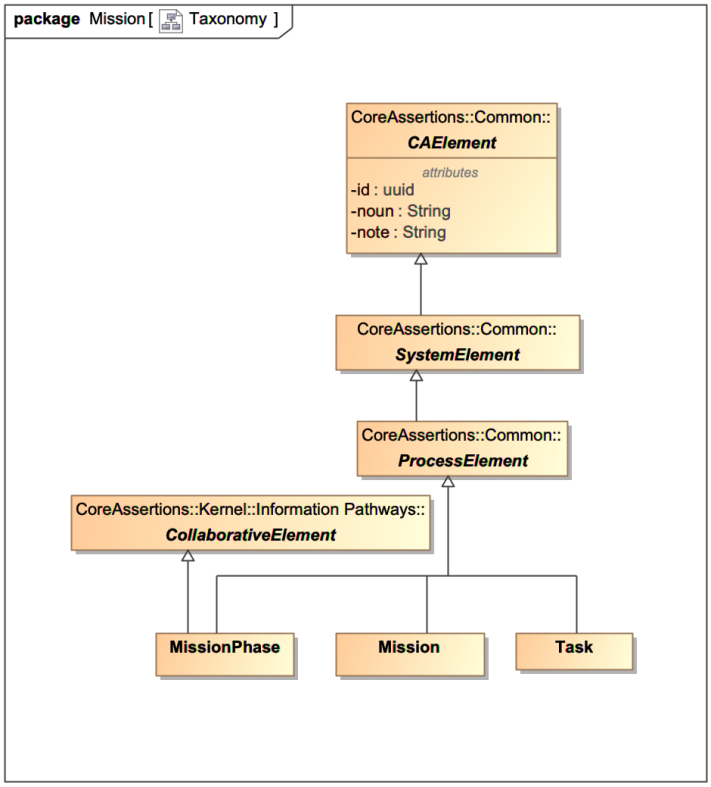

Figure 31 Taxonomy of the Mission package

13.2 Mission

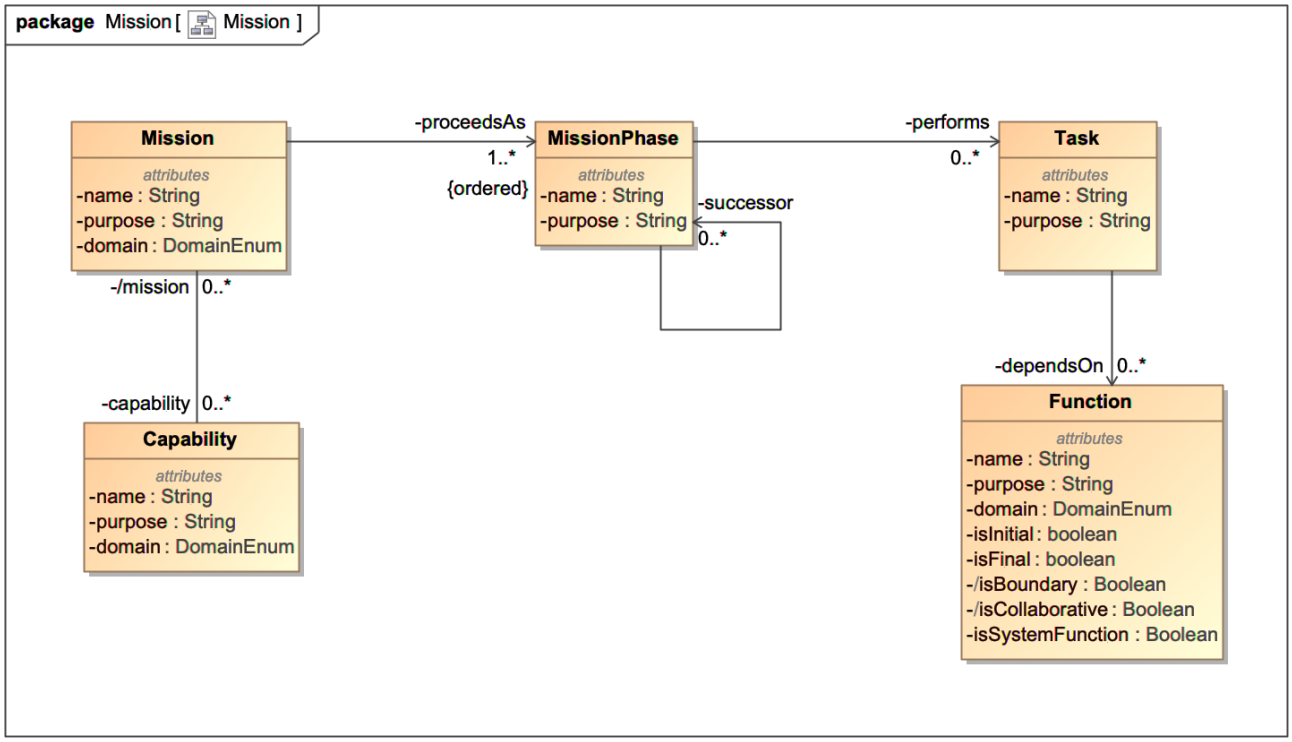

Figure 32 Mission

13.2.1 Mission

Mission - Strategic element that refers to an important task that people must accomplish. In a SysML model, a mission can be represented by an activity, a block, or a use case. Mission is an end-to-end "sequence" of flows usually corresponding to achieving a certain operational objective. Usually a mission involves multiple "phases", possibly involving different configurations of nodes, and with different functions enabled/disabled.

Properties

proceedsAs:Mission Phase[1..*]	An ordered list of one or more phases comprising the mission.
capability:Capability[0..*]	Specifies a unique set of capabilities upon which the mission depends

13.2.2 Mission Phase

Mission phase - Specific state involved in a mission which also involves a specific unit, and exchange configuration and specific functions. A Mission usually involves an ordered collection of "phases" with possibly different configurations of units involved, and different functions enabled/disabled.

MissionPhase element represents a snapshot of the SOI's architecture at a certain point in a Mission. Identification of such configuration is essential for assessing SOI with dynamically changing architecture, which is then represented as a series of static snapshots associated with various mission phases. Each MissionPhase configuration then has a static digital technical surface that can be assessed independently.

Properties

performs:Mission Task[0..*]	Zero or more tasks that the mission performs.
successor:MissionPhase[0..*]	Successor phases of the current phase

13.2.3 Task

Mission Task is an activity or a task, defined in the context of Mission Engineering. Mission Task is involved in a Mission Thread. A Mission Task is different from the Function Units or Emergent Functions of the SOI (at least uses a different vocabulary, one of Mission Engineering), and depends on some of Capabilities of the SOI (and transitively on some Function Units, possibly through some Emergent Functions).

See section Kernel::Impactful Elements::Mission for more detail.

Properties

task:Task[0..*]	Tasks into which the current Task is decomposed
dependsOn:Function[0..*]	Specifies Mission Tasks into which the current Mission Task is decomposed

This page intentionally left blank.

14 SPECTRA Characteristics package

14.1 Overview

Characteristics (together with Artifacts and Traits) provide a shared vocabulary to make assertions about the nature of the elements of SOI. Characteristics describe the common assertions related to the nature of various elements of the SOI.

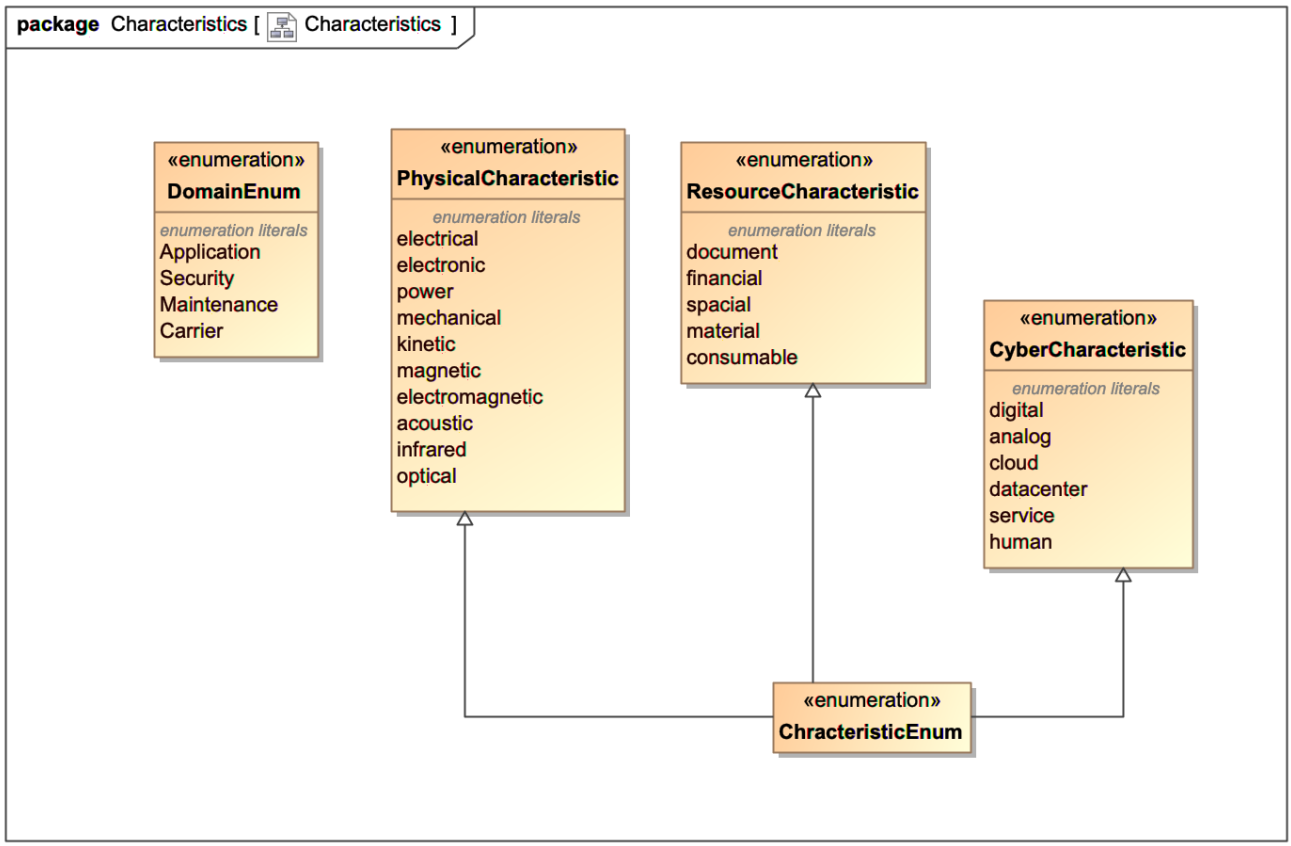

Figure 33 Characteristics

14.2 Characteristics

14.2.1 Domain Enumeration

Domain Characteristic is an abstract element that identifies the domain to which the annotated element belongs to. A domain is a specific subject area in which the SOI is involved, a field of study that defines a set of common requirements, terminology, and functionality for any system or a software program constructed to solve a problem in a given field. For the purposes of risk and cybersecurity assessments of cyber and cyber-physical systems, the following domains are important:

