



DPC: 23/30456852 DC

BSI Group Headquarters

389 Chiswick High Road London W4 4AL

Tel: +44 (0)20 8996 9000

Fax: +44 (0)20 8996 7400

www.bsigroup.com

Date: 29 November 2023

Origin: International

Latest date for receipt of comments: 22 January 2024

Project No. 2022/01217

Responsible committee: AMT/10 Robotics

Interested committees:

Title: Draft BS ISO 22166-202 Robotics - Modularity for service robots. Information model for software modules.

Please notify admin.start@bsigroup.com if you are aware of any keywords that might assist in clarifying or identifying the standard or of the content of this standard:

- i) has any issues related to 3rd party IPR, patent or copyright
- ii) affects other national standard(s)
- iii) requires additional national guidance or information

WARNING: THIS IS A DRAFT AND MUST NOT BE REGARDED OR USED AS A BRITISH STANDARD. THIS DRAFT IS NOT CURRENT BEYOND 22 January 2024

This draft is issued to allow comments from interested parties; all comments will be given consideration prior to publication. No acknowledgement will normally be sent. **See overleaf for information on the submission of comments.**

No copying is allowed, in any form, without prior written permission from BSI except as permitted under the Copyright, Designs and Patent Act 1988 or for circulation within a nominating organization for briefing purposes. Electronic circulation is limited to dissemination by e-mail within such an organization by committee members.

Further copies of this draft may be purchased from BSI Shop <http://shop.bsigroup.com> or from BSI Customer Services, Tel: +44(0) 20 8996 9001 or email cservices@bsigroup.com. British, International and foreign standards are also available from BSI Customer Services.

Information on the co-operating organizations represented on the committees referenced above may be obtained from <https://standardsdevelopment.bsigroup.com/>.

E-mail: admin.start@bsigroup.com

Introduction

This draft standard is based on national and international discussions. Your comments on this draft are invited and will assist in the preparation of the consequent standard.

For international standards, comments will be reviewed by the relevant UK national committee before sending the consensus UK vote and comments to the international committee, which will then decide appropriate action. If the international standard is approved, it is usual for the text to be published as a British Standard.

For national standards, comments will be reviewed by the relevant UK national committee and the resulting standards published as a British Standard.

UK Vote

Please indicate whether you consider the UK should submit a negative (with supporting technical reasons) or positive vote on this draft. Please indicate if you are aware of any reason why this draft standard should not be published as a British Standard.

Submission of Comments

- **Annotated drafts are not acceptable and will be rejected.**
- All comments should be submitted online at <https://standardsdevelopment.bsigroup.com/>. You will need to register in order to comment.

Template for comments and secretariat observations

Date: xx/xx/20xx	Document: ISO/DIS xxxx
------------------	------------------------

1	2	(3)	4	5	(6)	7
MB	Clause No./ Subclause No./Annex (e.g. 3.1)	Paragraph/ Figure/Table/Note	Type of comment	Comment (justification for change) by the MB	Proposed change by the MB	Secretariat observations on each comment submitted
	3.1	Definition 1	ed	Definition is ambiguous and needs clarifying.	Amend to read '...so that the mains connector to which no connection...'	
	6.4	Paragraph 2	te	The use of the UV photometer as an alternative cannot be supported as serious problems have been encountered in its use in the UK.	Delete reference to UV photometer.	

DRAFT INTERNATIONAL STANDARD

ISO/DIS 22166-202

ISO/TC 299

Secretariat: SIS

Voting begins on:
2023-11-29

Voting terminates on:
2024-02-21

Robotics — Modularity for service robots —

Part 202:

Information model for software modules

ICS: 25.040.30

THIS DOCUMENT IS A DRAFT CIRCULATED FOR COMMENT AND APPROVAL. IT IS THEREFORE SUBJECT TO CHANGE AND MAY NOT BE REFERRED TO AS AN INTERNATIONAL STANDARD UNTIL PUBLISHED AS SUCH.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

RECIPIENTS OF THIS DRAFT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.

This document is circulated as received from the committee secretariat.



