NATO STANDARD

AEP-4754

NATO GENERIC VEHICLE ARCHITECTURE (NGVA) FOR LAND SYSTEMS

VOLUME V: DATA MODEL

Edition B Version 1
FEBRUARY 2023

NORTH ATLANTIC TREATY ORGANIZATION

ALLIED ENGINEERING PUBLICATION

Published by the
NATO STANDARDIZATION OFFICE (NSO)
© NATO/OTAN

INTENTIONALLY BLANK

NORTH ATLANTIC TREATY ORGANIZATION (NATO)

NATO STANDARDIZATION OFFICE (NSO)

NATO LETTER OF PROMULGATION

3 February 2023

1. The enclosed Allied Engineering Publication AEP-4754, Volume V, Edition B, Version 1 NATO GENERIC VEHICLE ARCHITECTURE (NGVA) FOR LAND SYSTEMS VOLUME V: DATA MADEL, which has been approved by the nations in the NATO Army Armaments Group (AC/225 NAAG), is promulgated herewith. The agreement of nations to use this publication is recorded in STANAG 4754.

2. AEP-4754, Volume V, Edition B, Version 1 is effective upon receipt and supersedes AEP-4754, Volume V, Edition A, Version 1, which shall be destroyed in accordance with the local procedure for the destruction of documents.

3. This NATO standardization document is issued by NATO. In case of reproduction, NATO is to be acknowledged. NATO does not charge any fee for its standardization documents at any stage, which are not intended to be sold. They can be retrieved from the NATO Standardization Document Database ((https://nso.nato.int/nso/) or through your national standardization authorities.

4. This publication shall be handled in accordance with C-M(2002)60.

[illegible]

Dimitrios SIGOULAKIS
Major General, GRC (A)
Director, NATO Standardization Office

INTENTIONALLY BLANK

RESERVED FOR NATIONAL LETTER OF PROMULGATION

INTENTIONALLY BLANK

RECORD OF RESERVATIONS

CHAPTER	RECORD OF RESERVATION BY NATIONS
Note: The reservations listed on this page include only those that were recorded at time of promulgation and may not be complete. Refer to the NATO Standardization Document Database for the complete list of existing reservations.	

INTENTIONALLY BLANK

RECORD OF SPECIFIC RESERVATIONS

[nation]	[detail of reservation]
Note: The reservations listed on this page include only those that were recorded at time of promulgation and may not be complete. Refer to the NATO Standardization Document Database for the complete list of existing reservations.	

INTENTIONALLY BLANK

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

1.1. Purpose

The aim of the NGVA Standard AEP-4754 Volumes I through VII is to enable the member nations to realize the benefits of an open architecture approach to Land vehicle platform design and integration, especially in regard to the vehicle platform electronic data and power infrastructure and the associated safety and verification & validation process.

1.2. Application of the NGVA Standard

The NGVA Standard is to be applied to all future land vehicle platforms and vehicle platform sub-systems, as well as current vehicle platform refurbishment and upgrade programmes.

This NGVA Standard is applicable to land vehicle platforms, ranging from simple to complex implementations. The requirements for these implementations are determined by the functionality required by the vehicle platform as a whole system including all sub-systems, and not the automotive or power elements alone. The requirements address equipment to be fitted as part of the initial operating capability and equipment likely to be fitted throughout the life of the vehicle platform. These requirements are expressed in the national system requirements documents and/or the sub-system requirements documents for the individual vehicle platforms concerned.

1.3. Agreement

Ratifying nations agree that the NGVA Standard is to be applied to all future land vehicle platforms and vehicle platform sub-systems, as well as current vehicle platform refurbishment and upgrade programmes. Nations may propose changes at any time to the NATO Standardization Office (NSO).

Germany will act as custodian to maintain Configuration Management (CM) and change management of this Standard and its associated AEP Volumes.

Ratifying nations have agreed that national orders, manuals and instructions implementing this Standard will include a reference to the AEP 4754 Volumes I through VII for purposes of identification.

The NGVA Standard and its associated Volumes I through VII shall be considered as the foundation standard for vehicle sub-system integration, and should any conflict arise between this and other extant NATO documentation, this document shall take precedence.

Deviations from the NGVA Standard shall be agreed by the relevant national procurement office.

1.4. Ratification, implementation, and reservations

Ratification, implementation and reservation details are available on request or through the NATO Standardization Office (NSO) (internet: http://nso.nato.int).

1.5. Feedback

Any comments concerning this publication should be directed to: NATO/NSO – Bvd Leopold III - 1110 Brussels - Belgium.

Proposals for changes and improvements of the NGVA Standard AEP 4754 volumes I through VII shall be sent to the NSO and then forwarded to the custodian who will collect them and will propose new editions of the NGVA Standard AEP 4754 Volumes 1 through 7.

The NGVA Standard Point-of-Contact as assigned by the NGVA Standard Custodian is BAAINBw K1.2, Ferdinand-Sauerbruch-Str.1, D-56073 Koblenz, Germany.

CHAPTER 2 DEVELOPMENT OF NGVA STANDARD

The NATO Generic Vehicle Architecture (NGVA) Standard was developed under the auspices of the Military Vehicle Association (MILVA).

MILVA is an association of government agencies and industries promoting Vehicle Electronics (Vetronics) in the military environment. MILVA provides an open forum to its members and publishes guidelines and standards on Vetronics issues. MILVA works in close co-operation with NATO through the Land Capability Group on Land Engagement of the NATO Army Armament Group (NAAG).

2.1. NGVA Standard Structure

Figure 1 below illustrates the Standard structure, the Volumes relationships and technical areas covered under each Volume.

NGVA Standard AEP 4754

Volume I:	NGVA Architecture Approach (Describes the NATO Generic Vehicle Architecture (NGVA) concept)
Volume II:	NGVA Power Infrastructure (Defines the design constraints on power interfaces which form the NGVA Power Infrastructure)
Volume III:	NGVA Data Infrastructure (Defines the design constraints on the electronic interfaces that form the NGVA Data Infrastructure)
Volume IV:	NGVA Crew Terminal Software Architecture (Defines the design guidelines and constraints for standardized “Crew Terminal Software Applications”)
Volume V:	NGVA Data Model (Describes the NATO GVA Data Model (NGVA DM) approach used to produce the NGVA DM, the delivery and change management processes and finally gives implementation and deployment guidance)
Volume VI:	NGVA Safety (Outlines the generic procedures to incorporate system safety related planning, development, implementation, commissioning and activities in systems engineering)

Volume VII:	NGVA Verification and Validation (Provides guidance for the verification and validation of NGVA systems regarding their conformity to the AEPs associated with this STANAG)

Figure 1: NGVA Standard AEP 4754

2.2. General Notes

2.2.1. Scope

NGVA is the approach taken by NATO and related industry to standardize the interfaces and protocols for military vehicle systems integration. The Vehicle Architecture (including data and power architectures) is considered as the fundamental enabler that can provide new capabilities on military platforms so as to improve overall effectiveness (including cost) and efficiency within the whole vehicle life cycle. The NGVA Standard does not include standard automotive electronics and automotive power related information.

2.2.2. Warning

National governments, like their contractors, are subject to laws of their respective countries regarding health and safety. Many NATO STANAGs and Standards set out processes and procedures that could be hazardous to health if adequate precautions are not taken. Adherence to those processes and procedures in no way absolves users from complying with their national legal requirements.

2.3. Normative References

The documents and publications shown in Table 1 below are referred to in the text of this AEP Volume as normative. Documents and publications are grouped and listed in alpha-numeric order:

1. DDS	OMG – Data Distribution Service (DDS) (https://www.omg.org/spec/DDS/)
2. DDS Consolidated XML Syntax	OMG-DDS Consolidated XML Syntax (https://www.omg.org/spec/DDS-XML/)
3. DDSI-RTPS	OMG – The Real-time Publish-Subscribe Protocol DDS Interoperability Wire Protocol (https://www.omg.org/spec/DDSI-RTPS/)
4. DDS-XTypes	OMG - Extensible and Dynamic Topic Types for DDS (https://www.omg.org/spec/DDS-XTypes/)
5. IDL	OMG – Interface Definition Language (https://www.omg.org/spec/IDL/)
6. UML Specification	OMG – Unified Modeling Language https://www.omg.org/spec/UML/

Table 1: Normative References

Reference to NGVA Data Model implementation-related standards, e.g. OMG standard versions, will be included in the Reference Model Delivery Package.

Reference in NGVA Standard AEP-4754 to any normative references refers to, in any Invitation to Tender (ITT) or contract, the edition and all amendments current at the date of such tender or contract, unless a specific edition is indicated. For some standards, the most recent editions shall always apply due to safety and regulatory requirements.

In consideration of the above and as best practice, those setting the requirements shall be fully aware of the issue, amendment status and application of all normative references, particularly when forming part of an ITT or contract.

2.4. Conventions

For the purposes of all AEP Volumes all requirements are specifically detailed in tables with each requirement classified as in the paragraph 2.5. Where an AEP Volume contains no specific requirement tables they should serve as implementation guidance until technical standardization requirements are developed and included.

2.5. Requirements Classifications

The following classifications are to be used for all NGVA related requirements.

2.5.1. Compulsory Requirement (CR)

The requirement needs to be implemented in order to conform to NGVA Standard AEP-4754 and to gain certification. Compulsory requirements are listed in the Requirements Tables inside the AEPs and marked as “CR”.

2.5.2. Optional Enhancement (OE)

Optional Enhancements do not need to be implemented in order to conform to NGVA Standard AEP-4754. However, if such a capability is present, it needs to be implemented according to the stated specification in order to be compliant. Optional Enhancements are listed in the Requirements Tables inside the AEPs and marked as “OE”.

2.6. Abbreviations

Abbreviations referred to in this AEP Volume are given in ANNEX C.

2.7. Terms and Definitions

2.7.1. NGVA Definitions

1. **Base Vehicle**: The basic vehicle structure and those systems needed to enable it to perform its automotive functions and mobility. Where fitted it also includes those systems needed to control turrets and other physical elements e.g. a mine plough.
2. **Base Vehicle Sub-System**: A system that forms part of the base vehicle.
3. **Crew Terminal**: An electronic hardware device that is used for entering data into and presenting visual and audio data from the NGVA Data Infrastructure connected to the Base Vehicle and all its Mission Sub-Systems.
4. **Electronic Architecture**: The combination of the electronic based sub-systems and electronic infrastructure that supports the vehicle crew to undertake their operational tasks.

5. **NATO Generic Vehicle Architecture (NGVA):** The term ‘NATO Generic Vehicle Architecture’ refers to the open, modular and scalable architectural approach applied to the design of vehicle platforms.
6. **Mission Sub-System:** Separable elements or collections of equipment or software added to a Vehicle Platform that provides operationally required capabilities over and above those delivered by the base vehicle.
7. **Modular**: A modular Electronic Architecture is designed in such a way as to allow the replacement or addition of Mission Sub-Systems and upgrades as required without any undesirable emerging properties.
8. **NGVA Compliant:** NGVA Compliance applies to the whole Vehicle Platform and all added Mission Sub-Systems and means that the Electronic Architecture of the Vehicle Platform complies with the requirements defined in NGVA Standard AEP-4754.
9. **NGVA Data Infrastructure:** The physical cables and connectors that provide means of distributing data around a Base Vehicle. It also includes any enabling logical components and functions e.g. core platform management software, interface software, transport protocols and message definitions.
10. **NGVA Power Infrastructure:** The physical cables, connectors and other components that provide the means of distributing and controlling electrical power around a Base Vehicle.
11. **NGVA Ready**: NGVA Ready applies at a sub-system level and means that sub-systems and components have been developed to a level where they can be efficiently integrated within a “NGVA Compliant” vehicle Electronic Architecture. This would mean passing an incremental process with two sequentially-related Compatibility levels:
    a. **Connectivity Compatibility**: Ensures that the (sub-) system can be physically connected to the NGVA Power and Data Infrastructure without any negative impacts to existing NGVA (sub-) systems. Physical power and network interfaces comply with the requirements of Power and Data Infrastructure AEPs.
    b. **Communication Compatibility**: Connectivity Readiness and data interfaces (DDS/PLEVID) with associated NGVA Data Model implementation that comply with the requirements of Data Model and Data Infrastructure AEPs.
12. **Operator:** Any person required to monitor and control vehicle sub-systems.
13. **Power Management:** The means of prioritizing and controlling the electrical power loads throughout the Vehicle Platform.
14. **Scalable**: The trait of a system in being able to scale in order to handle increased loads of work.
15. **System:** A combination, with defined boundaries, of elements that are used together in a defined operating environment to perform a given task or achieve a specific purpose. The elements may include personnel, procedures, materials, tools, products, facilities, services and/or data as appropriate.
16. **Vehicle Crew:** All personnel located in the Vehicle Platform with defined roles needed to fulfil the necessary operational functions.
17. **Vehicle Platform**: The platform for the Mission Sub-Systems, which comprises all Base Vehicle Sub-Systems, the NGVA Power and Data Infrastructure and all common sub-systems, such as; crew terminals, processing units and other common platform resources (e.g. sighting systems).

2.7.2. AEP Specific Definitions

1. **Class**: A class describes a set of objects that share the same specifications of features, constraints, and semantics [6]. A Class is an element of a Class Diagram; one of the diagram types that form UML.
2. **Configuration Item:** A Configuration Item (CI) is an artefact that is treated as a self-contained unit for the purposes of identification and change control. All CIs are uniquely identified by CI version numbers. In the NGVA DM context, a CI is primarily one or more DM domains and, if applicable, associated specification documents.
3. **DDS:** Data Distribution Service (DDS) is a Data-Centric Publish-Subscribe (DCPS) model for distributed application communication and integration [1]. The DDS middleware enables the delivery of information from information producers to matching consumers.
4. **DDS DataReader**: A DataReader is attached to a DDS Subscriber and allows an application to access received typed DDS data. [derived from 1]
5. **DDS DataWriter**: A DataWriter acts as a typed accessor to a DDS Publisher. The DataWriter communicates data-objects of a given type to a publisher for dissemination. [derived from 1]
6. **DDS Domain**: A domain is a distributed concept that links all the applications able to communicate with each other. It represents a communication plane: only the publishers and the subscribers attached to the same domain may interact. [1] Each Domain is uniquely identified by an integer domain ID that assures domains are completely independent from each other.
7. **DDS Publisher:** A Publisher is an object responsible for data distribution. It may publish data of different data types using different DataWriters. [derived from 1]
8. **DDS Subscriber:** A Subscriber is an object responsible for receiving published data and making it available to the receiving application. It may receive and dispatch data of different specified types. [derived from 1]
9. **Data Model Domain**: Each DM domain represents a distinct subject matter within the NGVA DM addressing a specific aspect of a vehicle sub-system or service. It aims at defining the concepts and messages to be realized by a sub-systems/service in a platform-independent manner.
10. **IDL:** IDL is a descriptive language used to define data types and interfaces in a way that is independent of the programming language or operating system/processor platform. [5]
11. **Middleware**: Middleware is software that provides services beyond those provided by the operating system to enable the various components of a distributed system to communicate and manage data.
12. **Model Driven Architecture**: Model Driven Architecture (MDA) is an open specification involving business-specific platform-independent modelling of application-subject matters (domains) and translation of these models to one or more platform-specific models and to platform-specific implementations afterwards.
13. **NGVA Data Model**: A Data Model is formed from DM domains defining behavior and types used as a basis for the generation of DDS Topics that are exchanged on a NGVA compliant vehicle platform.
14. **NGVA Resource**: An entity of a Mission Sub-System comprising software, and optionally hardware, which publishes or subscribes to DDS topics that are uniquely

identifiable within an NGVA-based Vehicle Platform. A Mission Sub-System may implement multiple NGVA Resources.

15. **Operation**: An operation is a behavior feature that models the way a behavior is invoked in UML. It specifies the entry point to a behavior; however the behavior itself might be modeled independently of the operation and in different ways, e.g. an activity or a state machine.
16. **PIM**: A Platform Independent Model is a UML model, which is independent of any hardware and software platform.
17. **PSI**: A Platform Specific Implementation is the programming language level representation of the Platform Specific Model.
18. **PSM**: A Platform Specific Model is a UML model, which includes deployment information relating to a specific hardware or software platform.
19. **QoS:** Quality of Service (QoS) is a general concept that is used to specify the behavior of a service. Programming service behavior by means of QoS settings offers the advantage that the application developer only indicates 'what' is wanted rather than 'how' this QoS should be achieved. [1]
20. **QoS Policy:** A QoS Policy is an independent description of a specific DDS behavior attribute that associates a name with a value [1]. It can also be referred as QoS Parameter.
21. **QoS Profile:** The QoS Profile is a set of QoS Policies conveniently configured.
22. **QoS Pattern:** A QoS Pattern identifies a DDS behavior, which is associated to a category of system information flow, e.g. Command, State. It is implemented via a QoS Profile.
23. **Topic:** A topic is a DDS entity that associates a name (unique in the DDS domain), a data-type (topic type), and a related set of DDS QoS parameters [derived from 1].
24. **Topic Instance:** The Topic Instance is the instantiation of a Topic. A given Topic Instance is uniquely identified by the value of the "key" fields as defined in the Topic Type. A Topic can be instantiated by one or more Topic Instances. A given Topic Instance can be supported by many Data Writers.
25. **Topic Type:** A topic type is a DDS data structure that has a name and a number of attributes.
26. **UML**: Unified Modeling Language (UML) is a specification defining a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems [6].