- Application Domain
- Security Domain
- Maintenance Domain
- Carrier Domain (or networking domain)

Table 11 Enumeration Literals for Domain

Enum Value	Definition
Application	This stereotype identifies the element as belonging to the application domain of the SOI (without further specifying what that domain is, e.g. a medical device, an electrical substation, an avionics system, a weapons system). Marking an element as belonging to the application domain distinguishes it from other elements that are marked as belonging to other domains, relevant to SPECTRA (security, maintenance, carrier).
Security	This stereotype identifies the element as belonging to the security domain. Cyber Security is protection of digital information, as well as the integrity of the infrastructure housing and transmitting digital information. More specifically, cyber security includes the body of technologies, processes, practices and response and mitigation measures designed to protect networks, computers, programs and data from attack, damage or unauthorized access to ensure confidentiality, integrity and availability. Security Domain is common to systems in various Application Domains. Marking an element as belonging to the application domain distinguishes it from other elements that are marked as belonging to other domains, relevant to SPECTRA (application, maintenance, carrier).
Maintenance	This stereotype identifies the element as belonging to the Maintenance Domain. Maintenance Domain is about the capabilities that support the evolution of the system (e.g. configuration, upgrades, hot-swap, patching). Maintenance involves functional checks, servicing, repairing or replacing of necessary devices, equipment, software, building infrastructure and supporting utilities in system installations. Maintenance domain is involves certain capabilities of SOI, such as logging, fault management, alerting and software uploads (configuration, patches, hot-swap updates or upgrades). Some data communications within SOI may be related to maintenance, such as delivering an image for a hot-swap without interrupting the regular operations of the system. Maintenance Domain is common to systems in various Application Domains. Marking an element as belonging to the maintenance domain distinguishes it from other elements that are marked as belonging to other domains, relevant to SPECTRA (application, security, carrier).
Carrier	This stereotype identifies the element as belonging to the Carrier (Networking) Domain. Computer networks are a key part of the infrastructure for cyber and cyber-physical systems. Computing devices use common communication protocols over digital interconnections to communicate with each other. These interconnections are made up of telecommunication network technologies based on physically wired, optical, and wireless radio-frequency methods that may be arranged in a variety of network topologies. Carrier Domain is common to systems in various Application Domains. Marking an element as belonging to the carrier domain distinguishes it from other elements that are marked as belonging to other domains, relevant to SPECTRA (application, security, maintenance). Absence of a clear distinction between the application and carrier data is a common anti-pattern in SysML models that is a barrier for effective understanding of the core assertions made about the SOI.

Virtualization	

14.2.2 Characteristic Enumeration

Characteristics (together with Artifacts and Traits) provide a shared vocabulary to make assertions about the nature of the elements of SOI. Characteristics describe the common assertions related to the nature of various elements of the SOI.

Characteristics apply to nodes, channels, flows, information types, resources; The characteristics are common shared meanings that can be used to select appropriate rules and axioms in a knowledgebase and match them to specific elements of the SOI as represented by the model being ingested

Cyber Characteristics

Cyber - element that is involved with digital information processing. The characteristics are common shared meanings that can be used to select appropriate rules and axioms in a knowledgebase and match them to specific elements of the SOI as represented by the model being ingested. For a cyber system the elements are assumed to be cyber, however in a cyber-physical system it is important to distinguish cyber elements from physical elements.

Table 12 Enumeration Literals for Cyber Characteristics

Enum Value	Definition
Digital	Digital - An element involving automated processing of digital information (as opposed to human)
Analog	
Cloud	Cloud - any infrastructure-as-code element that is implemented in a public or private cloud
DataCenter	A data center is a set of computer systems, telecommunications and storage systems, often housed in a dedicated space such as a building or a group of buildings on the same site. While early data centers separated computing hardware from the remote terminal equipment, modern data centers usually host cloud-based virtualized infrastructure.
Human	Human - any element involving humans, subject to social engineering attacks

Physical Characteristics

Physical - "physical concerns for cyber-physical systems. "Physical" characteristics can be applied to a replaceable element, conveying element, or resource. Applying a physical characteristic to a conveying element refers to its physical communications medium and is a shortcut for describing the proper carrier interface. For the purposes of cybersecurity risk assessment, SPECTRA distinguishes several frequently used subclasses, usually the choice is straightforward. The characteristics are common shared meanings that can be used to select appropriate rules and axioms in a knowledgebase.

Table 13 Enumeration Literals for Physical Characteristics

Enum Value	Definition
Electrical	Electrical - channel or node or flow or protocol or resource involving electrical current, usually involving some kind of conductors, e.g. house wiring for light fixtures, power substation, electronics, e.g. alarm triggers; flow or protocol involving electricity; DC or AC current.
Electronic	
Power	
Mechanical	Mechanical - channel or node or flow or protocol or resource involving physical forces or motion, e.g. brakes, wheel axle, antenna gimbal; not to be confused with computer hardware
Kinetic	While Kinetics is the branch of classical mechanics that is concerned with the relationship between the motion and its causes, specifically, forces and torques, the term Kinetic warfare is sometimes used as a term for military combat or other forms of directly-destructive warfare. In SPECTRA, kinetic characteristic is defined as producing a mechanical impact. It is somewhat overlapping with another SPECTRA characteristic “mechanical”.
Magnetic	
Electromagnetic	Electromagnetic - channel or node or flow or protocol or resource involving electro-magnetic spectrum, e.g HF, VHF, etc.
Acoustic	Acoustic - channel or node or flow or protocol or resource involving sound, acoustic element (using sound-waves)
Infrared	Infrared - channel or node or flow or protocol or resource involving infrared spectrum, technically, this is also EMF, but this is very specific and also short range
Optical	Optical - channel or node or flow or protocol or resource involving optical element (usually involving lasers); over fiber optic channels or any other appropriate media
Spacial	

Resource Characteristics

Table 14 Enumeration Literals for Resource Characteristics

Enum Value	Definition
Document	Document - a physical document on some media, e.g. paper, optical disc, etc. This characteristic applies to resource elements.
Consumable	Consumable - a consumable physical resource (tangible or intangible), e.g. fuel, time-left-unit-maintenance. This characteristic applies to resource elements.
Financial	Financial - this characteristic applies to financial resources or information types.
Material	Material - a characteristic that can be applied to resources to describe a piece of equipment and supplied in a supply-chain management context.

15 SPECTRA Traits package

12.1 Overview

Traits (together with Artifacts and Characteristics) provide a shared vocabulary to make assertions about the nature of the elements of SOI. Traits describe the common assertions related to the nature of the replaceable elements of the SOI and their role in the missions and capabilities supported by the SOI. The vocabulary of Traits is coordinated with the vocabulary of Operational Data (hence this package is named “Traits and Data”).

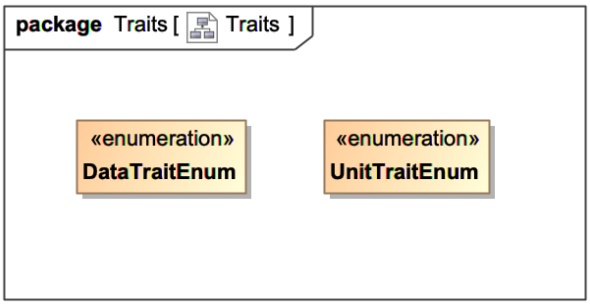

Figure 34 Traits

15.2 Traits

15.2.1 UnitTraitEnum (Enumeration)

Trait describes the nature of a replaceable unit. Specific subclasses of Trait provide a vocabulary of shared meanings to describe the nature of the replaceable elements of cyber and cyber-physical systems. The vocabulary of Application Traits provides only a useful top-level vocabulary across multiple application domains (e.g. avionics, electrical substations, electrical vehicles, medical devices, etc. can have their own extensive domain ontologies, not addressed by SPECTRA). On the other hand, the vocabulary for Security Traits and Maintenance Traits is more detailed, as these domains are common across various cyber and cyber-physical systems.