Reference number
ISO/DIS 22166-202:2023(E)

AVA Documents
Document Sharing Hub
© ISO 2023



COPYRIGHT PROTECTED DOCUMENT

© ISO 2023

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword	iv
Introduction	v
1 Scope	1
2 Normative references	1
3 Terms and definitions	2
4 Information model for software modules	4
4.1 General.....	4
4.2 Class model of a software information model.....	5
4.2.1 General.....	5
4.2.2 Class for Module ID.....	6
4.2.3 Class for Properties.....	7
4.2.4 Class for IOVariables.....	12
4.2.5 Class for Status.....	13
4.2.6 Class for Services.....	14
4.2.7 Class for Infrastructure.....	14
4.2.8 Class for SafeSecure.....	16
4.2.9 Class for Modelling.....	16
4.2.10 Class for ExecutableForm.....	17
Annex A (Normative) Assignment Rule of a Software Module ID	18
Annex B (Normative) Representation of common information for software modules	21
Annex C (Informative) Examples for application of the software information model	33
Annex D (Informative) How to use software information models	34
Annex E (Informative) Services	36
Annex F (Informative) Mapping between an Information Model and ROS 2	38
Annex G (Informative) Mapping between an Information Model and OMG SDO/RTC/ OpenRTM's RTCProfile	46
Bibliography	49

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 299 *Robotics*.

A list of all parts in the ISO 22166 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

This document provides an information model for software modules based on ISO 22166-1:2021 Modularity for Service Robots Part 1-General Requirements and DIS 22166-201 Modularity for Service Robots Part 201- Common Information Model for Modules. The information model for software modules defined here is based on current industrial best practices and recent research results in model-driven system engineering. It is designed to enhance the interoperability, reusability, and composability of modules. Using the information model for software modules, a robot system builder and developer of composite modules (as defined in ISO 22166-1) can easily compare modules of different vendors, identify the modules matching their requirements, and easily connect those modules to form a service robot system, a subsystem thereof, or a larger composite module. The information model is a digital datasheet for service robot modules that either are only software or contain software.

This document addresses two main user groups: service robot makers, including service robot system integrators, and providers of modules for service robot systems. Within both user groups, it primarily addresses software developers, including system designers and integrators. In the case of module providers, it further addresses personnel concerned with technical documentation and – to lesser degree – technical marketing. In the case of service robot makers and service robot system integrators, it further addresses personnel concerned with safety engineering and – to lesser degree – personnel concerned with acquiring modules for use in a robot system. Providers of modules for service robots have to provide as complete and accurate model as possible for their modules, including information regarding the runtime requirements of the module, information regarding its interfaces, and information relevant for performing a safety assessment of systems built from such modules. Robot system integrators have to be able to make design decisions based on the information provided by the module makers. Third parties can use the information to build tools to automate or otherwise support parts of the software system integration process

The information model of the robot software modules presented in this document are focused on the common characteristics which all types of software modules shall have and whose examples are as follows:

1. Module ID;
2. Properties;
3. Input and Output Variables;
4. Status;
5. Services;
7. Infrastructure;
8. Safety and Security;
9. Modelling;
6. Executable Forms.

This document focuses on software module-related interfaces, properties, variables, behaviour, and status of modules.

Future editions and parts of this family of International Standards might include more specific requirements on various types of composite modules including those used in service robots.

The document presents guidelines for interoperability, reusability, and composability of software modules for ensuring their effective connectivity and their correct functionality and providing the certification level for published robot system safety/security requirements.

Robotics — Modularity for service robots —

Part 202: Information model for software modules

1 Scope

This document complies with the ISO 22166-1 and DIS 22166-201 on modularity for service robots. This document presents requirements and recommendations for information models for software modules used in service robots.

This document presents how the information model for software modules relates to interoperability, reusability, and composability and takes into account the safety and security provided in the published standards.

It describes a structured method to define the characteristics of a software module, or a module that has a software-related interface (modules with software aspects, as defined in ISO 22166-1:2021). The information model for software modules, which extends the common information model defined in ISO 22166-201, defines a digital datasheet for software modules for service robots. This datasheet provides relevant information for users of the module, where the “user of the module” refers to anyone, who combines the module with other modules and components (as defined in ISO 22166-1:2021) to form a service robot system or any subsystem thereof, like composite modules (defined in ISO 22166-1:2021).

It is the task of the module provider to also provide the information described in this document in the form described herein. A service robot module is only in compliance with the 22166 series of standards, in particular with 22166-1, -201 and -202, if it is accompanied by a matching model.

User of such modules, e.g. robot system integrators, should be able to rely on the model information when designing a service robot system and assessing the robustness and safety characteristics of that system.