CHAPTER 3 NATO GENERIC VEHICLE ARCHITECTURE DATA MODEL

3.1. NGVA Data Model Introduction

The NGVA DM is the expression of the data interface needs for a NATO land vehicle integrated mission system, modelled in a technology independent way. It enables automatic generation of technology specific (DDS) data interfaces for vehicle sub-systems and components. These data interfaces can then be incorporated into software applications embedded on a vehicle platform that supports standardized data distribution over an Ethernet network. The NGVA Data Model is a set of agreed upon data model domains that have achieved the desired level of maturity to be part of a given version of the NGVA standard.

The NGVA DM, along with associated auto-generated artefacts, employs the industry standards Unified Modelling Language (UML) and Interface Definition Language (IDL) to define the data structure and format to be used by sub-systems and components communicating via Data Distribution Service (DDS) middleware installed on a compliant land platform.

The NGVA DM is a data interface 'Reference' model used when integrating a mission sub-system into a vehicle platform. Each NGVA compliant vehicle platform and each NGVA ready mission sub-system implement those data model domains of a specific release of the NGVA DM that are appropriate to its requirements.

Unique ID	Requirement Type	Requirement Text
Use of Data Model Domains		
NGVA_DM_001	CR	An NGVA compliant vehicle platform or NGVA ready mission sub-system shall implement the DM domains needed to fulfill its requirements.

Table 2: Data Model Domain Usage Requirements

The released versions of the NGVA Reference Model Delivery Package are accessible at the NGVA Home Page. On behalf of NATO, MILVA maintains a Data Model repository that holds all NATO Data Model releases.

3.2. The Model Driven Architecture Strategy

The Model Driven Architecture (MDA) strategy, adopted for the NGVA DM, is based on a combination of modularity, abstraction, and automation. Abstraction means that modelers construct a set of simple but precise models of the system data architecture, without prejudice to the implementation technologies. The top-level data architecture is defined in terms of subject matter domains, which typically correspond to sub-system functions and services on a vehicle platform.

The NGVA DM specifies the classes, attributes, relationships, operations and states encapsulated within each domain in an abstract, execution platform independent way. Thereby, it formalizes the message exchange and behavior, both at the system level and the component level, to which any NGVA compliant system must conform, in a clear and unambiguous model based form with human-readable graphical views that could not easily be achieved using textual message definitions.

Use of automation in the form of model translators eliminates the need to manually create and maintain derived artefacts, such as the design model, message definitions and documentation, and ensures that all artefacts are mutually consistent.

Both the MDA approach and the UML notation are defined in specifications owned and maintained by the Object Management Group (OMG), which is an international not-for-profit technology standards consortium. MDA-based UML modelling enables architects to capture and formalize expertise in a way that is uncontaminated by specific implementation technologies. This has the dual benefits of:

1. Ensuring the models become long-life reusable assets, immune to technological changes, both over time and across differing deployment platforms, and
2. Making the models simpler, as each domain model confines itself to a single subject matter, and is uncontaminated with platform-specific implementation details.

Platform Independent Models that embed technology-agnostic definitions of the data required on a vehicle platform, and model translators that embed system-wide NGVA implementation policies enable generation of highly structured, consistent interface definitions for intra-platform communication. These definitions, expressed using Interface Definition Language (IDL) are compiled into executable code for deployment.

The strategic goal of the NGVA DM is that use of the same data definitions by all vehicle electronic architectures will achieve a high level of data compatibility and interoperability between components, reducing the cost and risk of platform integration and through-life maintenance and upgrade.

3.3. Platforms and Platform Independence

A central concept in MDA is the Platform Independent Model (PIM). In this context, the term “platform” refers to the “execution platform” onto which the PIM will ultimately be deployed, typically by automatic translation via a Platform Specific Model (PSM) and Platform Specific Implementation (PSI). The term “execution platform” covers the set of technologies chosen for system deployment.

MDA in the context of the NGVA is based upon a number of fundamental concepts, which are summarized in Figure 2 and outlined in the following sections. This figure illustrates that in the case of NGVA, the route to deployment is via the “Publish/Subscribe” PSM and then to a PSI based on IDL tailored for the Data Distribution Service (DDS).

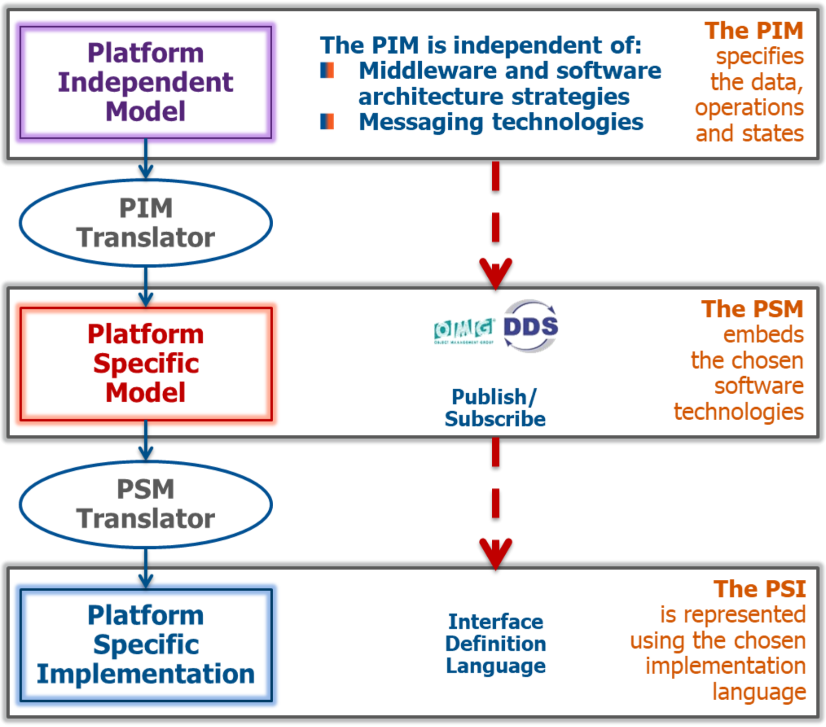

Platform Independent Model
The PIM is independent of:
Middleware and software architecture strategies
Messaging technologies
The PIM specifies the data, operations and states
PIM Translator
Platform Specific Model
DDS
Publish/ Subscribe
The PSM embeds the chosen software technologies
PSM Translator
Platform Specific Implementation
Interface Definition Language
The PSI is represented using the chosen implementation language

Figure 2: MDA Model Transformations

3.4. NGVA Data Model Domains

The primary unit of modularity in MDA is the domain. A data model domain represents a single subject matter. The MDA benefits of simplicity, portability and reuse which are realized primarily by separation of concerns into data model domains, and avoidance of "domain pollution". Pollution occurs when the rules and policies of one data model domain are mixed up with the rules and policies of another domain. Examples include:

- Polluting an application subject matter with technology subject matter. For example, building a data model domain that includes information about GPS positions along with information about how this is transmitted using DDS. This is the kind of pollution that is avoided by separating the PIM from the PSM by means of a model translator;
- Mixing two application subject matters. For example, building a "Routes" domain in which a route knows how to render itself graphically and/or textually on an HMI. This kind of pollution is avoided by separating those two subject matters into two different data model domains.

A consequence of domain partitioning is that the different aspects of one "real world" entity are represented as separate classes in different domains.

3.5. Platform Independent Models

The key premise behind MDA is that if an architecture is to be applicable to many deployed systems, and remain valid for several decades, then it must be specified in a platform-independent way, using a Platform Independent Model (PIM). "Platform" in this case refers to:

- **Middleware**, such as DDS in the case of NGVA.

- **Bus Technology**, such as the Controller Area Network (CAN) Bus or Universal Serial Bus (USB);
- **Message Definition Language,** such as IDL in the case of DDS deployments;
- **Programming Language**, such as Java or C++;
- **Database Technology**, such as ObjectStore or Oracle;
- **Hardware Architecture**, such as single node, multi-node, centralized or distributed;
- **Software Architecture**, such as single process, multi-process, cooperative scheduling or pre-emptive scheduling.

This means that the NGVA DM:

- **is simpler** than its platform specific equivalent, and therefore easier to understand and cheaper to build and maintain;
- **is unaffected by changes** in any of the above technologies, and therefore has greater longevity and less volatility than any equivalent PSM;
- **is reusable** across a range of deployment platforms, for which suitable PIM translators are available.

Platform independence in the NGVA DM domain models is achieved using the same principles used by high level languages such as Java and Ada – use of abstractions that suppress the implementation details of the underlying platform. In the NGVA DM, UML is used to provide abstractions such as classes, operations, and state machines.

3.6. MDA Process and NGVA Artefacts

The MDA process and UML notation have been adopted for the NGVA Data Model. The process is summarized in Figure 3.

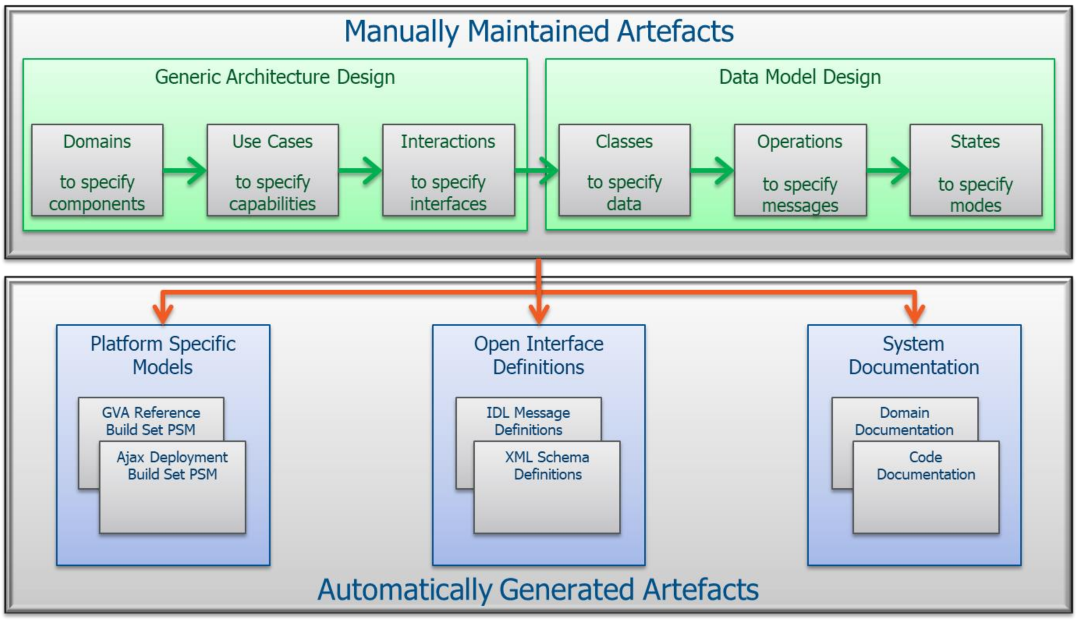

Manually Maintained Artefacts
Generic Architecture Design
Domains to specify components
Use Cases to specify capabilities
Interactions to specify interfaces
Data Model Design
Classes to specify data
Operations to specify messages
States to specify modes
Platform Specific Models
GVA Reference Build Set PSM
Ajax Deployment Build Set PSM
Open Interface Definitions
IDL Message Definitions
XML Schema Definitions
System Documentation
Domain Documentation
Code Documentation
Automatically Generated Artefacts

Figure 3: The NGVA Process and Artefacts

The MDA Process used for developing the NGVA DM involves constructing a set of models that together define the set of capabilities to be realized, and the set of components needed to realize those capabilities.
The principle artefacts that make up the NGVA DM are:

- **Data Model Domains** that represent a subject matter of the system;
- **Use Cases** that specify the required capabilities for each domain;
- **Interactions** that define how the NGVA components collaborate via a sequence of messages to realize the various use case scenarios;
- **Classes** that define the status data required to support those capabilities;
- **Operations** that specify the set of control messages required to realize those capabilities;
- **States** that specify the modes to be supported.

The essential concepts and notions for each of these are summarized below.

3.6.1. Use Case Model

The NGVA DM contains a number of use case models, each associated with a specific data model domain. While in most MDA based processes, use cases will be specified at the system level, without prejudice to any domain partitioning decisions, in the case of the NGVA DM, use cases are employed primarily to provide a high-level view of the capabilities provided by each data model domain.

Figure 4 shows an example use case model, depicted as a UML Use Case Diagram, in which the various components of a use case model are shown:

- **Actors**, representing external entities – typically human roles, items of hardware, or external systems;
- **Use Cases**, representing capabilities to be supported – typically to meet the needs of one or more human role actors;
- **Links and Dependencies**, showing the relationships between the actors and use cases.

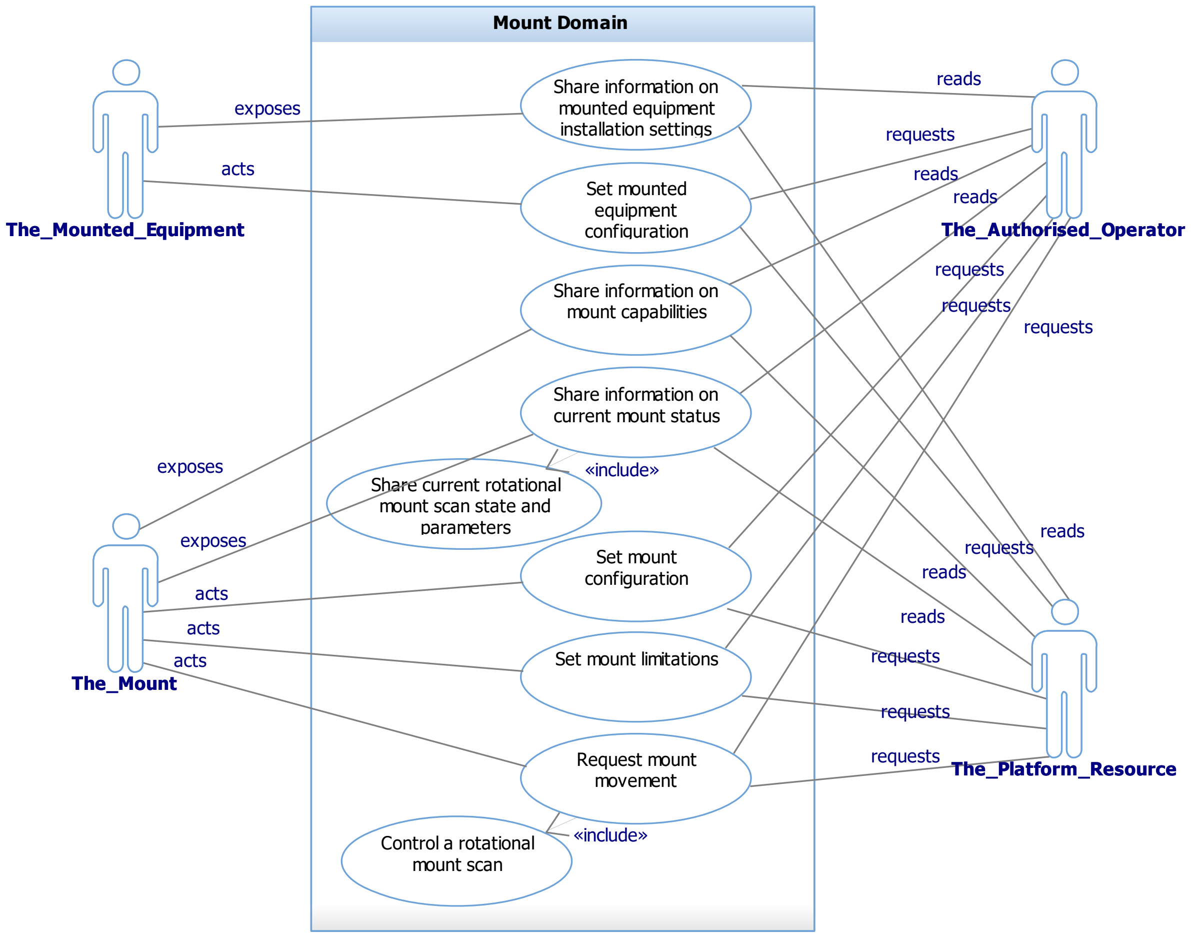


Figure 4: Domain Level Use Case Diagram Example: Mount

3.6.2. Domain Model

For the NGVA reference architecture and each deployed NGVA system, there is a Domain Model that depicts the included data model domains, along with the essential dependencies between them. The Domain Model is represented using a UML Package Diagram, as shown in Figure 5.

Each data model domain (shown as a UML package) on the Domain Model represents a distinct subject matter within the NGVA DM, whilst the broken directed arcs represent dependencies between the data model domains. A dependency indicates that the capabilities of the client (source) domain require the presence of the server (destination) domain. Each data model domain will be specified in the form of a PIM, the cornerstone of which is a UML Class Diagram.