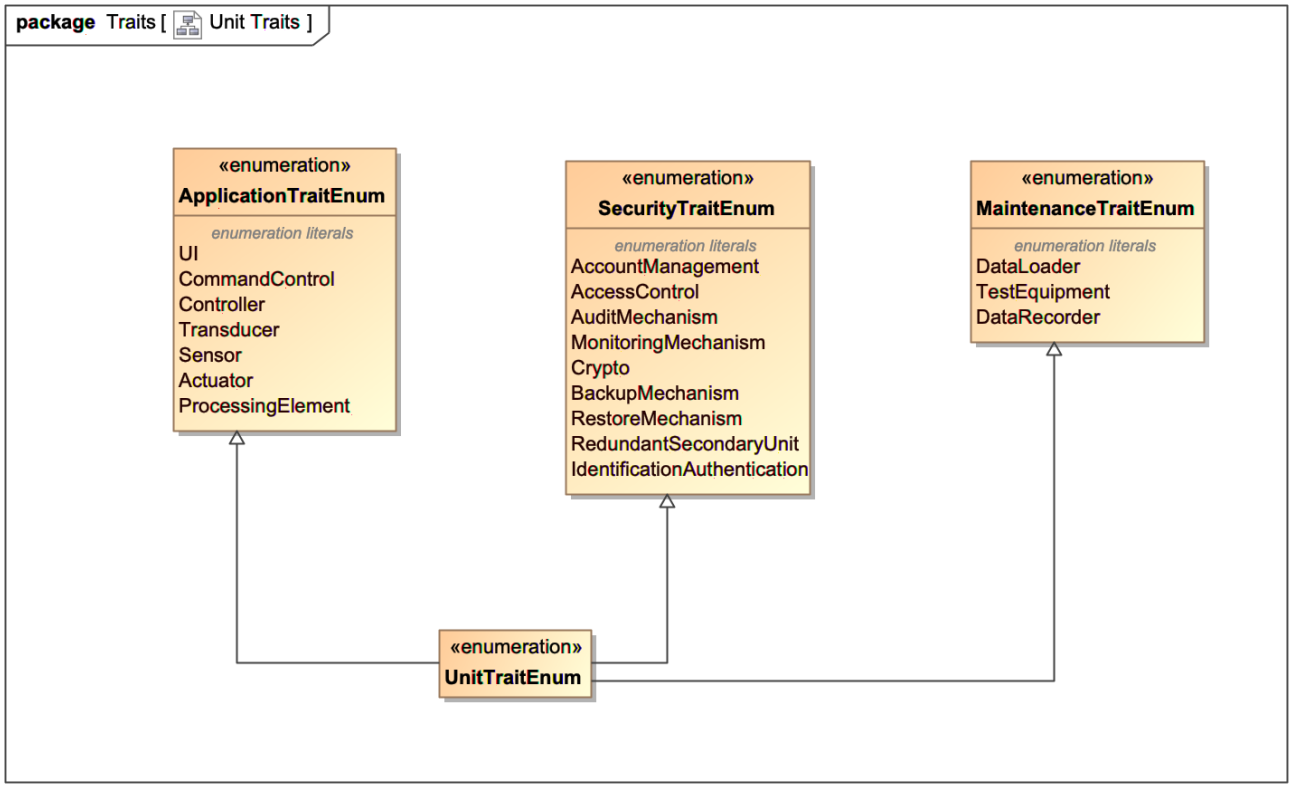

Figure 35 Unit Traits

15.2.2 Unit Trait Enumeration

Traits (together with Artifacts and Characteristics) provide a shared vocabulary to make assertions about the nature of the elements of SOI. Traits describe the common assertions related to the nature of the replaceable elements of the SOI and their role in the missions and capabilities supported by the SOI. The vocabulary of Traits is coordinated with the vocabulary of Operational Data (hence this package is named “Traits and Data”).

Trait describes the nature of a replaceable unit. Specific subclasses of Trait provide a vocabulary of shared meanings to describe the nature of the replaceable elements of cyber and cyber-physical systems. The vocabulary of Application Traits provides only a useful top-level vocabulary across multiple application domains (e.g. avionics, electrical substations, electrical vehicles, medical devices, etc. can have their own extensive domain ontologies, not addressed by SPECTRA). On the other hand, the vocabulary for Security Traits and Maintenance Traits is more detailed, as these domains are common across various cyber and cyber-physical systems.

Application Traits

Table 15 Enumeration Literals for Application Traits

Enum Value	Definition
UI	User Interface (UI) - user interface component that is used by some human operator. Usually implemented as part of some custom software.
CommandControl	Command control - decision making capability, usually implemented as part of some custom application software.
Controller	Controller (a control device) - is a component of a cyber-physical system that provides control signals to actuators or other physical

	devices. Usually, a controller is receiving digital information from other parts of the system.
Sensor	A sensor train represents the role of an element of a cyber-physical system that detects events or changes in its physical environment and sends the digital information to other elements, usually a processing element of some kind.
Actuator	An actuator is a component of a cyber-physical system that produces force, torque, or displacement, when a digital, electrical, pneumatic or hydraulic input is supplied to it. The effect is usually produced in a controlled way. An actuator translates such an input signal into the required form of mechanical energy. It is a special type of transducer. Some systems make distinction between a control device and an actuator, where the control signal is physical rather than digital. Modern systems often involve digitally controlled actuators. An actuator requires a control device (which provides control signal) and a source of energy. The control signal is usually relatively low in energy and may be voltage, electric current, pneumatic, or hydraulic fluid pressure. In the electric, hydraulic, and pneumatic sense, it is a form of automation or automatic control. SPECTRA traits can cover various situations. Several traits can be added to the same replaceable unit or subcomponents, if needed. See other Application Traits, Command Control, Controller, Sensor, Transducer.
Transducer	A transducer is a device that converts energy from one form to another. Usually, a transducer converts a signal in one form of energy to a signal in another. Transducers are often employed at the boundaries of automation, measurement, and control systems, where electrical signals are converted to and from other physical quantities (energy, force, torque, light, motion, position, etc.). An example of a transducer is an antenna, which can convert radio waves (electromagnetic waves) into an electrical signal to be processed by a radio receiver or translate an electrical signal from a transmitter into radio waves. Another example is a voice coil, which is used in loudspeakers to translate an electrical audio signal into sound, and in dynamic microphones to translate sound waves into an audio signal. Sensors and Actuators can be considered as specialized transducers.
ProcessingElement	An element that processes (and possibly stored) digital information. A cyber-physical system can be described as a combination of sensors, actuators and processing elements. Some processing elements can be identified as Command Control elements or Controllers.

Security Traits

Security Traits entail Security Domain. The elements in this package represent the common security mechanisms for cyber and cyber-physical systems and are aligned with NIST-800-53a “Assessing Security and Privacy Controls in Information Systems and Organizations”.

Table 16 Enumeration Literals for Security Traits

Enum Value	Definition
Account Management	A user is an Individual, or (system) process acting on behalf of an individual, authorized to access a system. A user account is the information about the user, often involving the identifier (such as a user name), the authenticator (e.g. a password), the access permissions, etc. Account management mechanism supports managing user accounts. This can be a service, a library, etc. Usually such a mechanism has a specific location within the information pathways of the SOI, in one of the replaceable units. Understanding the location of account management is important for the purposes of cybersecurity risk assessments.
AccessControl	In physical security and information security, access control (AC) is the mechanism that provides selective restriction of access to a place, function, capability or resource. Usually such a mechanism has a specific location within the information pathways of the SOI, in one or more of the replaceable units. Understanding the location of access control mechanisms is important for the purposes of cybersecurity risk assessments.
AuditMechanism	In the context of cyber and cyber physical systems, the audit (AU) mechanism is responsible for generating and managing audit records. Auditing is a formal, systematic and disciplined approach designed to evaluate and improve the effectiveness of processes and related controls. Audit often supports the non-repudiation objective of cybersecurity. Usually, auditing is governed by professional standards, completed by individuals independent of the process being audited, and normally performed by individuals with one of several acknowledged certifications. Usually, a mechanism that generates records for audit has a specific location within the information pathways of the SOI, in one or more of the replaceable units. Understanding the location of the audit mechanisms is important for the purposes of cybersecurity risk assessments.
MonitoringMechanism	In the context of cyber and cyber physical systems, the monitoring (MON) mechanism is a capability to monitor access and other security-relevant events, network traffic, etc. and to generate and manage monitoring records. Monitoring mechanism is different from audit. Monitoring is an on-going process usually directed by management to ensure processes are working as intended. Monitoring is an effective control within a process. Management uses monitoring tools and processes to verify that controls it has implemented are working on a routine basis and that business risks are being identified and addressed. However, because management is checking on their own operations, an inherent conflict is evident in that reporting may reflect what management prefers to report instead of what the actual results portray. Also, in many respects, operational personnel have a better understanding of the data and therefore may create the most appropriate and effective monitoring tools. However, those tools are not necessarily tempered by objectivity or the perspectives/ knowledge of an experienced auditor. Usually, a monitoring mechanism has a specific location within the information pathways of the SOI, in one or more of the replaceable units. Understanding the location of the monitoring mechanisms is important for the purposes of cybersecurity risk assessments.

Crypto	Crypto – A cryptographic asset including algorithms, protocols, certificates, keys, tokens, and secrets. Aligned with Cyclone DX component type "cryptographic asset"
BackupMechanism	Backup, or data backup is a copy of computer data taken and stored elsewhere so that it may be used to restore the original after a data loss event. The backup mechanism is a capability to conduct system backups (whether automatically or with the assistance of humans). Backups can be used to recover data after its loss from data deletion or corruption, or to recover data from an earlier time. Backups provide a simple form of IT disaster recovery; Backups are involved in the process of reconstituting a computer system or other complex configuration such as a computer cluster, active directory server, or database server. See also the Restore mechanism.
RestoreMechanism	Restore is the term used for creating original data from a backup. Backups are involved in the process of reconstituting a computer system or other complex configuration such as a computer cluster, active directory server, or database server. See also Backup mechanism.
RedundantSecondaryUnit	Redundancy refers to the duplication of critical components or functions of a system with the intention of increasing reliability. Identifying a certain unit as a Redundant Secondary Unit is important for correct interpretations of the model of SOI.
IdentificationAuthentication	Identification and authentication (IA) of users is a mechanism involved in verifying identity of a user of SOI. Authentication (FIPS 200, NIST-800-53) is a process of verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in a system. The process of authentication may be used to validate personal identifier, verifying the authenticity of a website or a protocol endpoint using a digital certificate, ensuring that a product or a document is not counterfeit, etc. IA might include an explicit examination of any key-based, “password-less” login capability and potential risks inherent from any deficiency in key management, account management, system and boundary protection, physical and environmental protection, and other safeguards to prevent identification and authentication bypass by unauthorized users or processes acting on behalf of users. A multi-factor authentication mechanism (NIST-SP-800-63-3, NIST-800-53) is an authentication system or an authenticator that requires more than one authentication factor for successful authentication. Multi-factor authentication can be performed using a single authenticator that provides more than one factor or by a combination of authenticators that provide different factors. The three authentication factors are something you know, something you have, and something you are.