This document only defines information needed for software modules.

This document focuses on interfaces, properties, composition, and execution-specific information, which are related to software modules. In particular, the interfaces are classified and described into 2 types such as variables and methods. The document can also be applied to the following software lifecycle stages: the design stage, development stage, operation stage, and maintenance stage.

NOTE 1 The variable as one type of interface is similar to a topic used in ROS and global variables used in general programs. The method as one type of interface is similar to a service used in ROS and a function or method used in general programs.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 22166-1:2021, *Modularity for service robots — Part 1: General requirements*

DIS 22166-201, *Modularity for service robots — Part 201: Common information model for modules*

ISO 13482:2014, *Robots and robotic devices — Safety requirements for personal care robots*

ISO 13849-1:2015, *Safety of machinery — Safety-related parts of control systems — Part 1: General principles for design*

IEC 62061 IEC-62061:2021, *Safety of machinery - Functional safety of safety-related control systems*

IEC 62443-4-2, *Security for industrial automation and control systems — Part 4-2: Technical security requirements for IACS components*

ISO/IEC 19505-1:2012, *Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 1: Infrastructure*

ISO/IEC 19516:2020, *Information technology — Object management group — Interface definition language (IDL) 4.2*

OMG formal/2008-10-01, *Platform Independent Model and Platform Specific Model for Super Distributed Object, Version 1.1*

OMG formal/2019-04-03, *Protocol DDS Interoperability Wire Protocol (DDSI-RTPS) Version 2.3*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply:

3.1 information model

IM
abstraction and representation of the entities in a managed environment, their properties, attributes and operations, and the way that they relate to each other

[SOURCE: ISO 22166-1]

3.2 common information model CIM

meta model of information that all kinds of modules use commonly.

[SOURCE: ISO 22166-201 modified]

3.3 module

component or assembly of components with defined interfaces accompanied with property profiles to facilitate system design, integration, interoperability, and re-use

Note 1 to entry: A module can be an assembly of modules.

[SOURCE: ISO 22166-1 modified]

3.4 module property property

attribute or characteristic of a module

[SOURCE: ISO 22166-1]

3.5 software module SW module

module whose implementation consists purely of programmed algorithms

[SOURCE: ISO 22166-1]

3.6

hardware module
HW module

module whose implementation consists purely of physical parts, including mechanical parts, electronic circuits, and any software, such as firmware, not externally accessible through the communication interface

[SOURCE: ISO 22166-1]

3.7

module with hardware aspects and software aspects
HW-SW Module

module whose implementation consists of physical parts, including mechanical parts and electronic circuits, and any software that is accessible through the communication interface

3.8

module manager

software that manages the module lifecycle, such as instantiating and destroying software modules, and provides various other functions for the module to operate

3.9

instance

particular entity of a specific module's class

Note 1 to entry: In object-oriented programming, "instance" means a specific realization of an object.

3.10

performance level
PL

discrete level used to specify the ability of safety-related parts of control systems to perform a safety function under foreseeable conditions

[SOURCE: ISO 13849-1:2015]

3.11

component

part of something that is discrete and identifiable with respect to combining with other parts to produce something larger

Note 1 to entry: A component can be either software or hardware.

Note 2 to entry: A component is a part, which is not standardized according to the ISO 22166 series.

[SOURCE: ISO 22166-1]

3.12

middleware

software that helps to make communication and data management easy in distributed applications

Note 1 to entry: Middleware provides services to software applications beyond those available from the operating system.

Note 2 to entry: Middleware may be a component on which groups of components and/or modules are executed. ROS, OpenRTM, and OPRoS are types of middleware.

3.13

hardware abstraction interface (HAI)

abstraction interface for a component/module that contains hardware aspects, with the abstraction interface providing control of the component/module via a software interface

[SOURCE: ISO 22166-1 modified]

3.14

sensing software module (will be removed if not used in main text)

software module for collecting or acquiring data about the world around the robot or the state of the robot for use by other modules to support the robot system in performing its task(s)

Note 1 to entry: The module can access HW-SW modules such as lidars, encoders, and cameras using HAI, device drivers or dedicated drivers.

EXAMPLE lidar sensing software modules, camera sensing software modules, force sensing software modules, etc.

4 Information model for software modules

4.1 General

The information for software modules (or software information model, SIM) shall consist of items shown in [Table 1](#), which is based on DIS 22166-201, the common information model.