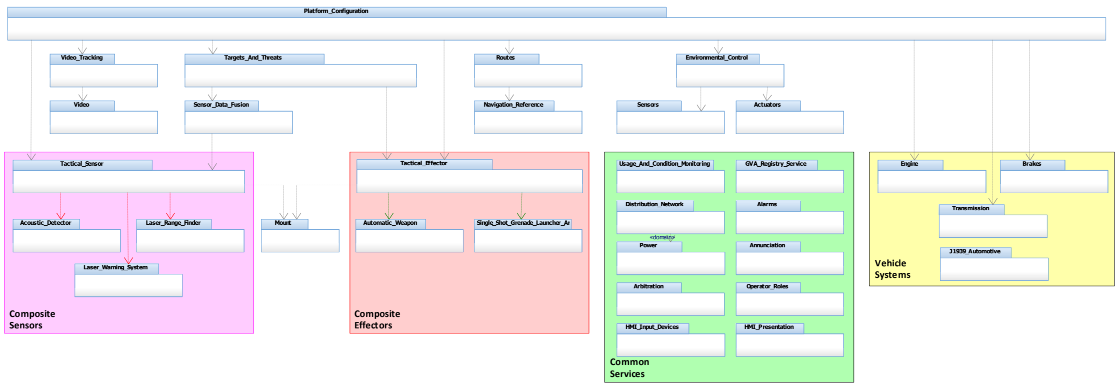


Figure 5: Domain Model Example

The shaded rectangles on this Domain Model are annotations only – they are included to show related groups of domains, e.g. pertaining to vehicle systems and common services. Not all dependencies are shown graphically to preserve the readability of the diagram.

In the NGVA DM, data model domains can represent:

- **Vehicle Platform Services**, such as "Arbitration" and "Alarms". The purpose of such domains is to define the concepts and messages to be realized by any compliant vehicle platform offering that service. Note that the implementation of the service is left to the discretion of the platform integrator, and is not specified in the model. However, users of these services must respect specific protocols to ensure that the service works at the system level. For that reason, these domains contain a number of sequence diagrams that define the protocols in terms of a sequence of messages;
- **Capabilities**, such as "Sensor_Data_Fusion";
- **Hardware Devices**, such as "Sensors", "Actuators", and "Navigation_Reference";
- **Generic Components,** providing the baseline for data structures and messages used by specific components, to ensure standardization of core data and messages across the architecture. For example, the generic "Tactical_Sensor" domain specifies the minimum set of classes and relationships for all sensors. This minimal dataset is specialized and extended by specific tactical sensor domains such as "Acoustic_Gunshot_Detection_ System" and "Laser_Range_Finder";
- **Automotive Sub-Systems**, such as "Engine", "Brakes" and "Transmission".

3.6.3. Class Models

Within each data model domain a Class Model, represented as UML Class Diagrams, describes the classes that embed the data and operations required to support the capabilities of that domain. The Class Model is a structured declaration, a static view of the system, and describes the classes and the relationships between them. The class acts as a template for all of its instances (objects) which must all have the same characteristics (attributes).

An example Class Model is shown in Figure 6. The components are:

- **Classes**, specifying the attributes and operations associated with each entity in the domain;
- **Associations and generalizations**, representing the relationships that exist between classes.

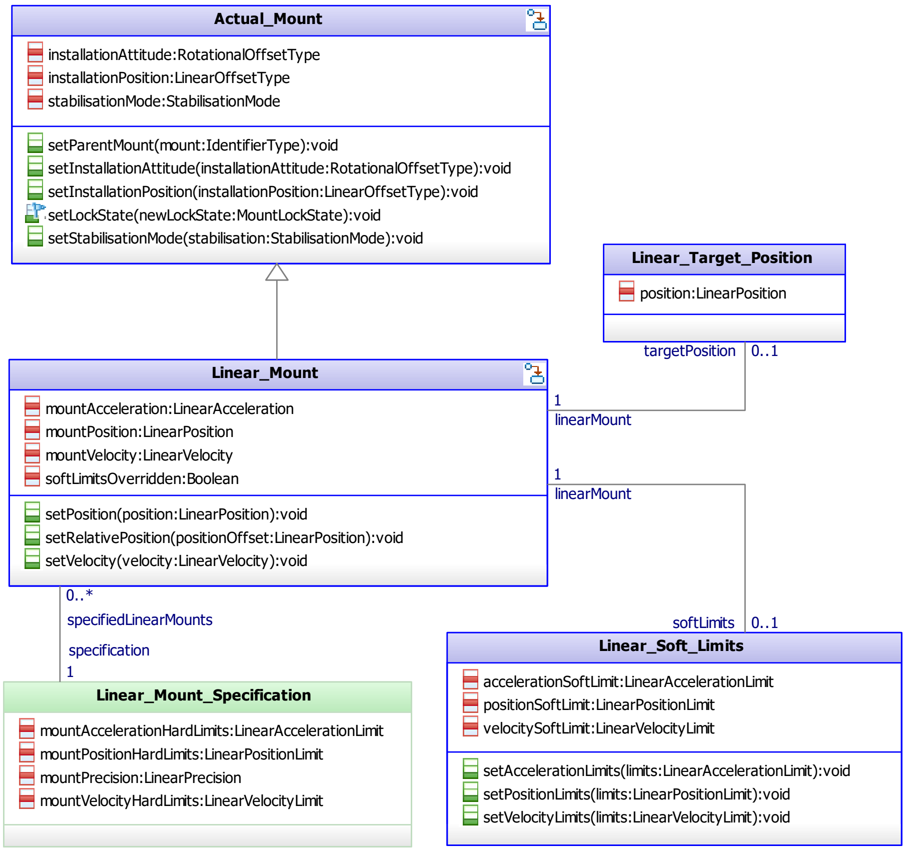

Actual_Mount
installationAttitude:RotationalOffsetType
installationPosition:LinearOffsetType
stabilisationMode:StabilisationMode
setParentMount(mount:IdentifierType):void
setInstallationAttitude(installationAttitude:RotationalOffsetType):void
setInstallationPosition(installationPosition:LinearOffsetType):void
setLockState(newLockState:MountLockState):void
setStabilisationMode(stabilisation:StabilisationMode):void
Linear_Target_Position
position:LinearPosition
targetPosition
0..1
1
linearMount
Linear_Mount
mountAcceleration:LinearAcceleration
mountPosition:LinearPosition
mountVelocity:LinearVelocity
softLimitsOverridden:Boolean
setPosition(position:LinearPosition):void
setRelativePosition(positionOffset:LinearPosition):void
setVelocity(velocity:LinearVelocity):void
1
linearMount
0..*
specifiedLinearMounts
specification
1
softLimits
0..1
Linear_Mount_Specification
mountAccelerationHardLimits:LinearAccelerationLimit
mountPositionHardLimits:LinearPositionLimit
mountPrecision:LinearPrecision
mountVelocityHardLimits:LinearVelocityLimit
Linear_Soft_Limits
accelerationSoftLimit:LinearAccelerationLimit
positionSoftLimit:LinearPositionLimit
velocitySoftLimit:LinearVelocityLimit
setAccelerationLimits(limits:LinearAccelerationLimit):void
setPositionLimits(limits:LinearPositionLimit):void
setVelocityLimits(limits:LinearVelocityLimit):void

Figure 6: NGVA Class Model Example: Mount Data Model Domain Fragment

3.6.4. State Models

State Models, depicted as UML Statechart Diagrams, are used with the NGVA DM to specify the valid set of modes for items of equipment and capabilities;

Figure 7 shows an example State Model for the “Actual_Mount” class in the “Mount” data model domain. The components of this model are:

- **States**, representing modes, exhibited for a period of time, in which a defined set of policies applies, and for which a certain set of messages are valid;
- **Transitions**, each labelled with a trigger condition, indicating the circumstances in which a class moves from one state to another.

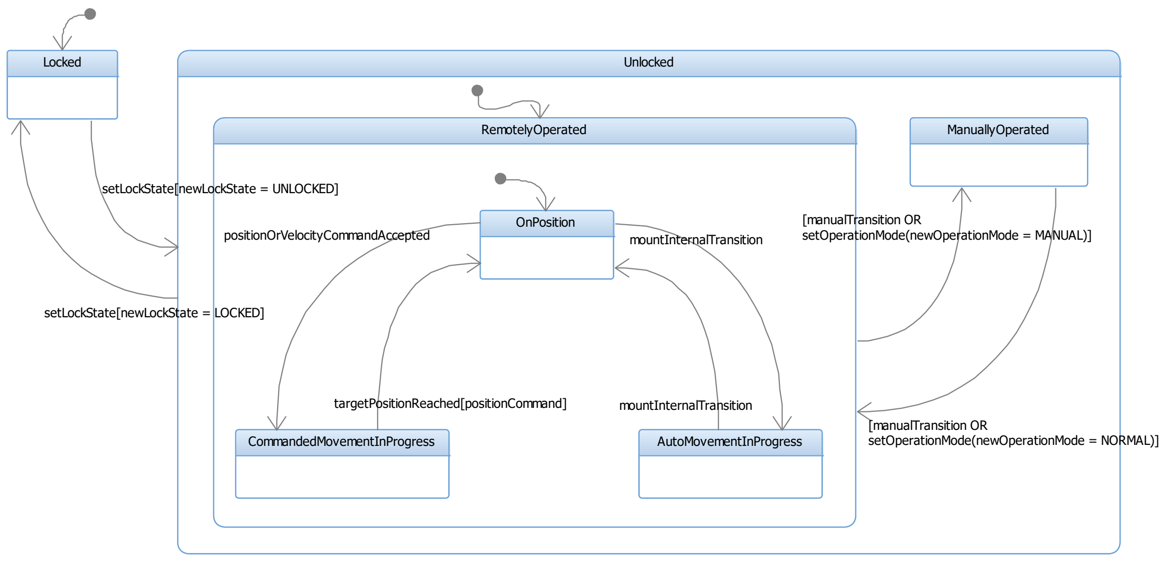
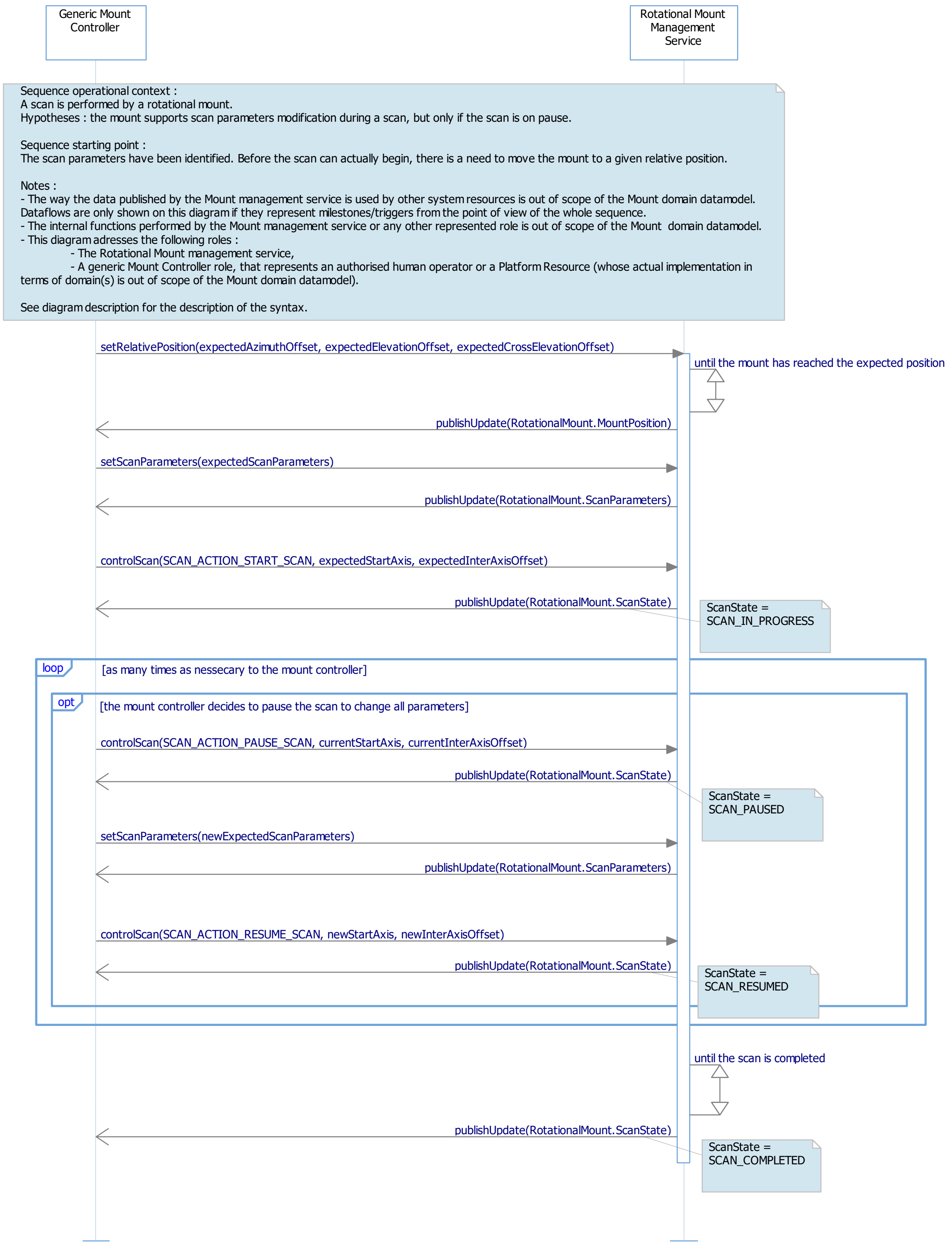

Locked
Unlocked
RemotelyOperated
ManuallyOperated
OnPosition
CommandedMovementInProgress
AutoMovementInProgress
setLockState[newLockState = UNLOCKED]
setLockState[newLockState = LOCKED]
positionOrVelocityCommandAccepted
mountInternalTransition
targetPositionReached[positionCommand]
mountInternalTransition
[manualTransition OR
setOperationMode(newOperationMode = MANUAL)]
[manualTransition OR
setOperationMode(newOperationMode = NORMAL)]

Figure 7: NGVA DM State Model Example: Actual_Mount Class

3.6.5. Interaction Models

Interaction models, depicted using UML Sequence Diagrams, are used to show the sequence of messages exchanged between components during execution of specific use case scenarios.



Sequence operational context :
A scan is performed by a rotational mount.
Hypotheses : the mount supports scan parameters modification during a scan, but only if the scan is on pause.

Sequence starting point :
The scan parameters have been identified. Before the scan can actually begin, there is a need to move the mount to a given relative position.

Notes :
- The way the data published by the Mount management service is used by other system resources is out of scope of the Mount domain datamodel. Dataflows are only shown on this diagram if they represent milestones/triggers from the point of view of the whole sequence.
- The internal functions performed by the Mount management service or any other represented role is out of scope of the Mount domain datamodel.
- This diagram adresses the following roles :
  - The Rotational Mount management service,
  - A generic Mount Controller role, that represents an authorised human operator or a Platform Resource (whose actual implementation in terms of domain(s) is out of scope of the Mount domain datamodel).

See diagram description for the description of the syntax.

Generic Mount Controller
Rotational Mount Management Service
setRelativePosition(expectedAzimuthOffset, expectedElevationOffset, expectedCrossElevationOffset)
until the mount has reached the expected position
publishUpdate(RotationalMount.MountPosition)
setScanParameters(expectedScanParameters)
publishUpdate(RotationalMount.ScanParameters)
controlScan(SCAN_ACTION_START_SCAN, expectedStartAxis, expectedInterAxisOffset)
publishUpdate(RotationalMount.ScanState)
ScanState = SCAN_IN_PROGRESS
loop
[as many times as nessecary to the mount controller]
opt
[the mount controller decides to pause the scan to change all parameters]
controlScan(SCAN_ACTION_PAUSE_SCAN, currentStartAxis, currentInterAxisOffset)
publishUpdate(RotationalMount.ScanState)
ScanState = SCAN_PAUSED
setScanParameters(newExpectedScanParameters)
publishUpdate(RotationalMount.ScanParameters)
controlScan(SCAN_ACTION_RESUME_SCAN, newStartAxis, newInterAxisOffset)
publishUpdate(RotationalMount.ScanState)
ScanState = SCAN_RESUMED
until the scan is completed
publishUpdate(RotationalMount.ScanState)
ScanState = SCAN_COMPLETED

Figure 8: Sequence Diagram Example: Mount

The components of a Sequence Diagram are:

- **Lifelines**, depicted as vertical bars, representing the NGVA actors and components;

- **Interactions**, depicted as directed arcs, representing the messages sent and received by the NGVA actors and components during execution of a use case scenario.

Time passes from top to bottom on the Sequence Diagram. Figure 8 shows an example Sequence Diagram for the “Mount” domain, showing scenarios for controlling the mount in a scan scenario.

Unique ID	Requirement Type	Requirement Text
Implementation of Specified Behavior		
NGVA_DM_002	CR	Sub-systems implementing NGVA DM modules shall adhere to the applicable UML Use Cases specification – e.g. w.r.t. pre-conditions, (alternative) flows, or post conditions needed to fulfill its requirements.
NGVA_DM_003	CR	Sub-systems implementing NGVA DM modules shall adhere to the applicable UML sequence diagrams – e.g. w.r.t. the sequence of commands and responses, loops or time-out behavior needed to fulfill its requirements.
NGVA_DM_004	CR	Sub-systems implementing NGVA DM modules shall adhere to the applicable UML Statechart diagrams – e.g. w.r.t. default states or allowed state transitions needed to fulfill its requirements.