Maintenance Traits

Table 17 Enumeration Literals for Maintenance Traits

Enum Value	Definition
DataLoader	Data loader is a utility program that receives data that represents an artifact image (or a patch) and updates the corresponding artifact in the SOI. For example, patches for proprietary software are typically distributed as executable files instead of source code. When executed these files load a program into memory which manages the installation of the patch code into the target program(s) on disk.

	Patches for other software are typically distributed as data files containing the patch code. These are read by a patch utility program which performs the installation. This utility modifies the target program's executable file—the program's machine code—typically by overwriting its bytes with bytes representing the new patch code.
TestEquipment	Test and maintenance equipment is connected to the SOI (at least at some Mission Phases), therefore it is important to identify such elements in the model of the SOI.
DataRecorder	The Data Recorder element (sometimes referred to as a system monitor) is very similar to Monitoring Mechanism in the Security Domain. A system monitor is a hardware or software component used to monitor system resources and performance in a computer system and generate Maintenance records for the benefits of the maintenance team and their processes.

15.2.3 DataTraitEnum (Enumeration)

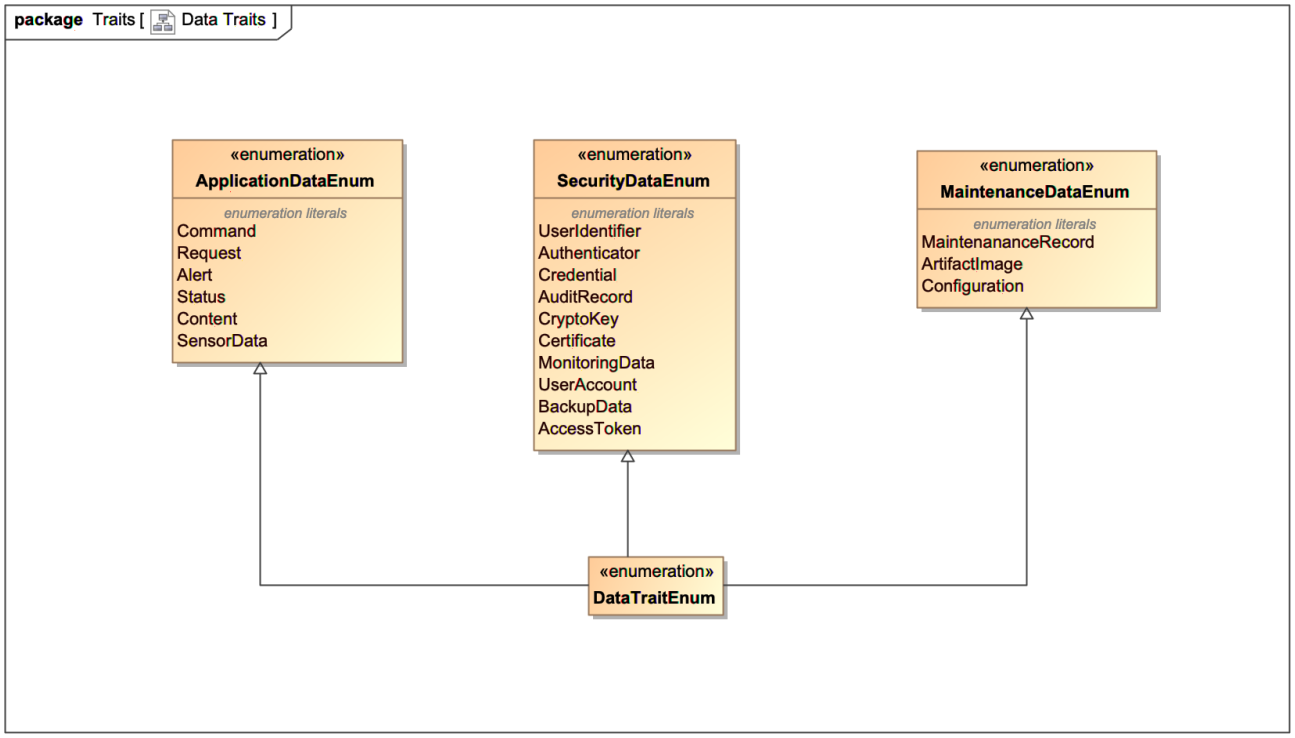

Figure 36 Data Traits

15.2.4 Datatype Trait Enumeration

Traits (together with Artifacts and Characteristics) provide a shared vocabulary to make assertions about the nature of the elements of SOI. Traits describe the common assertions related to the nature of the replaceable elements of the SOI and their role in the missions and capabilities supported by the SOI. The vocabulary of Traits is coordinated with the vocabulary of Operational Data (hence this package is named "Traits and Data").

Application Data

Table 18 Enumeration Literals for Application Data Traits

Enum Value	Definition
Command	Command – digital information that is interpreted by the recipient as a command. Often an element of a cyber-physical system, identified as a Command Control, or a Controller sends Command data to a Controller or an Actuator. The application data going in the opposite direction is often identified as Status.
Status	Status – digital information that is interpreted by the receiver as a status report. Often an element of a cyber-physical system, identified as a Command Control, or a Controller sends Command data to a Controller or an Actuator and would expect a Status.
Request	Request – digital information that is interpreted by the recipient as a request/query. Often an element of a cyber or a cyber-physical system, identified as a Processing Element or Command Control, or a Controller sends Request data to another Processing Element, and would expect a Content (and possibly a Status) back.
Content	Content – digital information that is interpreted by the receiver as an application-specific content (e.g. in response to a request). Often an element of a cyber or a cyber-physical system, identified as a Processing Element or Command Control, or a Controller sends Request data to another Processing Element, and would expect a Content (and possibly a Status) back.
Alert	Alert – digital information that is interpreted by the receiver as a notification of an event.
SensorData	Sensor Data – digital information that is interpreted by the recipient as a digital description of the physical environment of the SOI, often produced directly by some Sensor. Often an element of a cyber-physical system, identified as a Sensor, sends Sensor data to some Processing Element, Controller or Command Control, possibly as the result of a Request.

Security Data

Table 19 Enumeration Literals for Security Data Traits

Enum Value	Definition
UserIdentifier	Identifier (FIPS 201-2, NIST-800-53) is a unique data used to represent a person's identity and associated attributes. A name or a card number are examples of identifiers. A unique label used by a system to indicate a specific entity, object, or group. Identifier is a form of digital identity. A digital identity is data stored on computer systems relating to an individual, organization, application, or device. For individuals, it involves the collection of personal data that is essential for facilitating automated access to digital services, confirming one's identity on the internet, and allowing digital systems to manage interactions between different parties.

Authenticator	Authenticator (NIST-800-53) is something that the claimant possesses and controls (typically a cryptographic module or password) that is used to authenticate the claimant's identity.
Credential	Credential (NIST SP 800-634-3, NIST-800-53) is an object or data structure that authoritatively binds an identity, via an identifier or identifiers, and (optionally) additional attributes, to at least one authenticator possessed and controlled by a subscriber.
AuditRecord	Audit record is security related information generated from certain selected event types related to the SOI. Audit records are produced by the Audit mechanism. Audit records are often compiled into a system-wide audit trail that is time-correlated within organizational tolerances. Protection of the audit records may involve various cryptographic mechanisms.
CryptoKey	A key in cryptography is a piece of information, usually a string of numbers or letters that are stored in a file, which, when processed through a cryptographic algorithm, can encode or decode cryptographic data. Based on the used method, the key can be different sizes and varieties, but in all cases, the strength of the encryption relies on the security of the key being maintained. The security of a key is dependent on how a key is exchanged between parties. Establishing a secured communication channel is necessary so that outsiders cannot obtain the key. A key establishment scheme (or key exchange) is used to transfer an encryption key among entities. Key agreement and key transport are the two types of a key exchange scheme that are used to be remotely exchanged between entities.
Certificate	In cryptography, a public key certificate, also known as a digital certificate or identity certificate, is an electronic document used to prove the validity of a public key. The certificate includes the public key and information about it, information about the identity of its owner (called the subject), and the digital signature of an entity that has verified the certificate's contents (called the issuer). If the device examining the certificate trusts the issuer and finds the signature to be a valid signature of that issuer, then it can use the included public key to communicate securely with the certificate's subject.
MonitoringData	Data generated by a monitoring mechanism.
UserAccount	A user is an Individual, or (system) process acting on behalf of an individual, authorized to access a system. A user account is the information about the user, often involving the identifier (such as a user name), the authenticator (e.g. a password), the access permissions, etc.
Backup	Data that is generated by a backup mechanism and that is intended for a subsequent restore of the system.
AccessToken	In computer systems, an access token contains the security credentials for a login session and identifies the user, the user's groups, the user's privileges, and, in some cases, a particular application. An access token is an object encapsulating the security identity of a process or thread. A token is used to make security decisions and to store tamper-proof information about some system entity. An access token is usually generated by the logon service when a user logs on to the system and the credentials provided by the user are authenticated against the authentication database. The authentication database contains credential information required to construct the initial token for the logon session, including its user id, primary group id, all other groups it is part of, and other information.