Table 1 — Software information and the corresponding tag

NO.	ITEMS	SW module	Related Group/Tag name (Abbreviation of each group)
1	Module Name	M	GenInfo
2	Description	O	
3	Manufactures	M	
4	Examples	O	
5	Information model version	M	IdnType
6	Module ID	M	
7	Software Aspects	O	
8	Module properties ¹⁾	M	ModuleProp
9	Inputs	O	IOVariable
10	Outputs	O	
11	Status	M	-
12	Services (capabilities)	M	Service
13	Infrastructure	M	Infra
14	Safety/security	M	SafeSecure
15	Modelling	O	Modelling
16	ExecutableForm	M	ExecutableForm
M: Mandatory, O: Optional,			
1) It is mandatory only to those which can be influenced (set) from the outside or at least to those, which have an expected effect on other modules.			

The relationship between CIM and software information model (SIM) is shown in [Figure 1](#). That is, SIM is inherited from CIM. Like CIM, SIM can be described using the Unified Modelling Language (UML).

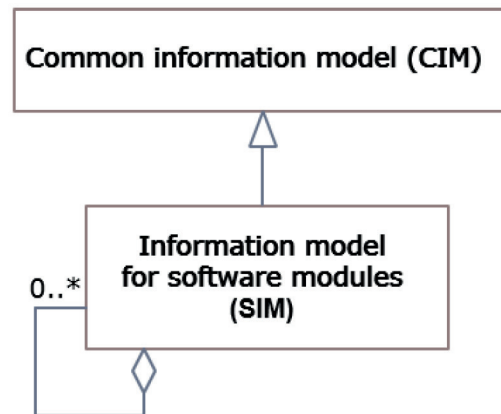


Figure 1 — Relationship between common information model and software information model

NOTE 1 [Annex D](#) provides how to use the software information model and includes information usage among various stakeholders such as module maker and system integrator.

4.2 Class model of a software information model

4.2.1 General

The software information model shall inherit from the common information model specified in DIS 22166-201, which is shown in [Figure 1](#). Access specifiers for the attributes used in SIM shall be public, which are the same as those of CIM.

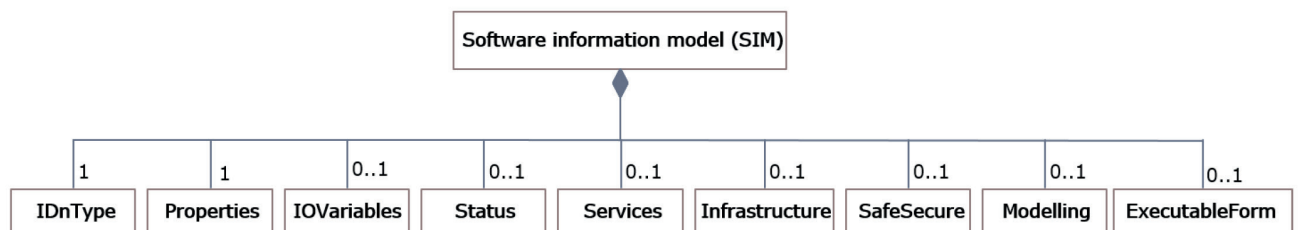


Figure 2 — Class of the software information model

The class of the software information model shall have attributes given in [Figure 2](#) and the attributes for the class SIM are shown in [Figure 3](#).

Values of attributes such as ModuleName, Description, Manufacturer, and Examples are provided in [Annex B](#). The information model version is the version number of the information model used when the module is specified, and is updated whenever the document is upgraded. IDnType, Properties, IOVariables, Status, Services, Infrastructure, SafeSecure, Modelling, and ExecutableForm in [Figure 3](#) refer to the class described in [4.2.2-4.2.10](#). Most of the information about a module is usually fixed when the module is implemented/manufactured/produced. The information shall be provided in file format. Hence, the information shall not be modified during the execution of the module.

NOTE 1 The access specifier for an attribute can be one of followings: private (-), protected (*), or public (+). The attribute name is given first and the data types of attributes are defined next. The separation symbol between the attribute name and its data type is a colon ":". If attributes are declared as public, it is not necessary to define the functions to access those attributes.

NOTE 2 Explanations not presented in this document refer to the corresponding part of the CIM.