Table **3**: Specified Behavior Implementation Requirements

3.7. Platform Specific Model

The PSM is the set of components needed to realize the PIM on the chosen deployment platform. In the case of NGVA, the PSM comprises a set of classes that define the DDS topic types and structures to be used by deployed DDS domain participants (Publisher and DataWriter, Subscriber and DataReader).

3.8. Platform Specific Implementation

The PSI is the programming language level representation of the model. In the case of NGVA, the Interface Definition Language (IDL) is used for message definition. The IDL files are generated automatically from the PSM, which was in turn generated from the PIM (cf. Figure 2). Similarly, the domain datasets used to configure certain domains could be specified using XML, and the XSD schemas for them could be automatically generated from the PIMs.

While the PIM-PSM mappings are quite complex, as they map from a technology-agnostic specification of the required data and messages to a middleware-specific (DDS in this case) architecture that addresses the needs of publishers and subscribers, the PSM-PSI mappings are typically relatively simple. Section 3.9 shows examples of automatically generated PSM and PSI components. In theory, it is not necessary to preserve a copy of the PSM, as it is used solely as an intermediate representation between the PIM and the PSI. However, the NGVA Reference Model Delivery Package (RMDP) does include the generated PSMs so that implementers and other interested stakeholders have an abstract representation of the DDS topic structure for reference when designing, reviewing and debugging the application code.

3.9. Model Translation

Using the MDA approach, the data that needs to move between vehicle sub-systems is represented as a set of PIMs that are automatically translated into PSMs and IDL. Figure 9 shows the elements of the MDA-based tool chain.

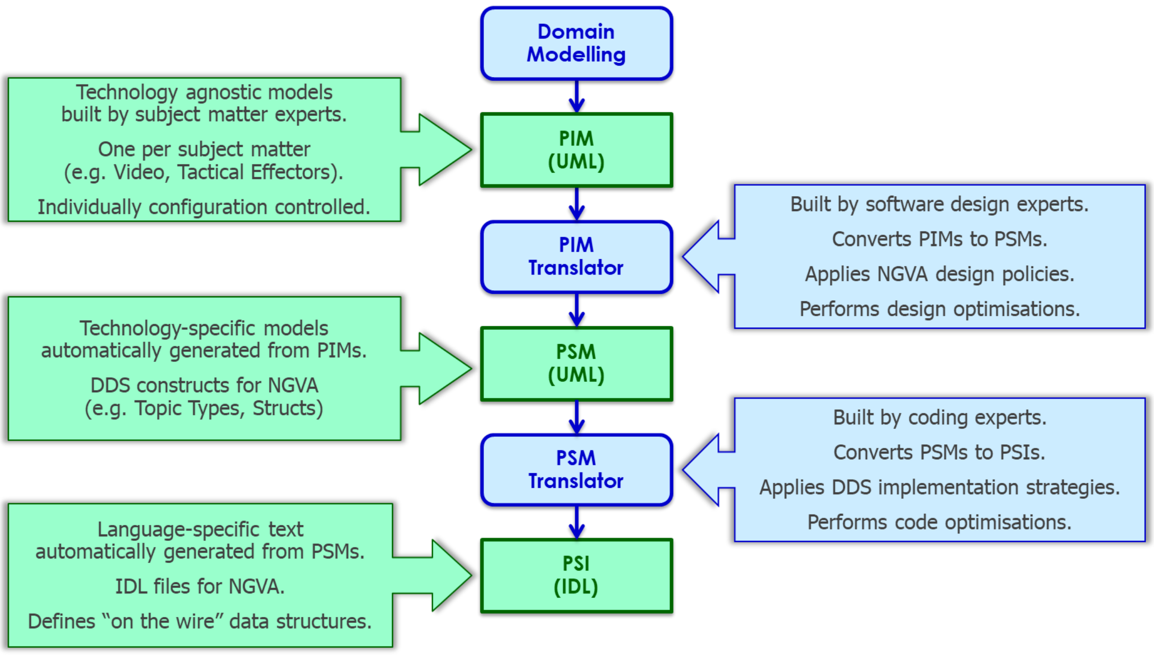

Domain Modelling
Technology agnostic models built by subject matter experts.
One per subject matter (e.g. Video, Tactical Effectors).
Individually configuration controlled.
PIM (UML)
PIM Translator
Built by software design experts.
Converts PIMs to PSMs.
Applies NGVA design policies.
Performs design optimisations.
Technology-specific models automatically generated from PIMs.
DDS constructs for NGVA (e.g. Topic Types, Structs)
PSM (UML)
PSM Translator
Built by coding experts.
Converts PSMs to PSIs.
Applies DDS implementation strategies.
Performs code optimisations.
Language-specific text automatically generated from PSMs.
IDL files for NGVA.
Defines "on the wire" data structures.
PSI (IDL)

Figure 9: MDA Toolchain used for NGVA DM

This section outlines and illustrates the mappings applied by the PIM Translator and the PSM Translator.

The PIM Translator is a “model to model” translator that takes a set of PIMs as its primary input, and generates a set of PSMs (one PSM for each PIM) as its primary output. The complete set of mappings performed by this translator are described in the translator documentation, which is also part of the RMDP, but in summary, the PIM Translator takes each PIM and generates a corresponding PSM in which:

- Each PIM class becomes a PSM class with the same name and attributes, plus key and foreign key attributes as well as metadata.
- Each PIM operation becomes a PSM class with source and recipient key attributes, metadata, and a reference number. [1]

The PSM Translator is a “model to text” translator that takes as its primary input a set of PSMs and generates a set of IDL files (one IDL file per PSM) as its primary output. An interface definition written in IDL completely defines the interface and fully specifies each operation’s parameters. An IDL interface provides the information needed to develop software applications that use the interface’s operations.

[1] This differs for NGVA DM versions 1.x. NGVA DM v1.x-based PSM classes do not have a metadata attribute, but have an explicit timeOfDataGeneration attribute instead.

The client software interfaces are not written in IDL, which is purely a descriptive language, but in languages for which mappings from IDL concepts have been defined. The mapping of an IDL concept to a software language will depend on the facilities available in the software language. The IDL files can be converted and used with a number of different software languages such as Java and C++.

3.9.1. IDL Naming Style

When translating a PSM (which was generated by the PIM Translator as described above) to IDL, the translator augments the PSM element names with some standard prefixes to signify the type of IDL element. The mappings onto from PSM elements onto IDL elements and the IDL prefixes used are shown in Table 4 below:

PSM Element	IDL Element	IDL Prefix
Domain / Build Set Module	Module	P_<moduleName>
Class	Struct	C_<className>
Attribute	Struct member	A_<attributeName>
Operation	Struct	C_<className>_<OpName>
Operation Argument	Struct member	A_<argumentName>
Data Type (Enumeration)	Enum	T_<typeName>
Enumeration Literal	Enum literal	L_<literalName>
Data Type (Structural)	Struct	T_<typeName>
Data Type (List)	Sequence	T_<typeName>
Data Type (Typedef)	Typedef	T_<typeName>

Table 4: IDL Naming Style

3.9.2. IDL Annotations

IDL annotations are automatically attached to the generated IDL elements. These are shown in Table 5 below:

PSM Element	IDL Element	IDL Prefix
Class	@mutable @appendable @final	Datatype extensibility. The value is determined by the “ExtensibilityKind” PIM property in the “GVA_Translator” group.

Key attributes	@key	Datatype member is a key field.
All attributes	@hashid	Specifies use of hashed member id (used to support use of X-Types).
Relevant attributes	@must_understand	Datatype member is required (if “MustUnderstand” property value is “true”).
Optional attributes (multiplicity [0,1])	@Optional	Datatype member is optional.

Table 5: IDL Annotations

3.9.3. IDL Generation

This section explains the general principles of IDL generation using PIM classes as an example.

Each PIM class maps to a PSM class. These PSM classes represent “Topic Types” in DDS. Figure 11 shows the PSM and IDL generated from the PIM fragment in Figure 10. Note that in the PSM:

- **The PIM associations** (“Class_A” to “Class_A_Specification”) no longer exist as UML associations. They are represented as **«**foreign**»** keys which are of type “IdentifierType” (from a standardized property). These **«**foreign**»** attributes are assigned values at runtime that match the **«**key**»** attribute value of the linked object(s).
- **The PIM composition** (to “Class_E”) no longer exists as a UML composition. It is represented as an embedded sequence of “A_componentEs” of type “Class_E” stereotyped as **«**part**»**. This means that the objects of “Class_E” will be embedded within objects of “Class_A” as a sequence of structures. [2]
- **The PIM state model** for “Class_A” no longer exists as a UML state model. It is represented as an attribute named “currentState” (from a standardized property) of type “Class_A_StateType”. Other elements of the state model are not translated into PSM elements;
- **Additional attributes** have been generated to support the use of DDS as required by the NGVA deployment policies. These are:
    - “sourceId” of type “IdentifierType” (from a standardized property), stereotyped **«**key**»** to indicate that this value must be unique across all DDS topic instances of this type;
    - “metadata” of type “MetadataType” (from a standardized property).[3]

[2] This differs for NGVA DM versions 1.x. There, composition are translated like associations.
[3] This differs for NGVA DM versions 1.x. NGVA DM v1.x-based PSM structs do not have a metadata attribute, but have explicit an explicit timeOfDataGeneration attribute instead.

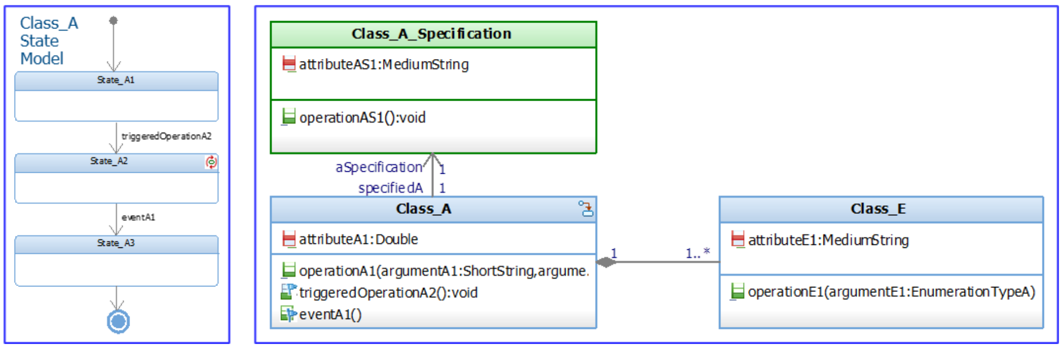
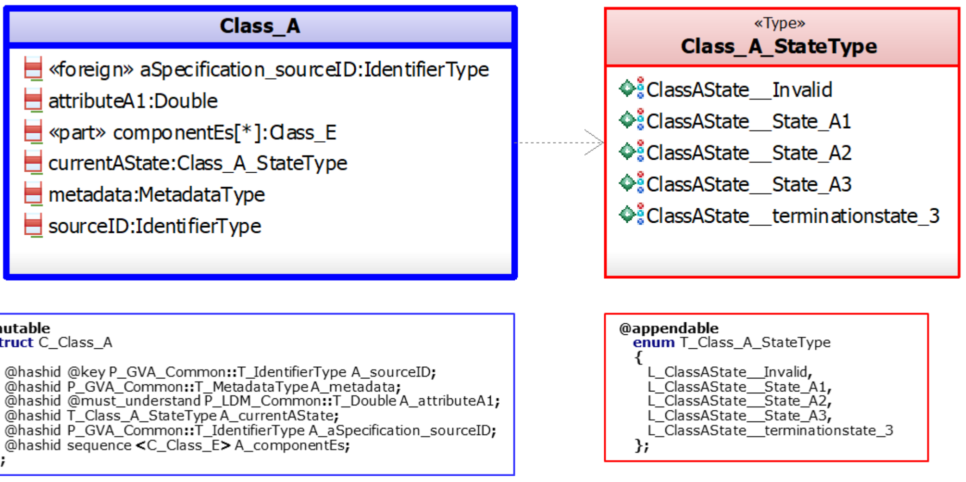


Figure 10: Example PIM Class for Translation



```
@mutable
  struct C_Class_A
  {
    @hashid @key P_GVA_Common::T_IdentifierType A_sourceID;
    @hashid P_GVA_Common::T_MetadataType A_metadata;
    @hashid @must_understand P_LDM_Common::T_Double A_attributeA1;
    @hashid T_Class_A_StateType A_currentAState;
    @hashid P_GVA_Common::T_IdentifierType A_aSpecification_sourceID;
    @hashid sequence <C_Class_E> A_componentEs;
  };
```

```
@appendable
  enum T_Class_A_StateType
  {
    L_ClassAState__Invalid,
    L_ClassAState__State_A1,
    L_ClassAState__State_A2,
    L_ClassAState__State_A3,
    L_ClassAState__terminationstate_3
  };
```

Figure 11: Generated PSM and IDL for PIM Class

Figure 11 shows that a number of annotations have been generated in the IDL:

- Each datatype member has been assigned an id in the form "@hashid", This is included to support use of DDS X-Types;
- «key» attributes are annotated with "@key";
- Each structure has been annotated with "@<extensiblityKind>". The possible values for <extensiblityKind> are "final", "appendable" or "mutable". A "final" class is one in which its attributes are fixed in name, type and position. An "appendable" class is one in which new attributes can be added to the class provided they have numerical identities greater than those already present. In other words, new attributes can be added to the ends of classes. A "mutable" class is one in which new attributes can be added to the class (either at the beginning, middle or end) and existing attributes can be deleted or transposed. The default value is "mutable".

Unique ID	Requirement Type	Requirement Text
IDL (Re-) Generation		
NGVA_DM_005	CR	In cases when IDL files have to be (re-) generated from NGVA DM modules, the NGVA translator provided in the RMDP shall be used.

Table 6: IDL Regeneration Requirements

3.10. Data Model Development and Documentation

In order to manage the development and maintenance of the Data Model, a number of software tools are employed. Further documentation in different formats is generated.

3.10.1. Rational Rhapsody

IBM Rhapsody is used to develop the NGVA Data Model. The IBM Rhapsody version needed to open a copy of the Data Model is specified in the RMDP. The IBM Rhapsody version used for the creation and modification of domains aligns with the Rhapsody version of the next RMDP.

3.10.2. NGVA Translator

Section 3.6 and subsequent sections describe the MDA approach to developing the Data Model. It is fundamental for the approach that PIM data model domains developed for the Data Model are translated into IDL files using the NGVA Data Model Translators specified for a baseline. The NGVA Data Model Translator used to generate a specific NGVA Data Model Baseline is always available at the NGVA Web Site as part of the NGVA RMDP. In addition, further artefacts such as the IDL files generated from the PIM are provided (c.f. section 4.1).

3.10.3. Rhapsody Export

For non-Rhapsody users an HTML and a Word Document export of the NGVA DM domain modules are generated and published along with the domain modules on the NGVA Home Page. Both exports have the same outline and content structure.

CHAPTER 4 DATA MODEL MAINTENANCE

This chapter describes the NGVA DM release artefacts and maintenance processes to be applied to adapt and improve the NGVA Data Model.

4.1. NGVA Reference Model Delivery Package

4.1.1. Description

The Reference Model Delivery Package (RMDP) shall provide a coherent & updated set of Specifications, Procedures, Guidelines, PIM Modules, Tools, and Artefacts to allow the Development & Integration of interoperable NGVA Resources.
The RMDP is a key product of the NGVA Configuration Management process.
The RMDP contains the required information to implement and it must be used to realize an NGVA-compliant platform or NGVA-ready mission sub-system.
It should be managed as a formal delivery of a given version of the set of NGVA domain modules, Common Module(s), and Build Set. Its contents should be considered as reference items for each NGVA-ready/-compliant system, at both platform and mission sub-system level.
The RMDP is self-contained, i.e. the need to search for additional documents/artefacts to implement the NGVA DM is limited to supporting standards, e.g. OMG Standards or NGVA STANAG AEP Volumes.

Unique ID	Requirement Type	Requirement Text
Reference Model Delivery Package		
NGVA_DM_006	CR	The RMDP shall be used to implement the data bus interface of any NGVA-compliant platform or NGVA-ready mission sub-system.
NGVA_DM_007	CR	Behavior defined in the RMDP that is relevant for the NGVA-compliant platform or NGVA-ready mission sub-system shall be implemented as defined in the RMDP.

Table 7: RMDP Requirements

4.1.2. Reference Model Delivery Package Structure

The RMDP shall be organized as depicted in Figure 12. It consists of the following directories:

1) **Release Note**, which describes the organization of the RMPD, additional elements, constraints, exceptions, and a summary of the included domain modules (see section 4.1.3 for details).
2) **Reference Data Model Rhapsody Project**, which is allocated in a dedicated directory and includes: (i) the Common Module, (ii) the set of the DM Domain PIMs, (iii) the set of DM Domain PSMs, and (iv) a Domain Model describing the relationships between DM Domain PIMs.
3) **Implementation**, which includes the set of directories containing artifacts supporting NGVA Application implementation:
    a) **Domain PSI (IDL) Directory**, which provides the IDL files of both Common and Domain Modules. The IDL files are generated according to the standard OMG syntax, and optionally for vendor specific versions: e.g. from RTI, Adlink.

   b) **Topic Names Directory**, which provides the topic names of each data model domain as *constant strings*. The set of constants shall be specified in IDL, one IDL file for each data model domain. They should be generated directly from the PIM model of each data model domain. The adopted IDL syntax shall be coherent with the one adopted for the data model domain PSI. **QoS Profile Directory**, which includes the set of standard QoS Profile specified in DDS Consolidated XML Syntax (DDS-XML) [2]. Optionally, it could also include vendor specific versions of the QoS Profiles, i.e. RTI and Adlink, which are specified as XML files.