Maintenance Data

Table 20 Enumeration Literals for Maintenance Data Traits

Enum Value	Definition
MaintenanceRecord	The Maintenance Record element is similar to the Monitoring Data element in the Security domain. Monitoring Data is usually generated by a Data Recorder component (as the Maintenance trait).
ArtifactImage	Artifact image is data that is intended to be used to modify an existing software resource such as a program or a file, often to fix bugs and security vulnerabilities (referred to as a patch). An artifact image may represent a hotfix (or a hot-swap image), a patch, a major or a minor release of the software or firmware.
Configuration	Configuration (usually a configuration file) is data used to configure the parameters and initial settings for some computer programs or applications, server processes and operating system settings. Some computer programs only read their configuration files at startup. Others periodically check the configuration files for changes. Managing, distributing and updating configurations is part of the maintenance domain for SOI.

This page intentionally left blank.

16 SPECTRA Mitigation package

16.1 Overview

This package is optional. It provides a mechanism to annotate certain elements in the model as representations of security controls with references to a certain common security control catalog. Examples of a Security Control Catalog are NIST SP-800-53 and ITSG-33. The elements in this package entail Security Domain. Some security controls overlap with Security Traits (as Security Traits represent the security mechanisms, aligned with NIST-800-53). Allocated Control Elements can be further stereotyped with Security Traits.

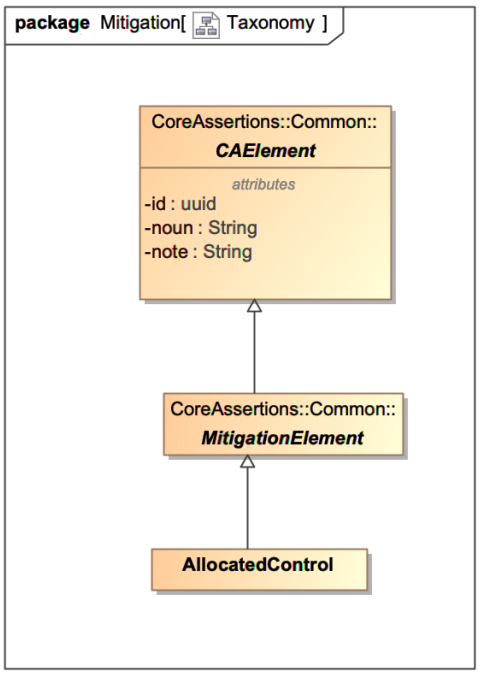

Figure 37 Taxonomy of the Mitigation package

16.2 Mitigation

Mitigation elements entail Security Domain.

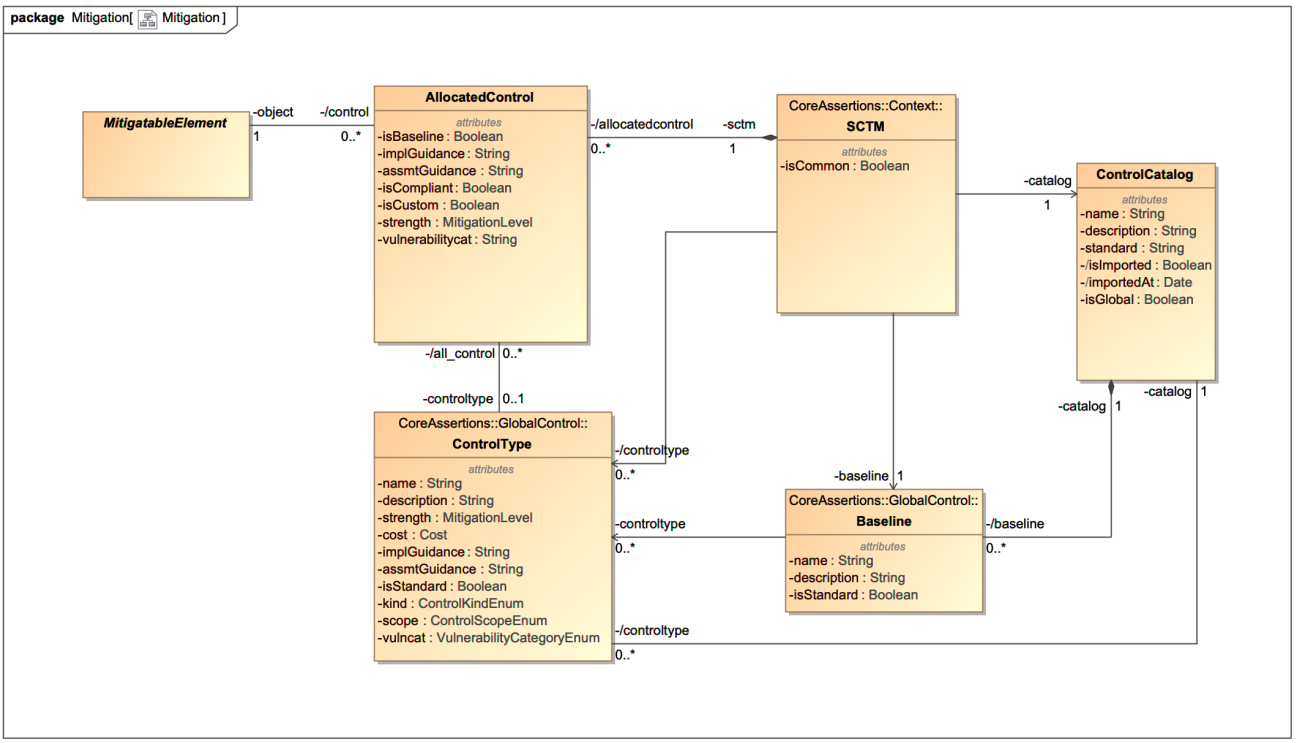

Figure 38 SCTM

16.2.1 Allocated Control

Allocated Control - representation of a statement, that a control type x from a certain control catalog, has been allocated to a certain node, or channel. By allocating controls to attackable units (as well as related channels, protocols and configurations) we can consider various related functional threads and understand segments of threads that are mitigated; by considering threads we can see how our protection covers other attackable targets; we can also understand control by-passes.

Some security controls overlap with Security Traits (as Security Traits represent the security mechanisms, aligned with NIST-800-53). Allocated Control Elements can be further stereotyped with Security Traits.

Properties

control name:String	Specifies the name of the control (as defined in the control catalog, identified in the Assessment Context element)
option key:String	Specifies the key of the mitigation option to which the allocated control belongs
allocated to:Mitigatable Element[1]	Specifies the Mitigatable Element to which the control is allocated (the scope of the control)

Semantics

Each Allocated Control element shall be associated with a Mitigation Option by using a matching option key string. The control name of the allocated control must match exactly one control type name for the corresponding Control Catalog. The Control Catalog is identified through the association between the Mitigation Option and its Assessment Context element, as the property control catalog of the later. SPECTRA does not enforce the identification of the control catalog by the control catalog string. Managing the identification of the control name in the corresponding control catalog shall be the responsibility of the compliant tools, and part of the SPECTRA validation capability.

Each Allocated Control element shall identify a specific Mitigatable Element as its scope, thus defining the exact place of the control in the Information Pathways of the SOI.

16.2.2 Mitigation Option

Mitigation Option - also known as Security Control Traceability Matrix (SCTM) – a collection of allocated controls for the replaceable units and channels of the system under assessment, especially if more than one collection is under consideration, e.g. several desired mitigation plans, several milestones, etc. This mechanism allows managing several collections of security controls, for example, as alternatives that are subject to the evaluation during the assessment, or as "before" and "after" snapshots, or as several milestones, etc. Each Mitigation Option is associated with an Assessment Context. A Mitigation Option may describe the baseline to be used by the assessment, which the Assessment Context defines the Control Catalog as the context in which control names and baselines are interpreted.

Properties

option key:String	Specifies a unique key of the mitigation option (shared by the allocated controls that belong to this mitigation option).
baseline name:String	(optional) Specifies the name of the baseline (as defined in the control catalog, identified in the Assessment Context element)
context name:String	Specifies the name of the Assessment Context element to which this Mitigation Option belongs

Semantics

Each Mitigation Option shall be associated with an Assessment Context element by using the matching context name. If Allocated Control elements are used, the model shall have at least one block with the stereotype Mitigation Option.

16.2.3 Mitigatable Element (abstract)

Mitigatable Element is an abstract class that is defined as a union of SPECTRA elements to which allocated controls may be applied. Using a certain SPECTRA element as a Mitigatable Element to identify the placement of some security controls and their scope does not entail Security Domain for the Mitigatable Element.