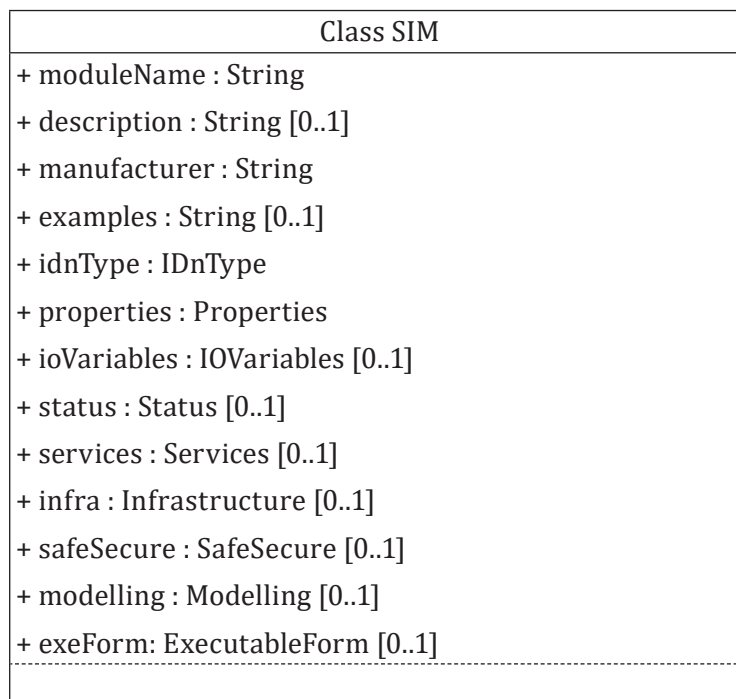


Figure 3 — Class diagram of class SIM

EXAMPLE 1 [Annex C](#) illustrates an information model for a SLAM (Simultaneous Localization and Mapping) module, exemplifying the concept of a composite module based on the class SIM.

4.2.2 Class for Module ID

Information for Module ID shall be defined in the class IDnType. Class IDnType shall have attributes given in [Figure 5](#) and [Table 3](#). Values of the attribute, “moduleID” and other attributes in [Figure 4](#) and [Table 2](#) are provided in [Annex A](#) and [Annex B](#). SW modules shall be classified into singleton and multiton. The former means that only one instance is created from a SW module and the latter means that multiple instances may be created from a SW module. For singleton type of modules, Instance ID (IID) shall be zero. For multiton type of modules, IID shall be assigned to unsigned integer of 8 bits, starting from 0 and increased by one, whenever a new instance is created.

NOTE 1 Types of instances can be a process type or thread type.

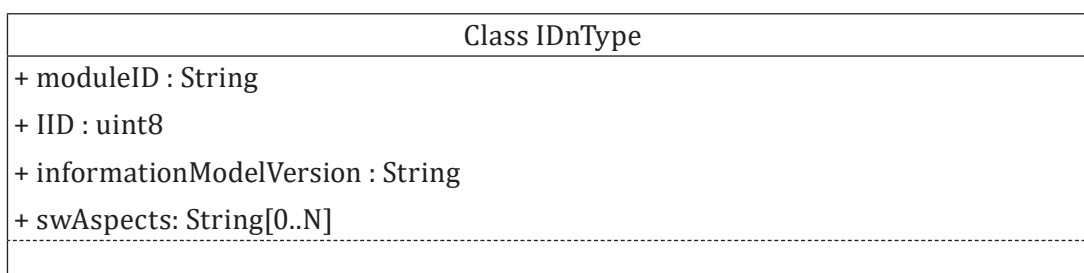


Figure 4 — Class diagram of class IDnType

Table 2 — Description of Class IDnType

Description: Class IDnType for the software information model				
Derived From: Class IDnType in CIM				
Attributes:				
moduleID	String	M	1	The module ID of a module except instance ID
IID	uint8	M	1	InstanceID (IID), default value is 0

Table 2 (continued)

informationModelVersion	String	M	1	Version number of the information model
swAspects	String	O	N	List of IDs of child modules, whose root module is located at the first level in Figure 5 .

4.2.3 Class for Properties

The class Properties is shown in [Table 15](#) and Figure 6. The class Properties shall consists of 4 mandatory classes and 1 optional class. The four mandatory classes are class OStype, class CompilerType, class ExecutionType, and class Libraries. The one optional class is class Property, which shall be defined if a software module uses additional properties except properties defined in the above 4 mandatory classes. The class Property is modified based on DIS 22166-201. Information for the class Property shall be provided using XML or JSON, where the XML format is provided in [Annex B](#).