4) **Translator Directory**, which includes:
   a) both the PIM and PSM translator adopted to generate the artifacts included in the RMDP;
   b) The Topic Name translator, which generates the Topic Names;
   c) the set of translator documentation;
   d) the translation configuration used to generated the IDL files.
5) **Specification Directory**, which includes the set of NGVA Specifications, which apply to the current version of the Reference Data Model (e.g. Arbitration Service and Registry Service Specifications).
6) **Documentation Directory**, which includes a set of supporting technical notes providing data model domain descriptions, procedure guidelines, process description, and the *Reference Model* export in HTML. Optionally, it also includes a Word and an UML/XMI export of the *Reference Data Model Project*.
7) **PIM Change Log Directory**, which includes a set of textual files, one for each data model domain explaining changes w.r.t. the previous version of the RMDP.

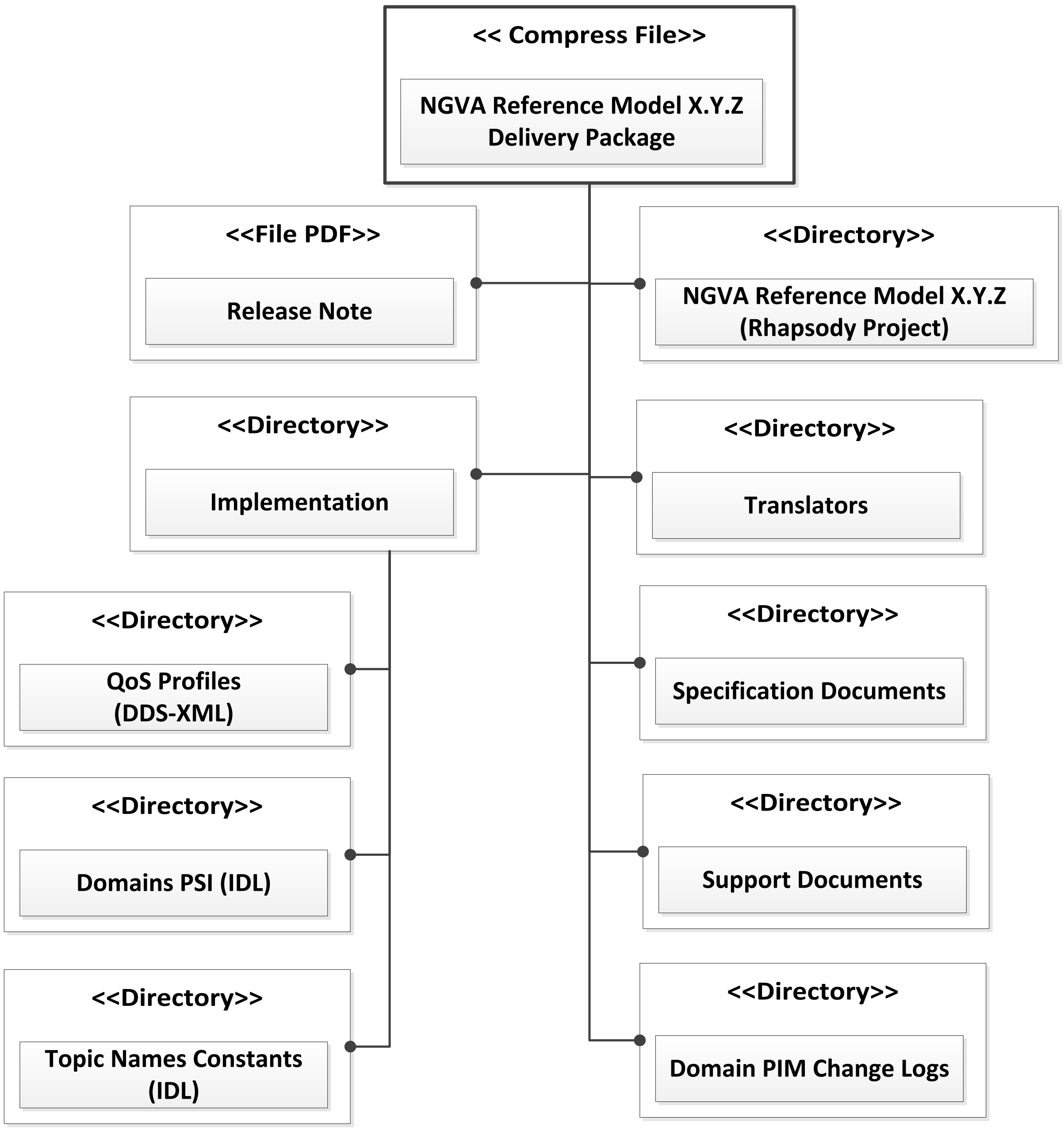


Figure 12: Reference Model Delivery Package Organization

4.1.3. Reference Model Delivery Package Release Note

The Release Note for the RMDP shall include the following sections:

1) **Release Description**, which describes the set of new features and change summary for the specific RMDP version.
2) **Delivery Package Structure**, which provides a graphical view of the RMDP structure (similar as shown in Figure 12);
3) **References**, which lists the set of reference standards documents including version numbers used in the creation of the RMDP. At least the following should be included:
   a) OMG DDS Standards, including DDS, X-Types, IDL, DDS-XML, etc.
   Optionally, the list could also include:
   b) OMG Model Driven Architecture standards;

c) OMG UML Standards.

4) **Generated IDL**, which provides for specific information on the IDLs, such as directory organization, adopted translators, exceptions.
5) **Deviations,** which describes any deviations w.r.t. Translation and X-types adoption.
6) **Specifications and support Documentation**, which contains the Specifications and support Documentation included in the Delivery Package.
7) **Table of included DM Domains**, which is a table where for each NGVA DM Domain the following is specified: (i) the DM Domain name, (ii) the subversion revision number for both the current and previous Reference Model tagged versions, (iii) the Module Maturity Level (cf. ANNEX B), (iv) the DM Domain Version, (v) Comments which summarize the new features added to and changes of the released data model domain version.

4.2. Data Model Configuration Management

4.2.1. Data Model Configuration Items

As described in CHAPTER 3, the Data Model consists of a number of data model domains where each domain is detailed in a PIM that relates to an area of concern hold in a Rhapsody Package. Each Rhapsody Package is a Configuration Item (CI) and is the lowest level at which configuration control is applied.

The Rhapsody Package containing the entire NGVA Reference Data Model is also a configuration item and configuration control is applied at this level in addition to that at the data model domain level.

4.2.2. Data Model Configuration

Consider a Data Model that consists of data model domains A, B, C, D, and E. Since each PIM is a CI, it will be assigned a specific version number. The complete Data Model, also a CI, will be assigned a specific version number that has an associated Release Note. The Release Note will specify the Data Model version and the data model domains, along with their version numbers, that form the Data Model release.

A Data Model 1 release consisting of data model domains A, B, C & D is illustrated in **Figure 13**.

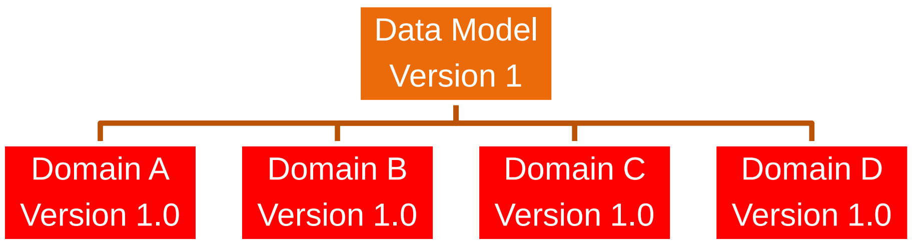


Figure 13: Data Model Version 1.0

Compare this with a Data Model Version 2 release consisting of domains A, B, C and E shown in Figure 14.

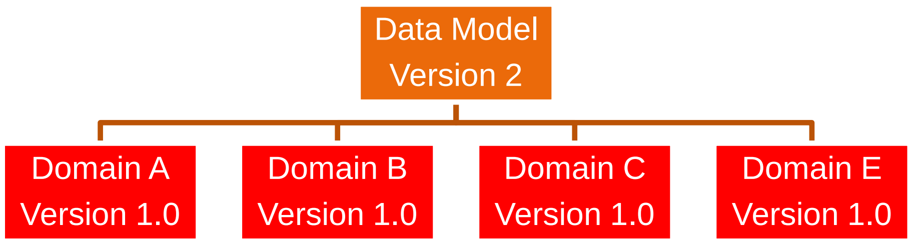


Figure 14: Data Model Version 2.0

The two Data Model releases will be given different version numbers because they are created from a different set of data model domains.

Now consider a further Data Model release that consists of domains A, B, C, and E but with data model domain A at version 1.1 as shown in Figure 15. This data model version will also be given a different version number since data model domain A is at version 1.1.

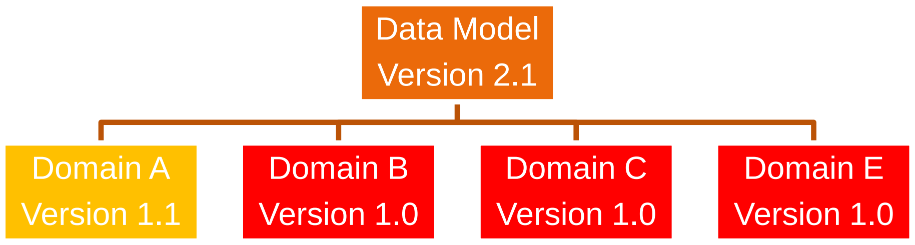


Figure 15: Data Model Version 2.1

In summary, Data Models are released as RMDP including data model domains in a specific version. The Release Notes for that RMDP will detail the data model domains (and their version numbers) that constitute that version.

4.2.3. Data Model Repository Structure

The data model resides in a subversion repository[4] and is available to authorized users. The following sections explain the structure of the subversion repository.

4.2.4. Top-Level

The Repository top level and immediate folders are shown in Figure 16. At this level, the repository contains three folders, and function of each folder is as follows:

[4] The data model is currently held in a subversion repository – https://repository.landopensystems.mod.uk/data/svndata/ngvadatamodel

1. **build_sets**
   Contains released versions of the NGVA reference data model

2. **domains**
   Contains data model domains that represent a single modelling concern. Modules in this category may be used as building blocks to construct an NGVA Reference Data Model, which is then released into the *build_sets* folder.

3. **work_in_progress**
   Working area for data model domains under development. Modules held in this folder are not under formal configuration control and are therefore subject to frequent changes. Once a data model domain's development is completed, it shall be moved to the *domains* folder and subjected to formal configuration control.

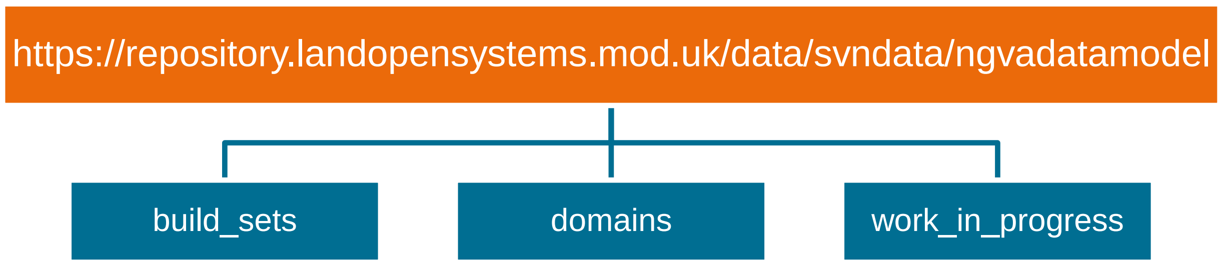

https://repository.landopensystems.mod.uk/data/svndata/ngvadatamodel
build_sets
domains
work_in_progress

Figure 16: Repository top-level structure, with root and underlying folders

4.2.5. Build_sets

The *build_sets* folder of the repository will contain a folder for each NGVA data model release, such that different data model builds are possible using the same fundamental building blocks or data model domains. Figure 17 illustrates this.

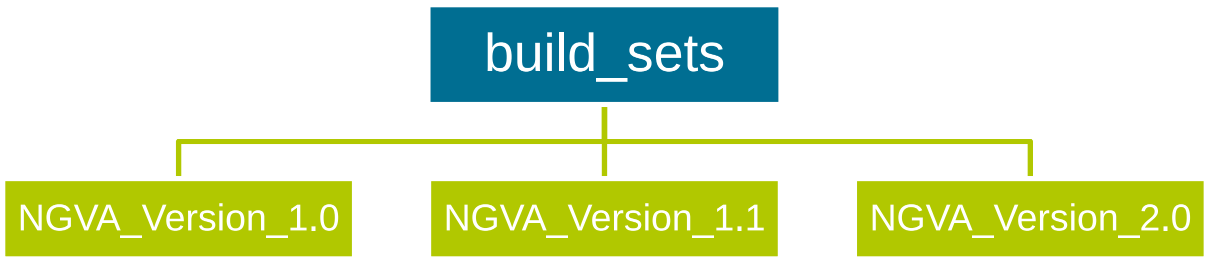


Figure 17: build_sets folder structure containing major and minor releases

4.2.6. Domains

The *domains* folder of the repository contains a folder for each data model domain, as shown in Figure 18. Data model domains are also Configuration Items, hence a further level of folders exists below each data model domain in order to facilitate the configuation control, these are:

1. **branches**

Contains a version of a data model domain that may contain differences from the version held in the *trunk*. Once a branch is regarded as complete it is usually merged into the *trunk* and a new release of that data model domain is created and placed in the *tags* folder.

2. **tags**
   Contains all released versions of the parent module.
3. **trunk**
   Contains the latest version of the data model domain. This should generally be identical to the latest released version of the data model domain held in the *tags* folder.

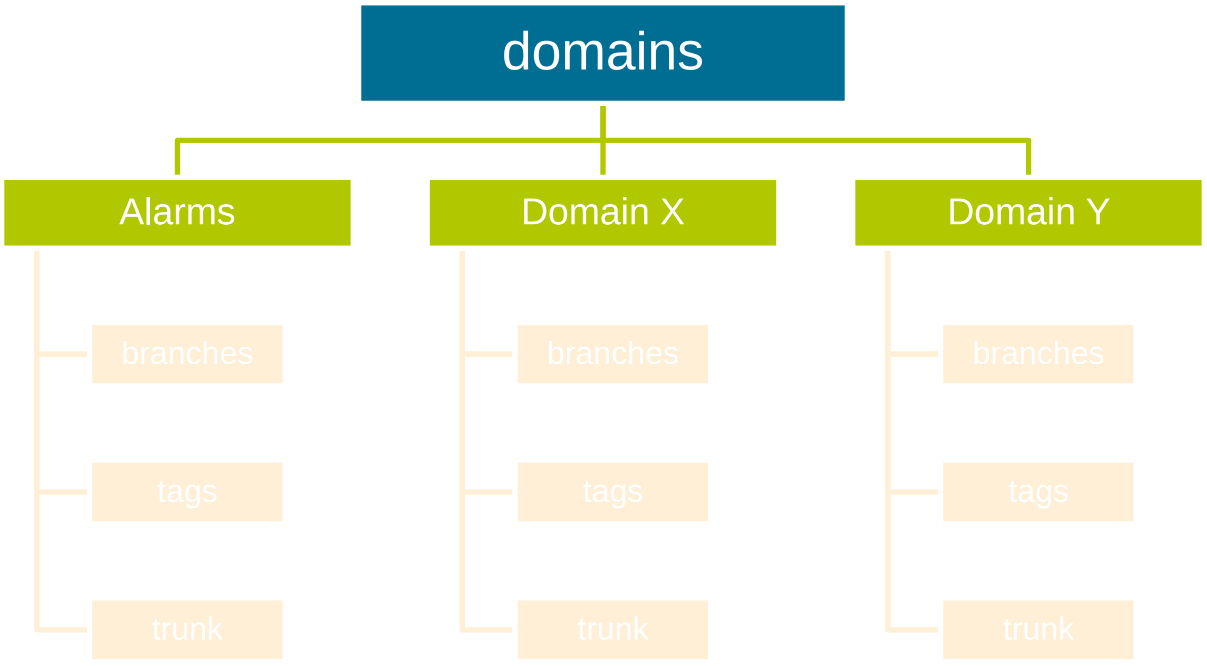


Figure 18: domain_modules folder structure, containing example modules and subfolders.

4.2.7. Work_in_progress

The *work_in_progress* folder follows the same structure as *domains.*

4.2.8. Direct Access to NGVA Reference Data Models

The current version of the NGVA RMDP is always accessible from the internet at https://www.natogva.org. Explanatory information regarding the modeling approach (Methodology and Module Development Approach, NGVA Model Maturity Guidelines, etc.) can also be found at this address.