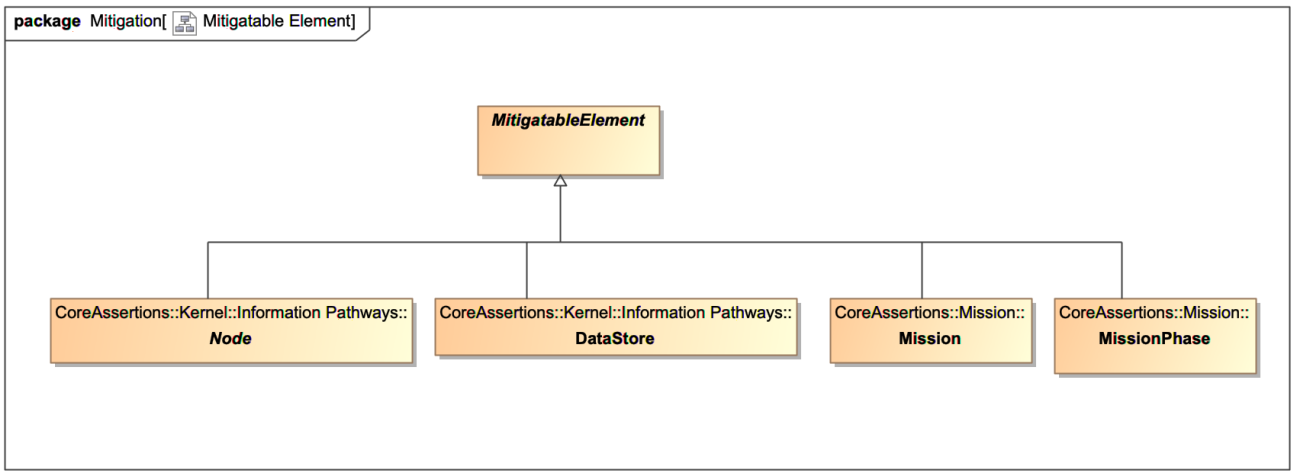

Figure 39 Mitigatable Element

This page intentionally left blank.

17 SPECTRA GlobalControl package

17.1 Overview

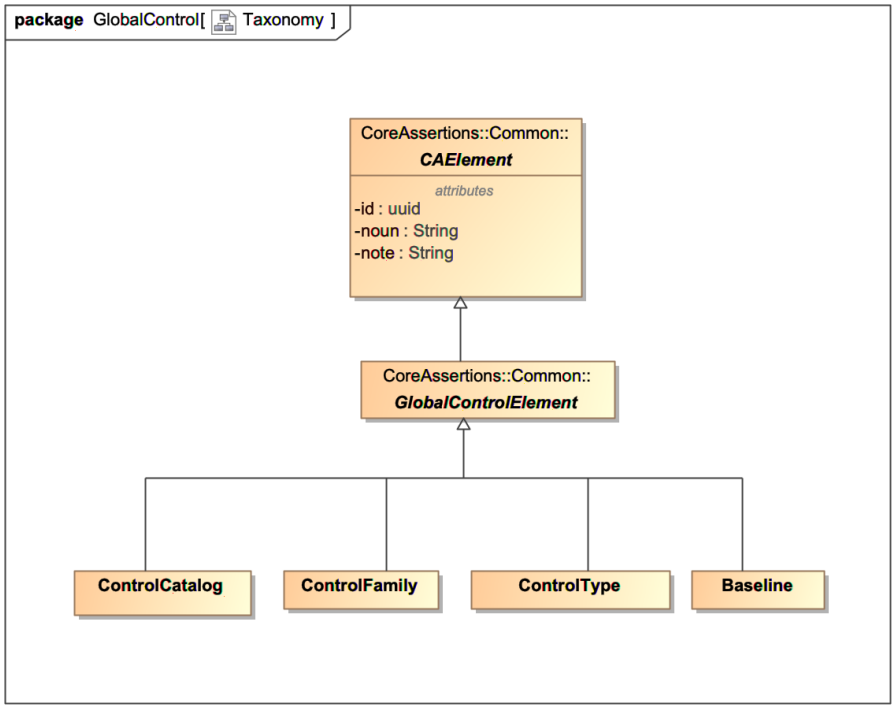

Figure 40 Taxonomy of the Global Control package

17.2 Control Catalog

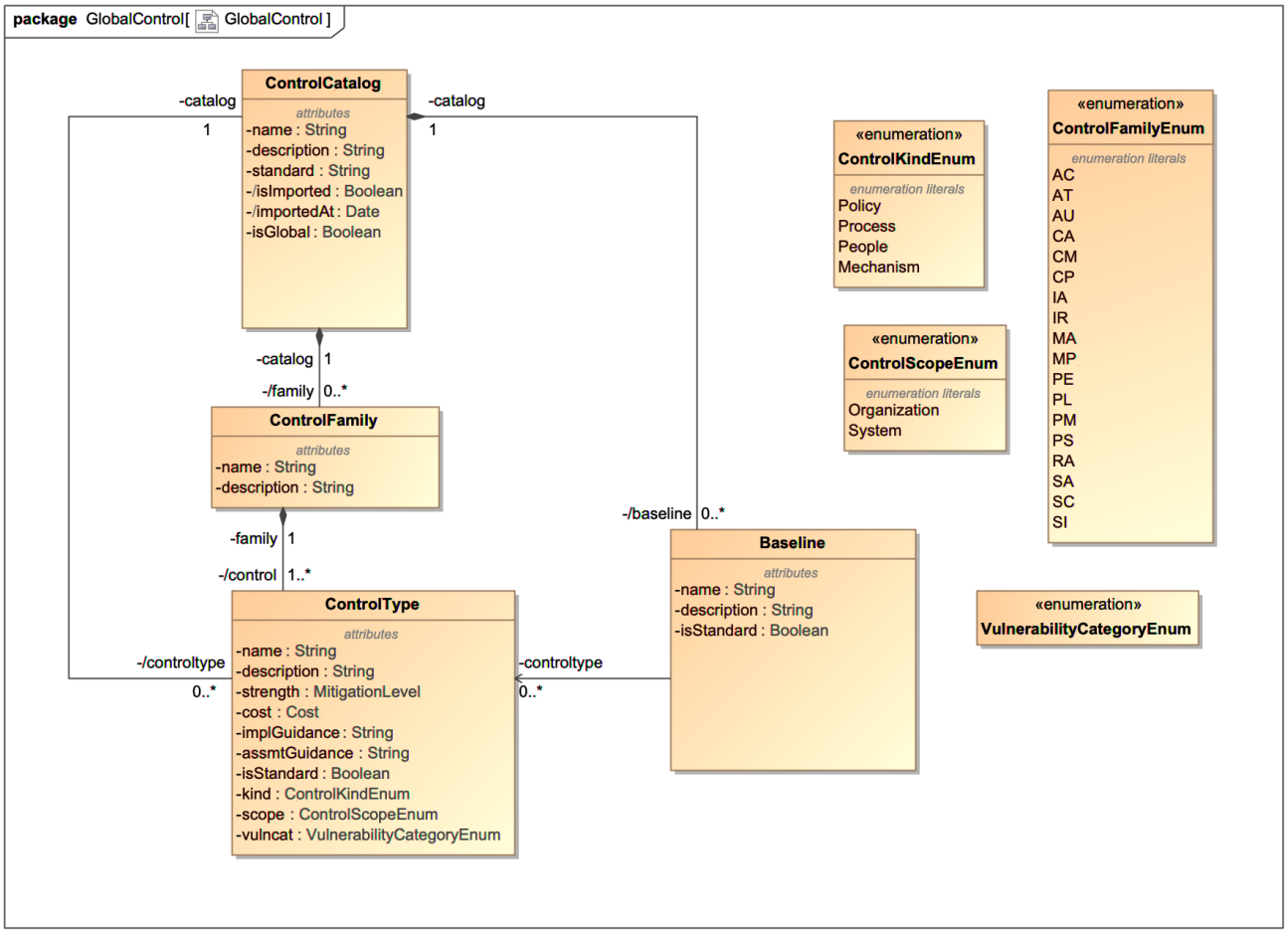

Figure 41 Control Catalog

17.2.1 ControlCatalog

Properties

control name:String	Specifies the name of the control (as defined in the control catalog, identified in the Assessment Context element)
option key:String	Specifies the key of the mitigation option to which the allocated control belongs
allocated to:Mitigatable Element[1]	Specifies the Mitigatable Element to which the control is allocated (the scope of the control)

17.2.2 ControlFamily

Properties

control name:String	Specifies the name of the control (as defined in the control catalog, identified in the Assessment Context element)
option key:String	Specifies the key of the mitigation option to which the allocated control belongs
allocated to:Mitigatable Element[1]	Specifies the Mitigatable Element to which the control is allocated (the scope of the control)

17.2.3 ControlType

Properties

control name:String	Specifies the name of the control (as defined in the control catalog, identified in the Assessment Context element)
option key:String	Specifies the key of the mitigation option to which the allocated control belongs
allocated to:Mitigatable Element[1]	Specifies the Mitigatable Element to which the control is allocated (the scope of the control)

17.2.4 Baseline

Properties

control name:String	Specifies the name of the control (as defined in the control catalog, identified in the Assessment Context element)
option key:String	Specifies the key of the mitigation option to which the allocated control belongs
allocated to:Mitigatable Element[1]	Specifies the Mitigatable Element to which the control is allocated (the scope of the control)

This page intentionally left blank.