The class OStype is shown in [Table 5](#), where:

The attribute "type" is the type of operating system (OS).

The attribute "bits" is the number of bits that the operating system supports.

The attribute "version" is the version of the operating system.

EXAMPLE 1 Consider the following as examples of OSs: Windows, Android, MacOS, iOS; Linux OSs: Debian, Gentoo, Ubuntu, Mint, Red Hat, CentOS, Fedora, Kali, Arch, OpenSUSE; Real-time OSs: VxWorks, OSE, VRTX, pSOS, Nucleus, SuperTask, uC/OS, QNX, OS-9, LynxOS, WindowsCE, RTX, Xenomai, RTLinux, and RTAI.

The class Library is shown in [Table 5](#), where:

The attribute "name" is the name of a library.

The attribute "version" is the version of that library.

The class Libraries is shown in [Table 6](#), where:

The attribute "libraries" includes the list of all libraries used in the module.

EXAMPLE 2 If a module use libraries such as OpenCV 4.5.3 and Boost 1.76.0. Then, these libraries have to be included in the class Libraries.

The class CompilerType is shown in [Table 8](#), and the class RangeString used in the class CompilerType is shown in [Table 7](#). For class CompilerType:

The attribute "osName" is the target operating system name.

The attribute "verRangeOS" is the supported version range of the OS.

The attribute "compilerName" is the name of the target compiler/interpreter suite.

The attribute "verRangeCompiler" is the supported version range of the compiler/interpreter.

The attribute "bitssnCPUarch" is the target bit-size and CPU architecture.

NOTE 1 If verRangeOS or verRangeCompiler has only one value such as only one available version number, the attribute "min" is used for storing the version number and the attribute "max" is set to NULL.

NOTE 2 If verRangeOS or verRangeCompiler is available from the specified version number to the higher version number, the attribute "min" is the specified version number and the attribute "max" is the string "Higher".

EXAMPLE 3 Consider the following table which includes some values of compilers as examples.

Attribute	Example OS type 1	Example OS type 2
osName	Windows	Ubuntu
verRangeOS (min, max)	7, 10	18,04, 20.04
compilerName	Microsoft Visula C++ 2017	GCC
verRangeCompiler (min, max)	14.0, Higher	9.0, Higher
bitsnCPUarch	64, X86 (min, max)	32, armv7hf

The class ExecutionType shown in [Table 11](#) has the following 5 attributes: “opType”, “hardRT”, “timeConstraint”, “priority”, and “instanceType”, where:

The attribute “opType” is the operation type of a software module, which shall have one of following enumeration values: {PERIODIC, EVENTDRIVEN, NONRT}, where “NONRT” means non real-time, which is shown in [Table 9](#).

The attribute “hardRT” is used to set whether the hard real-time property must be satisfied. If so, the value is TRUE. Its default value is “FALSE”.

The attribute “timeConstraint” is used to set the period for the operation type of “PERIODIC” and the deadline for the operation types of “EVENTDRIVEN”. Its default value is 0, which means that the related item is not used.

The attribute “priority” is used to set the instance’s priority of the software module.

The attribute “instanceType” is related to the instance of a module and shall have one of following enumeration values: {Singleton, MultitonStatic, MultitonCommutative}, as shown in [Table 10](#).

NOTE 3 Singleton means that only one instance is created. MultitonStatic means that one or more instances are created, but the created instance is linked to only one hardware-software module such as a sensor module or an actuator module. MultitonCommutative means that one or more instances are created and can be exchanged between modules with the same type.

NOTE 4 The attributes “priority” and “timeConstraint” for MultitonStatic and MultitonCommutative instances can be modified after their creation.

The class Organization, which is related to the composite module, is shown in [Table 13](#), and has the following 2 attributes: “owner” and “dependency” where:

The attribute “owner” is the owner of the module and has the module ID value of the owner.

The attribute “dependency” shall have one of following enumeration values: {OWNER, OWNED, OWNEROWNED, NONE}, as shown in [Table 12](#).