4.3. Data Model Change Control

4.3.1. Data Model Custodian

MILVA is responsible for maintenance of the NGVA Data Model and the NGVA Data Model Change Control Board (CCB) manages configuration and authorizes changes.

4.3.2. Data Model Change Control Board

The Change Control Board (CCB) for approval/rejection of changes to the NGVA Data Model shall consist of a Chairman and government representation from NATO nations. This group will be convened at least annually by the appointed CCB Chairman. All NGVA-ratifying nations shall be invited to all CCB meetings.

4.3.2.1. Data Model CCB Chairman

The Chairman shall be appointed by the NGVA STANAG Custodian.

4.3.2.2. CCB Quorum

CCB shall be regarded as quorate provided that representatives from four or more NGVA-ratifying nations are present, not including the CCB Chairman.

4.3.3. Configuration Control

The configuration control system is required to maintain traceability of changes made to the NGVA Data Model (cf. 4.2.3). The CCB will ensure that all requested changes do not impact on the Module Maturity Level (MML) by lowering the module's MML below MML 3. The CCB will be responsible for all changes to NGVA Data Model modules. The major input to the configuration control system is the Trac ticket system associated with the repository. A new ticket may be raised by anyone that has authorized access to the NGVA Data Model Repository. The change requestor shall suggest a priority based on his needs.

Once the ticket has been raised it shall be recorded within the ticket database and be visible via the various reports available on the NGVA Data Model website[5]. The ticket will then effectively enter the ticket workflow state diagram (see Figure 19) at the 'new' state.

The CCB Chairman is the responsible Point of Contact for all NGVA Data Model tickets. The CCB Chairman will inform the CCB of ticket priorities and status.

Once the ticket has been assigned an owner, discussions and solutions can be achieved via ticket updates until a convergent solution is reached. At this point the ticket shall be considered by the Change Control Board.

Table 8 describes the ticket workflow state/transition descriptions and Table 9 describes the roles associated to each state.

[5] https://repository.landopensystems.mod.uk/ngvatrac/report

4.3.4. Change Control Workflow

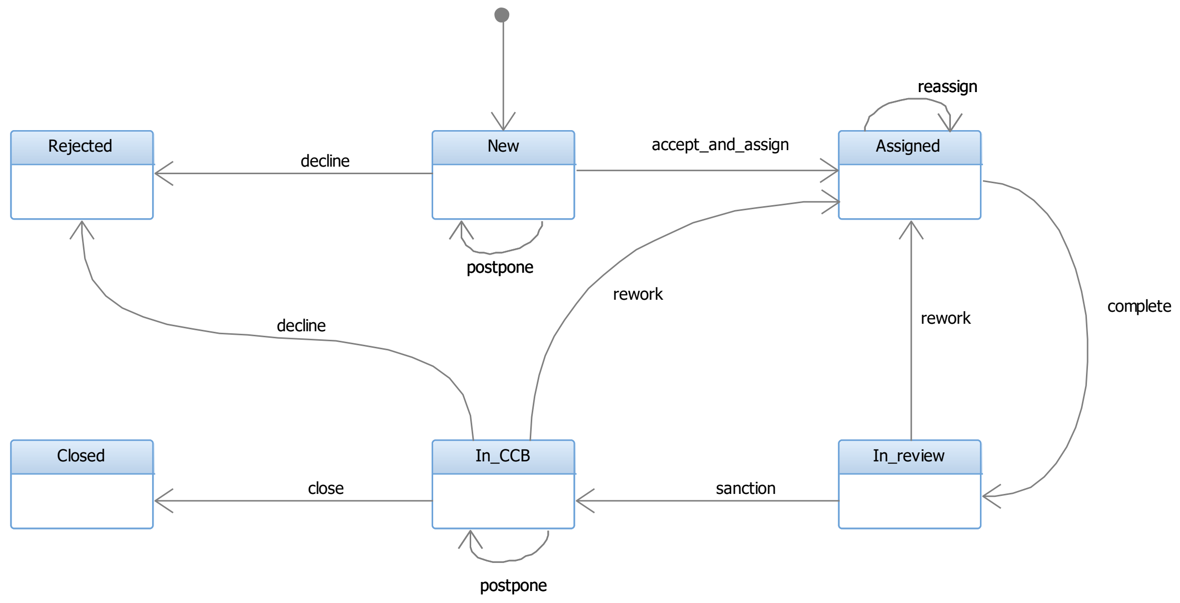


Figure 19: Ticket Workflow

State	State Description	Exit Transition	New State	Transition Description
New	A new ticket can be entered by anyone who has authorized access to the Trac database. The ticket starts in the “New” state from where it must be accepted or rejected.	accept_and_assign	Assigned	The ticket has been accepted as a valid change for which work should proceed and the ticket has been assigned to a specific user. The assignee will be the best-placed resource to begin the ticket analysis. When the ticket is assigned, additional resources to support the assignee can be defined. Moreover, when the ticket is assigned, the final reviewer must be defined.
		decline	Rejected	The ticket has been declined for a valid reason (i.e. it is a duplicate of another ticket or requests a change that is not acceptable).

State	State Description	Exit Transition	New State	Transition Description
		postpone	New	The available information does not allow the CCB to decide whether the ticket must be accepted and assigned or declined. The CCB postpones the decision about the ticket.
Rejected	The ticket has not been accepted.	N/A	N/A	No further actions will be required on this ticket. Once a ticked has been declined, the process must be finished.

State	State Description	Exit Transition	New State	Transition Description
Assigned	Someone is now responsible for processing the ticket before progressing to the In_Review state. The activities to implement the ticket are in progress. It can also be re-assigned to someone else.	reassign	Assigned	The assignee has allocated the ticket to a different user. The new assignee will be the best-placed resource to continue the activities. The reassignment decision can be done through agreement of the involved resources and without CCB involvement. The reassignment decision must be notified to the CCB.

State	State Description	Exit Transition	New State	Transition Description
		complete	In_review	The assignee has completed the required work and the change can be reviewed before submission to the CCB. The assignment to the reviewer will be done through agreement of the two parties involved without a CCB. The assignment to the reviewer must be notified to the CCB.
In_review	Someone is now responsible for checking that the ticket has been satisfactorily resolved.	sanction	In_CCB	The local review has been completed satisfactorily and the change details can be submitted to the CCB.

State	State Description	Exit Transition	New State	Transition Description
		rework	Assigned	The reviewer has assessed the proposed changes and has recommended further activities. The reassignment decision will be done through agreement of the involved resources and without CCB involvement. The reassignment decision must be notified to the CCB.
In_CCB	The ticket and proposed resolution will be examined by the CCB to confirm that the ticket can be closed.	close	Closed	The CCB has reviewed the proposed change and it has been accepted. The ticket is closed.
		rework	Assigned	The CCB has reviewed the proposed changes and has recommended further activities. In this case, the assignee and the reviewer must be defined.

State	State Description	Exit Transition	New State	Transition Description
		decline	Rejected	The CCB has reviewed the proposed change. The CCB decides that it will not be implemented. The ticket is declined.
		postpone	In_CCB	The CCB postpones the decision about the ticket.
Closed	The ticket has been resolved.	N/A	N/A	No further actions will be required on this ticket. Once a ticked has been closed, the process must be finished.

Table 8: Ticket Workflow State/Transition Descriptions

State	Roles	Role Description
New	requestor	The requestor is anyone who has authorized access to the NGVA Trac database. When a new ticket is raised, the requestor should describe the change proposal.
	CCB	CCB is responsible for accepting and assigning, postponing or declining the new ticket.
Rejected	N/A	N/A
Assigned	assignee	The assignee is responsible for processing the ticket and for implementing the required modifications. The original assignee also is responsible for notifying the CCB, e.g. in cases of reassignment.
In_review	reviewer	The reviewer is responsible for assessing the proposed changes and, when needed, for recommending further activities. He also is responsible for notifying the CCB, e.g. in cases of reassignment.
In_CCB	CCB	CCB is responsible for closing, declining, postponing, or reassigning for rework the proposed modifications.
Closed	N/A	N/A

Table 9: Roles description

4.3.5. CCB meeting

The CCB meeting should use the following process:

1. CCB meetings shall dedicate enough time to analyze new tickets and to progress existing tickets;
2. The list of the tickets to be discussed will be provided by the chairman before the meeting date;
3. During the CCB, all the new tickets must be reviewed to allow the CCB to decide, for each ticket, if it must be declined, accepted and assigned or postponed;
   a. For each ticket, the discussion shall be based upon opinion and solutions raised in the given ticket commentary. If needed, the tickets can also be presented by the requestor to facilitate discussion and decisions;
   b. When possible, consensus should support the decision to accept or decline new tickets. If no consensus can be found, the decision shall be taken according to the CCB majority that is present;
   c. When a decision cannot be taken (i.e. because a more detailed analysis is required), the ticket should be postponed to the next CCB;
   d. When a ticket is accepted, the resources for ticket assignment and review must be defined and, as far as feasible, the activities to resolve the ticket, should be defined within the CCB;
   e. If needed, during the CCB it could be possible to decide to raise further tickets to be resolved during the meeting or at next meetings;
4. During the meeting, all the completed tickets must be reviewed to allow the CCB to decide, for each ticket, if it must be closed, reworked, postponed or declined;
   a. When the ticket modifications are approved by the CCB, it can be closed and the NGVA data model can be updated;
   b. When the ticket requires more activities, it must be reassigned. In this case, the resources for ticket assignment and review must be defined;
   c. If needed, the tickets revision should be postponed to the next CCB;
   d. When the ticket modifications are not approved for valid reasons, the ticket will be declined;
   e. When possible, consensus should support the decision to close, rework or decline completed tickets. If no consensus can be found, the decision shall be taken according to the majority of the present CCB members.

CHAPTER 5 DATA MODEL IMPLEMENTATION AND DEPLOYMENT

5.1. DDS and Quality of Service

The DDS Middleware allows the application of a Quality of Service (QoS) on a DDS entity basis, which is effectively allowing the system architect to tune the performance of the system based on the data received by applications.

QoS Policies can be applied to a number of DDS entity types although not all policies can be applied to all entity types. The entity types to which QoS policies can be applied to are DomainParticipant, Topic, Publisher, Subscriber, DataWriter, and DataReader. With respect to NGVA sub-system interoperability, the QoS settings for Topic, DataWriter and DataReader are important and need to be defined. The DomainParticipant, Publisher, and Subscriber QoS are not relevant.

5.2. Quality of Service Policies

QoS policies permit applications to manage, prioritize, and shape data-flow in a network. Quality of Service policies used in NGVA QoS patterns are based on the minimum profile as specified in detail in the DDS specification [1, Annex A].

5.3. QoS Patterns

QoS Policies specify how the DDS service shall manage a given Topic Instance information flow.
A set of QoS Policies is referred to as a QoS Profile. A standardized QoS Profile is referred as QoS Pattern. A given QoS Pattern refers to a specific type of information, which typically relates to a specific data exchange scenario within the NGVA system. The following QoS Patterns have been identified:

- State Pattern
- Command Pattern
- Configuration/Specification Pattern
- Further, implementation specific QoS Patterns to distribute for example alarms, warnings, or periodically generated data.

The QoS settings for Topic, DataWriter and DataReader are specified in the DDS Quality of Service Patterns Specification, which is part of the RMDP.[6]

5.4. QoS Patterns Derivation

For deriving the QoS Pattern to be used with a specific topic, the IDL naming convention as well as the color convention in the PIM class diagram of the respective NGVA DM domain is utilized.

Specification topics represent specification classes of the PIM. Those classes are colored in green and their names end with “Specification”. Unless stated otherwise, specification topics shall be assigned the Specification QoS Pattern.

Command topics represent class operations of the PIM. Their name is formed from the camel-cased operation name joined with its class name via an underscore,

[6] The QoS Patterns to be used with NGVA DM v1.x are specified in Annex A of AEP-4754 Volume V: Data Model Edition A Version 1, February 2018.

C_Actual_Video_Source_setColourModel for example. The Command QoS Pattern shall be assigned to these topics.

State topics represent state classes of the PIM i.e. classes colored in blue. Unless stated otherwise, state topics shall be assigned the State QoS Pattern.

Further patterns e.g. for Information, Warning, Alarm, Periodic Data, and Streaming not conforming to naming conventions might be used. Their usage is dependent on the context (cf. section 5.3) and shall be documented by the system integrator.

Unique ID	Requirement Type	Requirement Text
QoS Patterns [7]		
NGVA_DM_008	CR	Specification topics (representing specification classes colored in green in the PIM and name ending with “Specification”) shall use the Specification QoS Pattern as defined in the QoS Patterns Specification from the RMDP.
NGVA_DM_009	CR	Command topics (representing class operations of the PIM) shall use the Command QoS Pattern as defined in QoS Patterns Specification from the RMDP.
NGVA_DM_010	CR	State topics (representing state classes colored in blue in the PIM) shall use the State QoS Pattern. In cases where implementation-specific patterns such as Warning or Periodic Data (as defined in the QoS Patterns Specification from the RMDP) are more suitable, these shall be justified and their use shall be documented by the system integrator.

Table 10: NGVA QoS Pattern Requirements

5.5. Topic Publishing & Subscription

This section specifies how state information of an NGVA subsystem shall be published and synchronized, which could be composed by several topics.

5.5.1. Definitions

This section defines specific terms adopted in the following of this paragraph.

1. **Aggregate**: An Aggregate is a cluster of associated classes, which is treated as a unit for the purpose of data changes. Each Aggregate is composed by a root class, which is an Entity, i.e. is uniquely identifiable in the system, and a set of part classes, which typically are Value Objects whose semantic complement the root class semantic to result in the semantic of the complete Aggregate. The root class is the only member of the Aggregate that outside classes are allowed to reference.
2. **Composite Aggregation**: A whole-part relationship, where the composite object has responsibility for the existence and storage of the composed objects
3. **Shared Aggregation**: An aggregation whose composition semantics allow for an entity to be part of multiple shared aggregations.

[7] The QoS Patterns to be used with NGVA DM v1.x are specified in Annex A of AEP-4754 Volume V: Data Model Edition A Version 1, February 2018.

4. **Value Object**: An object that represents a descriptive aspect of the domain with no conceptual identity is called a Value Object. Value Objects represent elements of the domain that care about only *what* they are, not *who* or *which* they are.
5. **Partitioned Topic**: A Topic whose PIM model is an Aggregate, i.e. root class plus a set of related part classes.
6. **Part Topic**: A Topic whose PIM model is a part class of the Aggregate, which model the related Partitioned Topic.
7. **Root Topic**: The Topic whose PIM model is the root class of the Aggregate, which model the related Partitioned Topic.
8. **Publishing Event**: The act performed by an application component to publish the Data Sample of a given Topic Instance. It is worth noting that in the case of Partitioned Topic, the Publishing Event is the composition of the Publishing Events of the topics it is composed by. Each data sample belonging to the same Publishing Event is uniquely identified by the *publishingEventID* field, which is a MetaData of the Topic Data Type.

5.5.2. PIM Domain Information Element Modelling

An Information Element (IE) provides for a self-standing semantic within the data model domain it belongs to. The PIM models any data model domain Information Element characterized by a given degree of complexity as an "Aggregate", see Definitions. The root class is an Entity, which provides for unique identification of the "aggregate" IE within the system, for this reason we refer to it as Root Entity.
The Root Entity is a concrete class, which is interconnected to the other classes, which act as parts of an aggregate IE, and so referred as Part Classes, by the following kind of relationships: specialization, composite aggregation, shared aggregation. For a specialization relationship, the Root Class is the more specialized class, see Figure 20.
Part Classes themselves can be interconnected with each other. Figure 20 depicts the Visual Band Sensor PIM as an example of an aggregate IE.
In this example the Visual_Band_Sensor class acts as Root Entity, it extends Optical_Sensor Entity, which also provides for common attributes modeled as Value Objects.
It is worth noting that in the Video Domain, the Visual_Band_Sensor identifies a specific Visual Band Sensor instance.

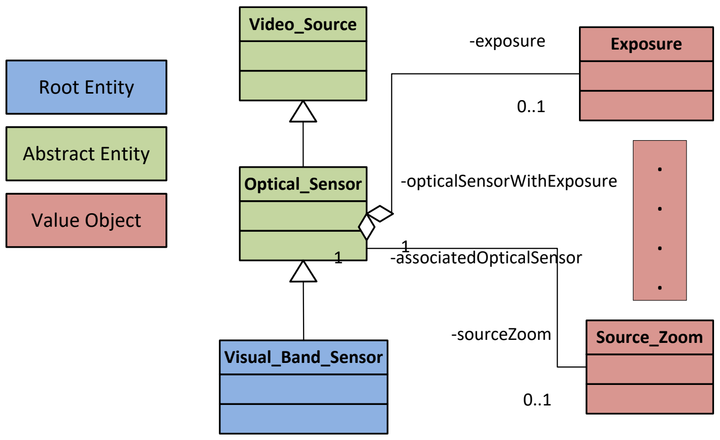


Figure 20: Visual Band Sensor Aggregate

5.5.3. Aggregate IE PIM Translation

The current version of NGVA translators, maps each class, which is part of an Aggregate IE, to a specific DDS Topic Data Type. This results in the semantic of a Topic which is modeled by an Aggregate to be fragmented in many topics each bearing a piece of information.
For sake of simplicity, we assume that the translation process is able to translate specialization associations in accordance to the IDL inheritance property.
As consequence, each chain of specializations is translated as a single topic data type.
An Aggregate IE, which models a Partitioned Topic is translated with the following set of DDS topics data types, see Figure 21:

- The **Root Topic** data type, which is the outcome of the translation of the Aggregate Root Class, and its set of inherited fields, if any.
- A set of **Part Topic** data types, as outcome of the translation of the set of Part Classes and its set of inherited fields, if any. It is worth noting that a Part Class can recursively have (sub-) parts. If this is the case, each (sub-) part will be translated as a separate Part Topic data type.