18 SPECTRA Context package

18.1 Overview

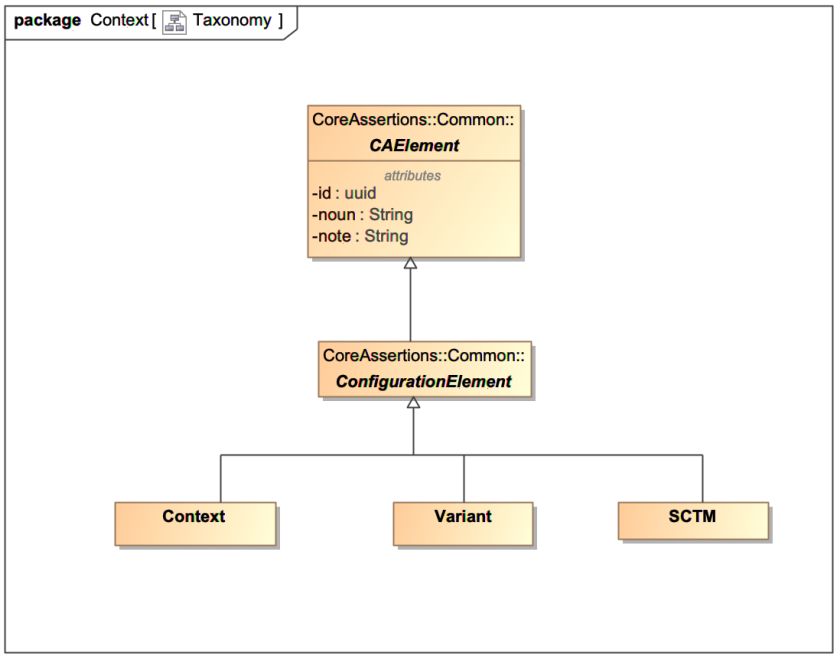

Figure 42 Taxonomy of the Context package

18.2 Context

18.2.1 Context

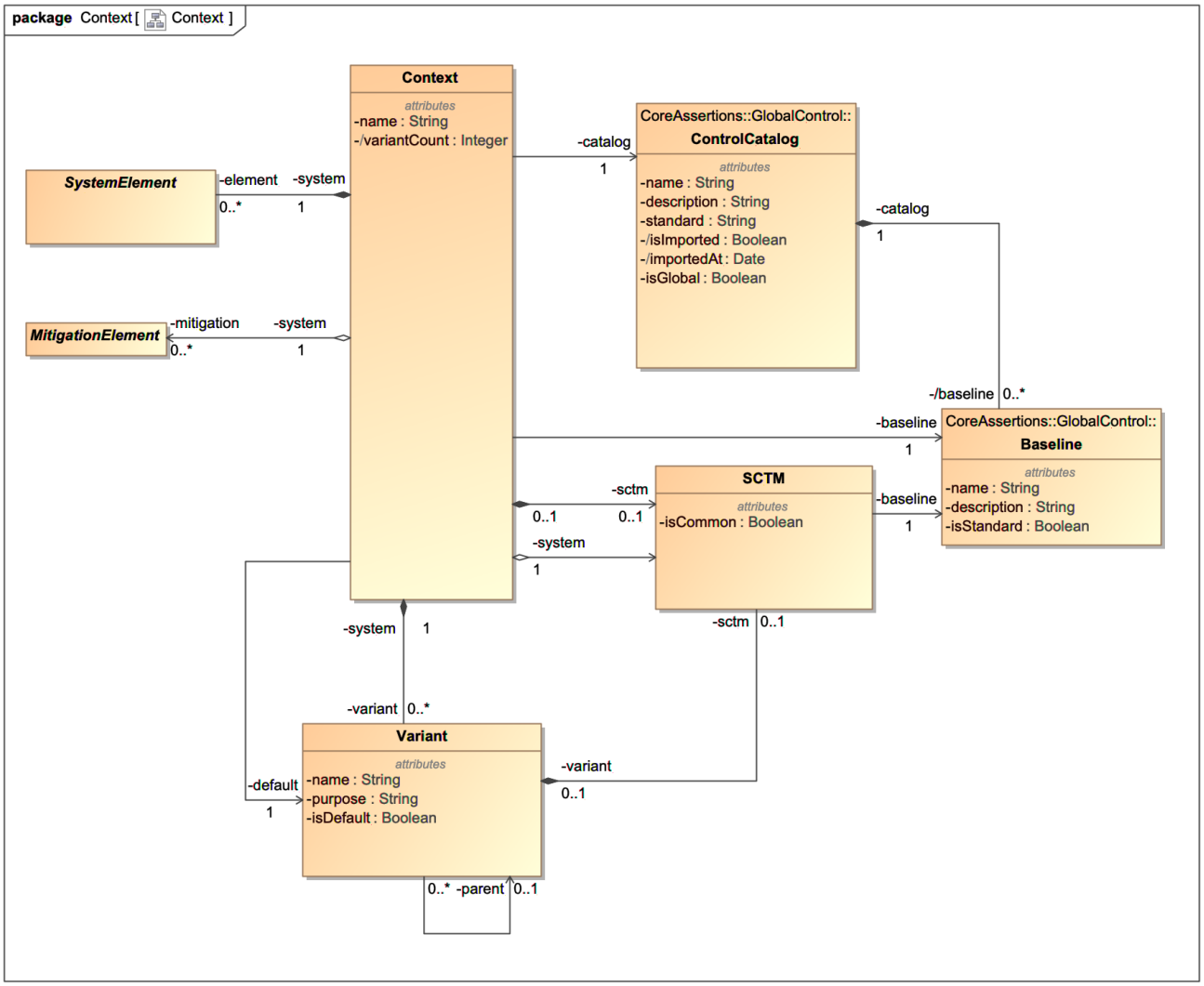

Figure 43 Context

18.2.2 Variant

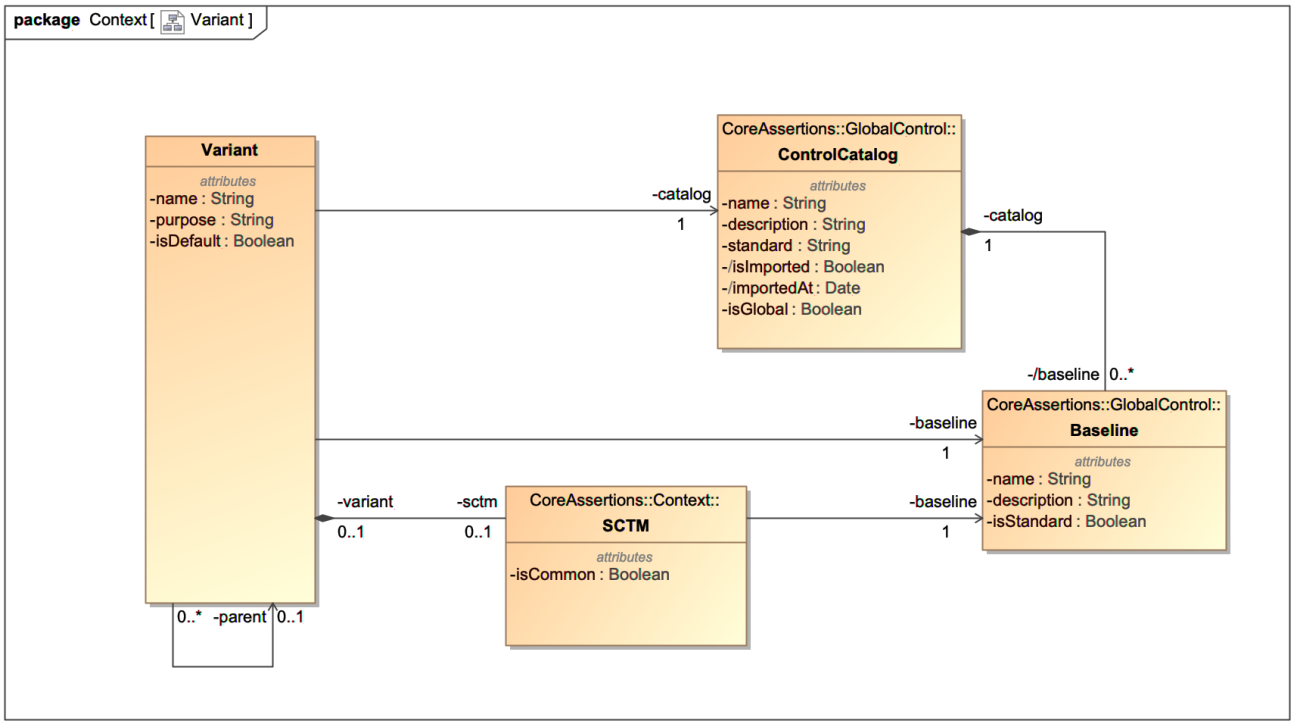

Figure 44 Variant

18.2.3 SCTM

Security Control Traceability Matrix (SCTM) is a container for a set of controls considered together.

This page intentionally left blank.

Annex A: JSON Schema for SPECTRA Core Assertions

(normative)

This page intentionally left blank.

Annex B: SPECTRA Core Assertions

(informative)

B.1 Overview

This section provides an overview of the SPECTRA Core Assertions. Core Assertions are defined by the SPECTRA Core Assertions Metamodel. This Appendix outlines the Core Assertions, as the foundation for defining the semantics of SPECTRA metadata for SysML v2. This is not a full specification of the Core Assertions Metamodel.
Prolog-like clauses were selected as the mechanism for describing the Core Assertions because of the fact that Prolog combines assertions to a database of facts, and logical inferences, and because of the abundance of practical tools. The definitions based on Prolog can be easily transferred to a multitude of other languages (relational database schemas, json, etc.) and other formalisms. Also the choice of Prolog for describing Core Assertions emphasized the technical and pragmatic nature of these definitions, leaving a full SPECTRA Ontology as a separate specification in the SPECTRA family.

B.2 Semantic Foundation

Core Assertions define some entities and relationships. The foundation for the Core Assertions has a simple triple organization.

Meta-term	Description
entity(id,noun)	meta-term representing an assertion that a certain entity exists, and that it can be uniquely identified by an identifier *id*, and that it belongs to some class, identified by *noun*.
attribute(id, name, value)	meta-term representing an assertion that a certain entity identified by an identifier *id*, has an attribute with name *name*, and with value *value*.
relationship(verbphrase,fromid,toid)	meta-term representing an assertion that a certain entity identified by an identifier *fromid*, has a relationship to another entity identified by *toid*, and the class of the relationship is identified by *verbphrase*.

Specific Core Assertions are defined as follows:

```
node( Id,Noun,Name) :- entity( Id,'Node' ), attribute( Id,noun, Noun), attribute( Id,'name',Name).
owner( Nid1, Nid2 ) :- relationship('nodeOwnsNode',Nid1,Nid2).
```

Semantics of the SPECTRA metadata for SysML v2 defines how custom instances of core assertions for a specific SOI are constructed from a well-formed SysML v2 model with compliant SPECTRA metadata for selected elements.