If a module is a composite type, the structure of the module has a hierarchical type, an example of which is shown in [Figure 5](#). The attribute “dependency” shall be set using a dependency type in [Table 12](#). All software modules, which are member modules of a composite module, shall have the class Organization with the proper dependency type.

NOTE 5 Member modules of a composite module are listed in the attribute “swAspects” of class IDnType in [4.2.2](#).

EXAMPLE 4 [Figure 5](#) is an example of a composite module, where dependency types are shown. The composite module consists of two basic modules and one composite module, where the member composite module also consists of one composite module and two basic modules.

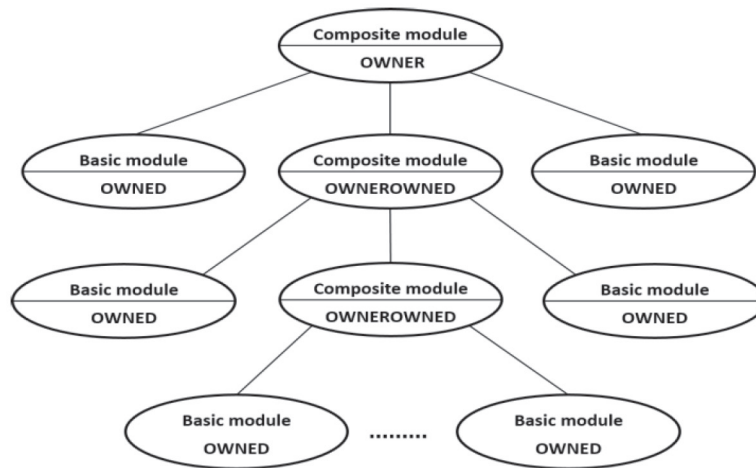


Figure 5 — Example diagram of composite module with dependency type

The class Property is shown in [Table 14](#) and shall inherit the class DataProfile, which is shown in Figure 4.6 in DIS 22166-201. A newly added attribute for SIM is the attribute “immutable”, which determines whether it can be changed to a new value during the operation of the module.

Table 3 — Enumeration NoBits

Description: Number of bits processed in OS, enumeration type	
Attributes:	
16	16 bits
32	32 bits
64	64 bits

Table 4 — Class OSType

Description: Defines the OS type that a module uses				
Derived From: None				
Attributes:				
type	String	M	1	OS types such as Windows or Linux, case insensitive
bits	NoBits	M	1	See table 3
version	String	M	1	OS version number

Table 5 — Class Library

Description: Defines a library that a module uses				
Derived From: None				
Attributes:				
name	String	M	1	Library name, case insensitive
version	String	M	1	Library version number

Table 6 — Class Libraries

Description: Defines libraries using class Library				
Derived From: None				
Attributes:				
libraries	Library	M	N	List of libraries, case insensitive

Table 7 — Class RangeString

Description: Class representing a range of minimum and maximum values				
Derived From: None				
Attributes:				
min	String	M	1	Minimum value
max	String	M	1	Maximum value

Table 8 — Class CompilerType

Description: Defines a compiler or interpreter that a module uses. This class is related to class Executable-Form described in 4.2.10				
Derived From: None				
Attributes:				
osName	String	M	1	Target OS name.
verRangeOS	RangeString	M	1	Supported version range of OS
compilerName	String	M	1	Name of the target compiler/interpreter suite,
verRangeCompiler	RangeString	M	1	Supported version range of compiler/ interpreter
bitsnCPUarch	String	M	1	Target bit-size and CPU architecture

Table 9 — Enumeration OpTypes

Description: Operation type of a software module	
Attributes:	
PERIODIC	Instance that operates or runs periodically
EVENTDRIVEN	Instance that operates or runs sporadically according to an occurrence of events
NONRT	Non-real-time instance

Table 10 — Enumeration InstanceTypes

Description: Instance type that a module can create	
Attributes:	
Singleton	Create only a single instance
MultitonStatic	Create multiple instances but tied to specific hardware such as robot arms or sensors
MultitonComm	Create multiple instances dynamically, Its example is just an instance of a module for processing.

Table 11 — Class ExecutionType

Description: Class providing the information needed to run an instance of a module				
Derived From: None				
Attributes:				
opType	OpTypes	M	1	Operation types that a module instance runs. (See Table 9)
realtime	Boolean	M	1	If the instance runs with hard real-time, this value is TRUE. Otherwise FALSE. If opType is NONRT, the value of realtime shall be FALSE