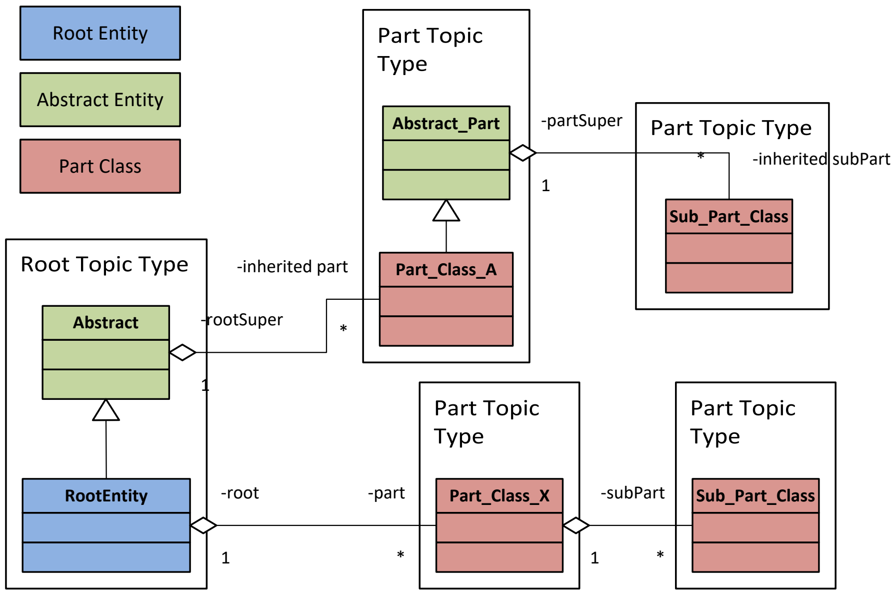


Figure 21: Partitioned PIM to DDS Translation

5.5.4. Publishing/Subscribing of a Partitioned Topic

Due to the fragmented nature of a Partitioned Topic its Publishing Event is the composition of the Publishing Events of the needed topics it is composed by, i.e. Root Topic and a set of Part Topics.
As an example of such a Publishing Event, Figure 22 below can be considered, which depicts three possible publishing events of a Partitioned Topic:

- T0: The publishing application publishes the whole set of topics composing the Partitioned Topic, i.e. Root Topic, Part Topic A, and Part Topic X.
- T1: The publishing application publishes only Root Topic and Part Topic A, i.e. the Part Topic X is optional and thus does not need to be published at every publishing event. The subscribing application does not raise any lost data sample warning because of the *A_ClassA_SourceID* field value being NIL.

- T2: The publishing application publishes all of the Partitioned Topic components again, but this time the data sample of the Part Topic A is lost. The subscribing application can detect the lost data sample due to the *A_ClassA_SourceID* value, which remains pending.

For each event the system needs to guarantee that:

- all of the objects which compose a given publishing event are received in the same acceptable time interval (**time coherence**)
- it is possible to detect lost objects, if any (**completeness**)

Moreover, to maintain a semantic coherence it is also necessary to guarantee that all of the topics composing the Partitioned Topic:

- are coherently, and univocally identifiable (**identity coherence**)
- are managed coherently by the DDS services (**service coherence**)
- are characterized by compatible QoS Patterns (**behavior coherence**)

Section 5.5.5 specifies a set of requirements, which address the above listed needs.

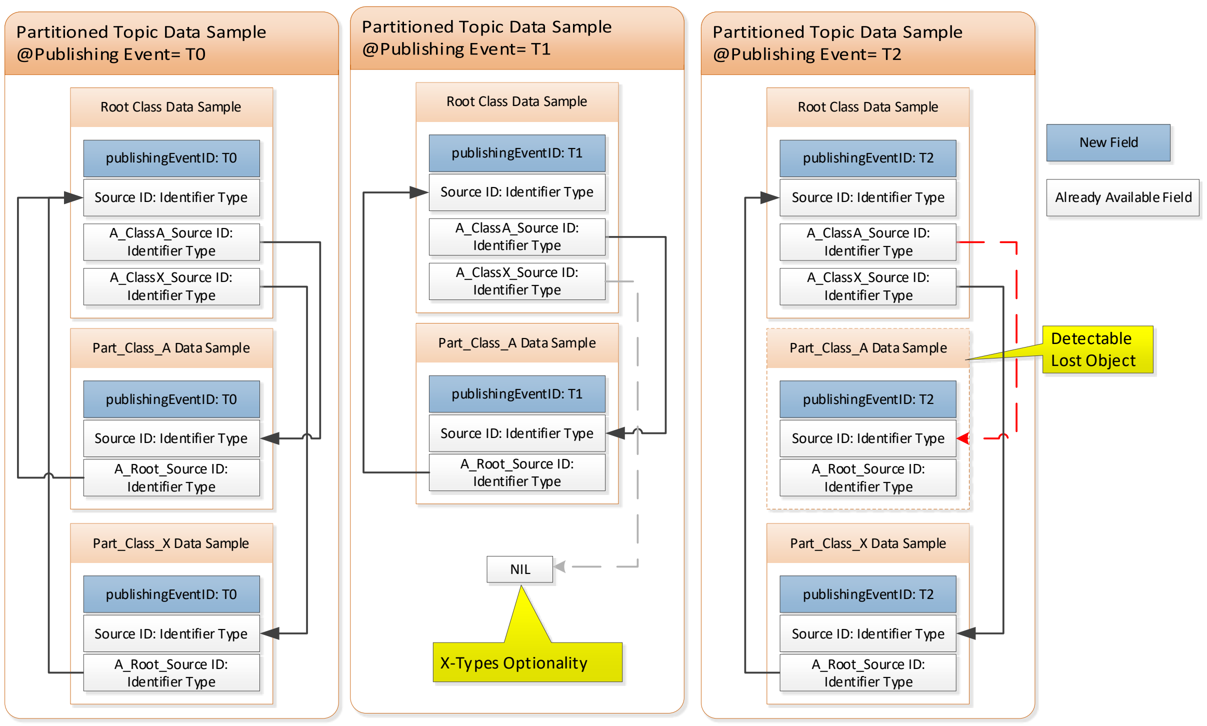

Partitioned Topic Data Sample @Publishing Event= T0
Root Class Data Sample
publishingEventID: T0
Source ID: Identifier Type
A_ClassA_Source ID: Identifier Type
A_ClassX_Source ID: Identifier Type
Part_Class_A Data Sample
A_Root_Source ID: Identifier Type
Part_Class_X Data Sample
Partitioned Topic Data Sample @Publishing Event= T1
publishingEventID: T1
NIL
X-Types Optionality
Partitioned Topic Data Sample @Publishing Event= T2
publishingEventID: T2
New Field
Already Available Field
Detectable Lost Object

Figure 22: Publishing Events of a Partitioned Topic

The NGVA PIM->PSM translation process will provide a set of basic means to allow the application to implement publishing/subscribing protocols, which satisfy the set of requirements to assure the correctness of the Publishing/Subscribing process as specified in section 5.5.5.

To each PSM topic data type, the NGVA PIM->PSM translator will add:

The *publishingEventID* attribute, a Metadata attribute which identifies a specific Publishing Event. This attribute supports the **time coherence** of a publishing event by allowing the Application to detect the set of data samples of a Root/Part Topic belonging to the same event, as depicted in Figure 22, and stated in section 5.5.5.

The *foreign key sourceID* attributes, which identify the relationships among Root/Part topic(s). This kind of attribute support the **completeness** of a publishing event, by

allowing the application to detect lost data samples, as depicted in Figure 22, and stated in section 5.5.5.
It is worth noting that these attributes are also coherent with the X-Types Optionality.
Section A.3 briefly describes a possible evolution of the current translation approach, which aims at simplifying the publishing/subscribing process.

5.5.5. Requirements for Publication / Subscription of Partitioned Topics

The fragmented nature of a Partitioned Topic imposes a set of requirements to be fulfilled by any application for its publishing/subscribing process to be correct:

Unique ID	Requirement Type	Requirement Text
Publishing of/Subscribing to Partitioned Topic		
NGVA_DM_011	CR	Both the publishing and subscribing sides of an Aggregate Topic shall be designed and implemented to be able to publish/subscribe to all of the Aggregate Topic components, i.e. the Root Topic and all of the Part Topics.
NGVA_DM_012	CR	If an NGVA subsystem implementation publishes Partitioned Topics using completeness/coherence mechanisms, the chosen approach shall be documented in the system design specification.
NGVA_DM_013	CR	If an NGVA subsystem implementation expects to receive Partitioned Topics using specific completeness/ coherence mechanisms, the expected approach shall be documented in the system design specification.
NGVA_DM_014	CR	The application shall guarantee the **time coherence** of each Publishing Event, i.e. the needed publishing events of the Root Topic and all of the Part Topics, shall be synchronized in time and identified by the same value of the "*publishingEventID*"[8] attribute.
NGVA_DM_015	CR	The application shall guarantee the **completeness** of each Publishing Event: a. The application which publishes a Partitioned Topic data sample shall allow any Subscribing peer to detect any lost data sample of the composing Root/Part Topics via the "*A_<related topic>_SourceID*" foreign key attributes, which refer the related Root/Part Topics. b. The application which subscribes a Partitioned Topic shall check for completeness the set of data samples belonging to the same Publishing Event.

[8] This differs for NGVA DM versions 1.x. NGVA DM v1.x-based IDL files do not have a metadata attribute containing publishingEventID attribute instead. Thus, this requirement is not applicable for them.

NGVA_DM_016	CR	The application shall guarantee a **coherent behavior** of each Publishing Event, i.e. the QoS Patterns adopted by the Root/Part Topic shall be coherent each other, and with the QoS Pattern specified for the kind of the Partitioned Topic, e.g. specification, or state.
NGVA_DM_017	CR	The application shall guarantee the **coherent identification** of the Root/Part Topic composing the same Partitioned Topic. Specifically: a. A Partitioned Topic shall be identified by the "*A_sourceID*" attribute of the Root Topic. b. The value of the "A_sourceID" attribute of each Part Topic should be the same as the Root Topic. c. It shall be always possible to achieve the Root Topic "A_sourceID" attribute value, by the "A_sourceID" attribute value of any Part Topic. d. If present, the "A_recipientID" attribute of the Partitioned Topic shall be the Root Topic one. e. When present, the value of the "A_recipientID" attribute of each Part Topic should be the same as the Root Topic one. f. It shall be always possible to achieve the Root Topic "A_recipientID" attribute value, by the "A_recipientID" attribute value of any Part Topic.
NGVA_DM_018	OE	The application should guarantee that the semantic of a DDS service requested for a Partitioned Topic is coherent with the same DDS service when applied to a simple Topic (**service coherence**).

Table 11: Publishing/Subscribing Process Requirements

The coherence of the following DDS Services is considered as particularly critical:

- Publishing Services: Suspend/resume publications, wait for acknowledgments, register/unregister instance, liveness management, get liveness lost status, dispose, get matched subscription;
- Subscribing Services: get sample lost status, read/query condition management, read with condition, content filtered topic, deadline missed management, get matched publications, liveness management;

Provided that an application satisfies the above listed requirements, this standard does not impose any specific state machine for the publishing/subscribing processes. Example of possible state machines for the Publishing/Subscribing are provided in ANNEX A.

Please note that the examples in this section concentrate on aggregated information elements having their origin in the same PIM module. NGVA subsystems implementing more than one PIM module, might use the same mechanism to publish/ subscribe to topics from different PIM modules which also form a single transaction.

Unique ID	Requirement Type	Requirement Text
Cross-Module Aggregated Information Distribution		
NGVA_DM_019	OE	In cases when aggregated information elements consist of topics originating from more than one PIM module, the requirements defined in Table 11 should be implemented for these topics.

Table 12: Cross-Module Aggregated Information Distribution Requirements

5.6. Core Topics and Optional Topics

The NGVA IDL translation converts PIM classes and class operations to IDL structures, which are used as topics in the implementation. However, some topics might be mandatory and some optional. Mandatory topics can derive from both the respective NGVA DM documentation and the NGVA DM implementation.

In every NGVA domain documentation, classes and class operations are defined by convention as core and optional. Core entities represent the core functionality of that domain and as such, they shall be present in the NGVA data model domain implementation. Optional ones are context-based and enhance that functionality.

Topics translated from core classes or operations are therefore seen as mandatory topics. Others count as optional topics.

Further, classes that are (transitively) associated with a cardinality of one or more to the implemented classes also become mandatory topics when translated to IDL. Specifications, for example, are always mandatory if the class they specify is implemented.

Another category of mandatory topics derives from the data model domain implementation. For instance, if a feature such as video zoom is supported, the class providing zoom information and operations relevant for changing the zoom become mandatory topics.

Defining Core Entities in the Domain Modules

To denote core classes and operations in PIM, these can be stereotyped to “core”, as shown in Figure 23.

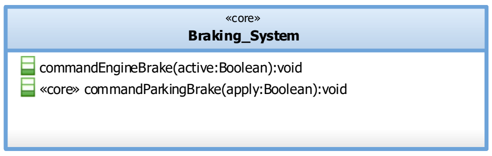

«core»
Braking_System
commandEngineBrake(active:Boolean):void
«core» commandParkingBrake(apply:Boolean):void

Figure 23 Core Entities

If core classes and operations for each domain are not defined in the Rhapsody Model, they will be defined as part of the NGVA RMDP.

Unique ID	Requirement Type	Requirement Text
Mandatory Topics		
NGVA_DM_020	CR	Topics resulting from core classes and core operations of a domain shall be implemented by mission sub-systems realizing this domain.
NGVA_DM_021	CR	Topics derived from classes that are (transitively) associated with a cardinality of one or more to classes whose topics are implemented by a mission sub-system shall be implemented as well.
NGVA_DM_022	CR	If specific functionality is declared (in specification topics) to be realized by a mission sub-system, the function-providing topics shall be implemented.
NGVA_DM_023	CR	Command topics resulting from operations in core classes are mandatory and shall be implemented by mission sub-systems realizing this domain unless the functionality is excluded in specification topics.

Table 13: Mandatory Topics Requirements

5.6.1. Example

A simplified NGVA domain class diagram is given in Figure 24 to clarify how mandatory topics are specified. For simplicity, some class attributes and operations are omitted from the diagram.

By convention, the Braking_System class is defined as a core class. The Braking_System_Specification, having a cardinality of one in the association with Braking_System, becomes a mandatory topic as well.

Further, if an optional class such as the Brake_Fluid_Reservoir is implemented, its specification Brake_Fluid_Reservoir_Specification shall be implemented.

In case the absSupported attribute in Braking_System_Specification is set to true, the Abs class as well its commandAbs operation shall be considered as mandatory when implemented as topics.