Custom assertions are facts, for example:

```
node('n01','ReplaceableUnit','SATCOM Radio').
node('n02','Subsystem','Communications Subsystem').
```

owner(‘n02’, ‘n01’).

The definition of the Core Assertion Schema involves the following terms:

Term	Description
schemadef(sid, name, version)	asserting existence of a Core Assertions Schema identified by *sid* and name *name*, also providing a *version* tag for this schema to manage the schema evolution.
entitydef(sid, noun)	asserting that schema identified by *sid* involves an entity class identified by *noun*
attributedef(sid, noun, name, type)	asserting that within the schema identified by *sid* an entity class identified by *noun* involves an attribute identified by *name*, with values belonging to type *type*
relationshipdef(sid, verbphrase, verb, noun1, noun2)	asserting that schema identified by *sid* involves a relationship class identified by *verbphrase*, that involves a verb *verb*, and two *nouns*. The *verbphrase* can be verbalized as “*noun1 verb noun2*”
typedef(sid, type)	asserting that within the schema identified by *sid* involves a primitive type identified by *type*
subclassdef(sid, noun1, noun2)	asserting that within the schema identified by *sid* the class identified as *noun1* is a subclass of the class identified by *noun2*
abstractdef(sid, noun)	asserting that within the schema identified by *sid* the involves an abstract class identified as *noun1*

Definition of the schema constitutes additional facts, for example:

```
schemadef( ‘s1’, “SPECTRA Core Assertions’, ‘1.0’).
abstractdef( ‘s1’, ‘Node’ ).
attributedef( ‘s1’, ‘Node’, noun, ‘String’).
attributedef( ‘s1’, ‘Node’, ‘name’, ‘String’).
abstractdef( sid, ‘Node’ ).
abstractdef( ‘s1’, ‘ReplaceableElement’ ).
entitydef( ‘s1’, ‘ReplaceableUnit’ ).
subclassdef( sid, ‘ReplaceableUnit’ , ‘ReplaceableElement’).
subclassdef( sid, ‘ReplaceableElement’ , ‘Node’).
typedef( ‘s1’, ‘String’).
typedef( ‘s1’, ‘Boolean’).
relationshipdef( ‘s1’, nodeOwnsNode, ‘owns’, ‘Node’,’Node’).
```

B.3 Core Assertions

This section enumerates the Core Assertions. This section focuses on the assertions related to SPECTRA Core entities and provides only a few examples of SPECTRA Core relationships. Full specification of the SPECTRA Core Assertions is provided in the SPECTRA Core Assertions Metamodel specification. The objective of this section is to describe the

key terms used in the semantic formulations in this specification, and to demonstrate the flexibility of the selected approach to describing the semantics of SPECTRA for SysML v2. This approach allows a multitude of "schemas" that can represent the facts extracted from a SysML v2 model annotated with SPECTRA metadata - ranging from a triple store format to a metamodel-driven format. This section demonstrates a hybrid format: instead of defining an individual term for each SPECTRA entity, the list below defines a common class for a selected abstract entity, e.g. "node", where the specific subclass of the entity is represented by its noun. The mapping of these terms is described by simple inference rules, as demonstrated in the Semantic Foundation section above.

Table 21. Specific Core Assertions

Term	Description
node(id,noun,name)	foundation for assertions that an element with certain id and name in a SysML model is one of the subclasses of metadata Node; the noun is the literal representing of the concrete subclasses of Node, e.g. 'ReplaceableUnit'
channel(id, noun, name)	foundation for assertions that element with certain id and name in a SysML model is one of the subclasses of metadata ConveyableElement; the noun is the literal representing of the concrete subclasses of Node, e.g. 'Link'
endpoint(nid, cid)	assertion that a node with id nid is an endpoint for the channel with id cid
owner(nid1, nid2)	assertion that node nid1 is the owner of node with id nid2
member(nid1, nid2)	assertion that node nid1 belongs to group nid2
datatype(id, noun, name)	foundation for assertions involving stereotype InformationType; the noun is the literal representing of the concrete subclasses of OperationalData, e.g. 'ApplicationData' or 'Request'
exchange(id, fromid, toid, dataid)	foundation for assertions that element with id is a Flow
carrierInterface(id, name, p1, p2, p3, isWireless)	foundation for assertions that element with id is a CarrierInterface
datastore(id, name)	foundation for assertions that element with id is a Datastore
agent(id, noun, name)	foundation for assertions that element with id is one of the subclasses of Agent; the noun is the literal representing of the concrete subclasses of Agent, e.g. 'Operator'
capability(id, name)	foundation for assertions that element with id is a Capability
mission(id, name)	foundation for assertions that element with id is a Mission
missionPhase(id, name)	foundation for assertions that element with id is a MissionPhase
function(id, noun, name)	foundation for assertions that element with id is one of the subclasses of FunctionElement; the noun is the literal representing of the concrete subclasses of Node, e.g. 'EmergentFunction'

thread()	foundation for assertions that element with id is a Functional Thread
characteristic(ownerid, noun)	foundation for assertions that element with id is one of the subclasses of Characteristic
artifact(id, noun, ownerid)	foundation for assertions that element with id is one of the subclasses of Artifact; the noun is the literal representing of the concrete subclasses of Artifact, e.g. ‘Hardware’
trait(id, noun, ownerid)	foundation for assertions that element with id is one of the subclasses of Trait; the noun is the literal representing of the concrete subclasses of Trait, e.g. ‘Controller’
internal(id, value)	assertion that element with id is internal (true) or external (false)
boundaryCrossing(id)	assertion that element with id is boundary-crossing
inbound(id)	assertion that element with id is inbound
outbound(id)	assertion that element with id is outbound
input(id)	assertion that operational data with id is input
output(id)	assertion that operational data with is is output
stored(id)	assertion that operational data with id is stored
processed(id)	assertion that operational data with id is stored
impactLevel(id, objective, level)	assertion that impactful element with id has Impact Level
note(id, ownerid, text)	assertion that element with id has a comment or a note

Annex C: References

1. Friedenthal S., Moore A., Steiner R., *A Practical Guide to SysML: The Systems Modeling Language* (2015). Morgan Kaufmann, OMG Press.
2. Friedenthal S., Oster C, *Architecting Spacecraft with SysML* (2017). AIAA.
3. Aleksandravicene A., Morkevicus A., *MagicGrid Book of Knowledge: A Practical Guide to Systems Modeling using MagicGrid from Dassault Systems*, (2021), Dassault Systems, Vitae Litera, Kaunas
4. McSweeny K., Finding a Single Source of Truth with Model-Based Systems Engineering, Northrop Grumman, https://www.northropgrumman.com/what-we-do/digital-transformation/finding-a-single-source-of-truth-with-model-based-systems-engineering
5. Mansourov N., Campara D., Systems Assurance: Beyond Detecting Vulnerabilities (2010), Morgan Kaufman, OMG Press.
6. Moir, I., *Military Avionics Systems* (2001) AIAA Education Series, CRC Press
7. INCOSE, Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, Wiley, 2015
8. [SysML v1] *OMG Systems Modeling Language (SysML),* Version 1.7 https://www.omg.org/spec/SysML/1.7
9. [UML] *Unified Modeling Language (UML),* Version 2.5.1 https://www.omg.org/spec/UML/2.5.1
10. [NIST SP-800-30] *NIST Special Publication 800-30, Guide for Conducting Risk Assessments. Information Security, 2012* https://doi.org/10.6028/NIST.SP.800-30r1
11. [NIST SP-800-37] *NIST Special Publication 800-37, Rev 2, Risk Management Framework for Information Systems and Organizations: A system Life Cycle Approach for Security and Privacy, 2018* https://doi.org/10.6028/NIST.SP.800-37r2
12. [NIST SP-800-53] *NIST Special Publication 800-53, Rev 5, Security and Privacy Controls for Information Systems and Organizations, 2020* https://doi.org/10.6028/NIST.SP.800-53r5
13. [NIST SP-800-53a] *NIST Special Publication 800-53A, Rev 5, Assessing Security and Privacy Controls in Information Systems and Organizations, 2022* https://doi.org/10.6028/NIST.SP.800-53Ar5
14. [FIPS-199] FIPS Pub 199, Federal Information Processing Standards Publication. Standards for Security Categorization of Federal Information and Information Systems, 2004, https://doi.org/10.6028/NIST.FIPS.199
15. [SPDX] ISO/IEC 5962:2021 Information technology – SPDX Specification V2.2.1, https://www.iso.org/standard/81870.html
16. [CycloneDX] CycloneDX Bill of materials specification, ECMA-424, 2024, https://ecma-international.org/publications-and-standards/standards/ecma-424
17. [ISO 42010] ISO/IEC/IEEE 42010:2022, Systems and software engineering – Architecture Description, https://www.iso.org/standard/74393.html
18. ISO/IEC/IEEE 15288:2023, Systems and software engineering – System life cycle processes, https://www.iso.org/standard/81702.html
19. ISO/IEC/IEEE 27005:2022, Information security, cybersecurity and privacy protection – Guidance on managing information security risks, https://www.iso.org/standard/74393.html
20. [KDM] ISO/IEC 19506:2012 Information technology Object Management Group Architecture-Driven Modernization (ADM) – Knowledge Discovery Metamodel (KDM), 2017, https://www.iso.org/standard/32625.html
21. [Prolog] ISO/IEC 13211:1:1995 Information technology — Programming languages — Prolog, Part 1: General Core, 2024, https://www.iso.org/standard/21413.html
22. [MITRE ATT&CK] https://attack.mitre.org

23. [MITRE D3FEND] https://d3fend.mitre.org
24. [CWE] Common Weakness Enumeration https://cwe.mitre.org
25. [CVE] Common Vulnerabilities and Exposures https://cve.mitre.org
26. [NVD] National Vulnerability Database https://nvd.nist.gov