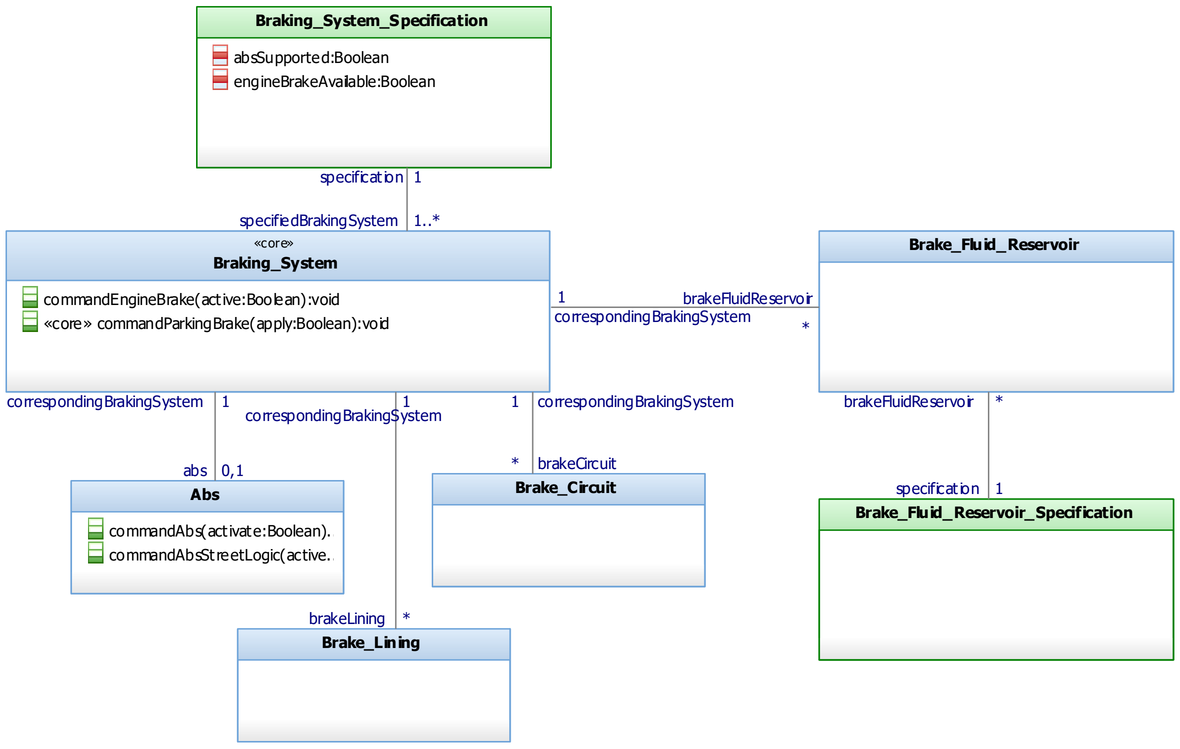

Braking_System_Specification
absSupported:Boolean
engineBrakeAvailable:Boolean
specification 1
specifiedBrakingSystem 1..*
«core»
Braking_System
commandEngineBrake(active:Boolean):void
«core» commandParkingBrake(apply:Boolean):void
1
brakeFluidReservoir
correspondingBrakingSystem
*
Brake_Fluid_Reservoir
correspondingBrakingSystem 1
correspondingBrakingSystem 1
1 correspondingBrakingSystem
brakeFluidReservoir *
abs 0,1
Abs
commandAbs(activate:Boolean).
commandAbsStreetLogic(active.
* brakeCircuit
Brake_Circuit
specification 1
Brake_Fluid_Reservoir_Specification
brakeLining *
Brake_Lining

Figure 24 Brakes Domain Module

5.7. DDS Domain Settings

The DDS Domain is a key concept in terms of interoperability since only sub-systems that belong to the same DDS domain can communicate using DDS. Besides, DDS Domains are logical scopes for Topic definitions.

A DDS domain is identified by a unique integer: the domain ID.

Unique ID	Requirement Type	Requirement Text
DDS Domain Settings		
NGVA_DM_024	CR	For each implemented DM domain, NGVA sub-systems shall use one and only one DDS domain ID for publishing and subscribing the DM domain topics on the NGVA data interface.
NGVA_DM_025	OE	The DDS domain ID used by a NGVA sub-system should be configurable at integration time with an integer between 0 and 229 (cf. [3, section 9.6.1.3]).
NGVA_DM_026	CR	If the DDS domain ID of a sub-system is not configured (either because the sub-system does not allow domain configuration or because the configuration means is not used), the default DDS domain ID shall be 0.

Table 14: DDS Domain Settings Requirements

5.8. Topic Naming

NGVA requires the use of DDS to enable the exchange of data between applications deployed into a NGVA vetronics infrastructure. DDS uses the concept of Topics to exchange data. A Topic is defined by a name, a type of data and a quality of service. For several applications to be able to exchange data, they must use Topics with the same name, a compatible data type, and a compatible quality of service. It is then necessary to standardize the names of the Topics to enable the interoperability in the NGVA context.

Topics are associated with classes defined in the PIM from the NGVA RMDP. STANAG 4754 derives the name of a Topic from the name of the class defining the Topic type. For example, a Topic is associated with the class Navigation_Resource from the package Navigation_Reference defined in the Navigation_Reference module of the NGVA PIM. The Topic name is Navigation_Reference__Navigation_Resource. A similar convention applies for command Topics. For example, the class Navigation_Resource defines the method setNorthReference associated to a command Topic. The Topic name is Navigation_Reference__Navigation_Resource__setNorthReference.

The NGVA RMDP provides an IDL file defining constants for all Topic names. If possible, an NGVA implementation should use this file to initialize the name of the Topics used in the system.

Unique ID	Requirement Type	Requirement Text
Topic Naming Requirements		
NGVA_DM_027	CR	The name of a non-command Topic shall be made of the sequence of: - The name of the package from the PIM containing the class describing the type of the Topic - The "__" token (two underscores). - The name of the class.
NGVA_DM_028	CR	The name of a command Topic shall be made of the sequence of: - The name of the package from the PIM containing the class defining the method associated to the command Topic - The “__” token (two underscores). - The name of the class defining the method - The “__” token (two underscores). - The name of the method
NGVA_DM_029	CR	A NGVA compliant implementation shall use the names contained in the IDL file in the NGVA RMDP to initialize the Topic names.

Table 15: Topic Naming Requirements

5.9. Value Initialization

Unique ID	Requirement Type	Requirement Text
Topic Attribute Requirements		
NGVA_DM_030	CR	For each DDS Topic instance, the attribute A_sourceID.A_resourceId shall take the value provided by the Registry service.
NGVA_DM_031	CR	For each DDS Topic instance, the attribute A_sourceID.A_instanceId shall take a value chosen by the publishing entity.
NGVA_DM_032	OE	For each DDS Topic instance, the attribute A_timeOfDataGeneration should use the timestamp of the original data generation.
NGVA_DM_033	CR	The attribute representing an association with multiplicity 1 shall take: - Either the value of the attribute A_sourceID from the associated DDS Topic instance. - Or the value of a T_IdentifierType with A_resourceId = 0 and A_instanceId = 0 if the system is not able to handle the relationship.
NGVA_DM_034	OE	The attribute representing an association with multiplicity 1 should always refer to an existing Topic instance.
NGVA_DM_035	CR	The sequence representing an association with multiplicity 1..* shall be initialized with: - Either the values of the A_sourceID attribute from the DDS Topic instances linked to the association. - Or the value of a T_IdentifierType with A_resourceId = 0 and A_instanceId = 0 if the system is not able to handle the relationship - Or send an empty sequence if the system is not able to handle the relationship.
NGVA_DM_036	OE	The sequence representing an association with multiplicity 1..* should contain only references to existing Topic instances and have at least one entry.
NGVA_DM_037	CR	In the case of an association with multiplicity 0..1, if the relation is null or if the system is unable to handle the association then the attribute representing the relation shall be initialized with a T_IdentifierType object with the values: - A_resourceId = 0 - A_instanceId = 0
NGVA_DM_038	CR	In the case of an association with multiplicity 0..1, if the relation is not null then the attribute representing the relation shall take the value of the A_sourceID attribute from the related DDS Topic instance.

NGVA_DM_039	CR	In the case of an association with multiplicity 0..*, if no relation exists or if the system is unable to handle the association then the sequence representing the relation shall be initialized empty.
NGVA_DM_040	CR	In the case of a relation with multiplicity 0..*, if a relation exists then the sequence representing the relation shall be initialized and populated with the values of the A_sourceID attribute from the related DDS Topic instances.
NGVA_DM_041	CR	By default, an attribute of type Boolean shall take the value FALSE.
NGVA_DM_042	CR	By default, an attribute of type Integer (either signed or unsigned) shall take the value 0.
NGVA_DM_043	CR	By default, an attribute of type Real shall take the value 0.0.
NGVA_DM_044	CR	By default, an attribute of type String shall take the value empty String.
NGVA_DM_045	CR	By default, an attribute of type character shall take the value \0.
NGVA_DM_046	CR	At initialization, an attribute of a structure representing a state from a state machine shall take the literal value representing the start state of the state machine.

Table 16: Topic Attribute Requirements

ANNEX A PARTITIONED TOPIC PUBLISHING/SUBSCRIBING STATE MACHINE EXAMPLES

This section provides an example of possible State Machines for both the Publishing and Subscribing processes of a Partitioned Topic.

A.1. PUBLISHING PROCESS FOR PARTITIONED TOPIC: A POSSIBLE SOLUTION

The publication of an aggregate IE object is a transaction, which includes the following steps:

- Write the root topic data;
- For each class part:
    - Publish part topic;

The publication of a part topic is a transaction, which includes the following steps:

- Write the part topic data;
- For each class subpart:
    - Publish subpart topic;

The publication of a subpart topic is the same as the publication of a part topic.
Each transaction shall be atomic, in the sense that it is not possible to start the publication of a new object of a given aggregate IE while the publication of a previous object of the same IE is still on-going.
It is worth noting that this section describes a top-down approach, but due to the mutual relationships between related classes, a bottom-up approach is also valid.
The data flow depicted in Figure 25 specifies the sequence of key actions.

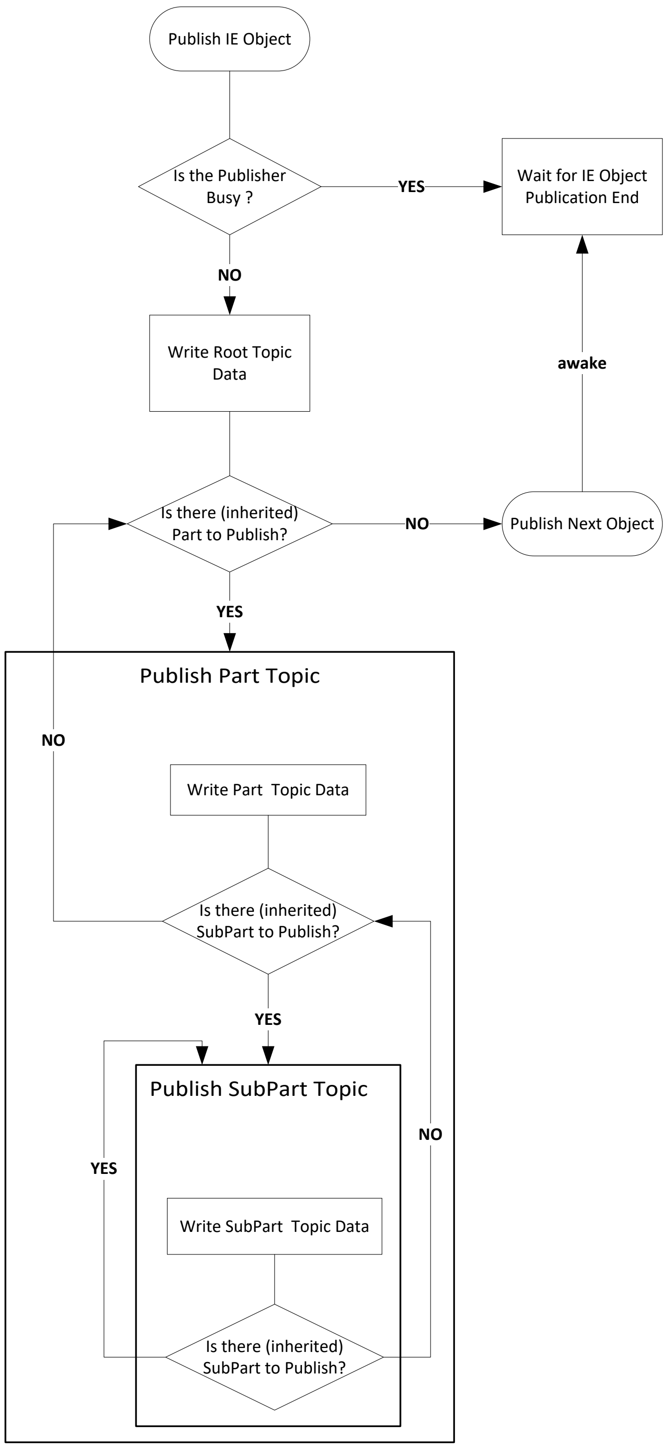


Figure 25: Publishing of Partitioned Topic

A.2. SUBSCRIPTION PROCESS FOR PARTITIONED TOPIC

The subscription of an aggregate IE object is a transaction, which includes the following steps:

- Read the root topic data;
- For each class part:
  - Subscribe part topic;

The subscription of a part topic is a transaction, which includes the following steps:

- Read the part topic data;
- For each class subpart:
    - Subscribe subpart topic;

The subscription of a subpart topic is the same as the subscription of a part topic.
Each transaction shall be atomic, in the sense that it is not possible to start the subscription of a new object of a given aggregate IE while it is still on-going the subscription of a previous object of the same IE.
It is worth noting that this section describes a top-down approach, but due to the mutual relationships between related classes, a bottom-up approach is also valid.
The data flow depicted in Figure 26 specifies the sequence of key actions.
The atomicity of publications should be assured via a mechanism similar to the DDS Coherent Set service.

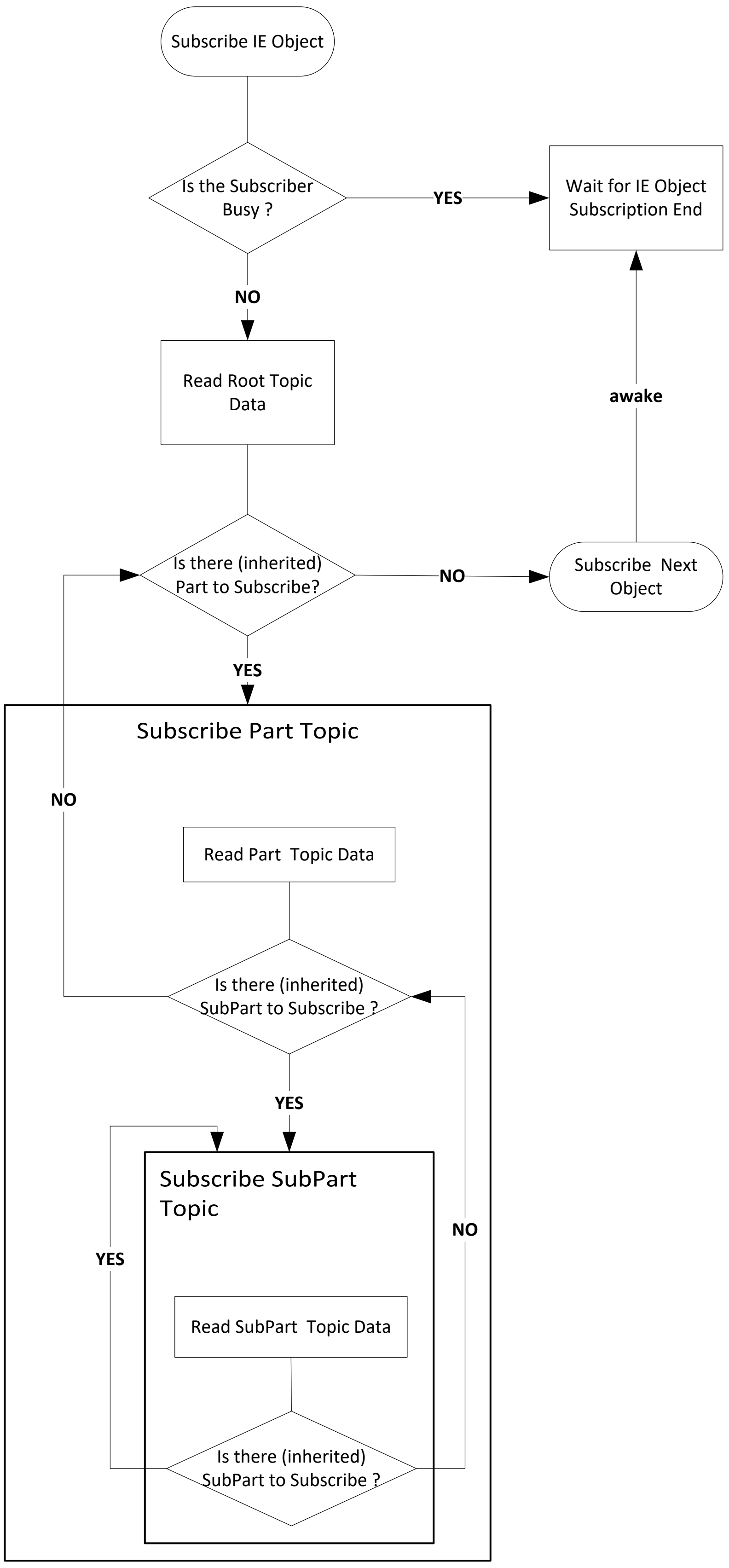


Figure 26: Subscription of Partitioned Topic

A.3. A POSSIBLE EVOLUTION OF THE CURRENT PIM MODELLING APPROACH: INTEGRATED TOPIC

Aggregate IE PIM to DDS Topic DataType Translation

The process, which translates a PIM IE into a DDS Topic Data Type, has to take into account that a DDS Topic is associated with only one Topic Data Type. Thus if we want a DDS Topic to coherently implement an Aggregate Information Element then all of the part classes composing this Information Element shall be integrated with the topic data type of the root class, the result being an Integrated Topic data type, as depicted in Figure 27.
This translation rule has the following advantages:

- It maintains the semantic correctness between PIM IE and PSM DDS Topic data type;
- It hides the organizational complexity of the aggregate IE of the PIM model to the Application, which has to deal with a single topic data type;
- Being all of the part classes integrated in the root class at translation time all of the set of requirements specified in section 5.5.5 are inherently satisfied.

Note that the adoption of a different modelling style, which aggregate each parts into the root class, get the above listed translation rules unnecessary.

Publishing and Subscribing Process for Integrated Topic

Given the above described aggregate translation process, the publication of an Integrated Topic data sample is the invocation of the correct Data Writer method, e.g. write(). Similarly, the subscription is the correct invocation of a Data Reader method, e.g. read().

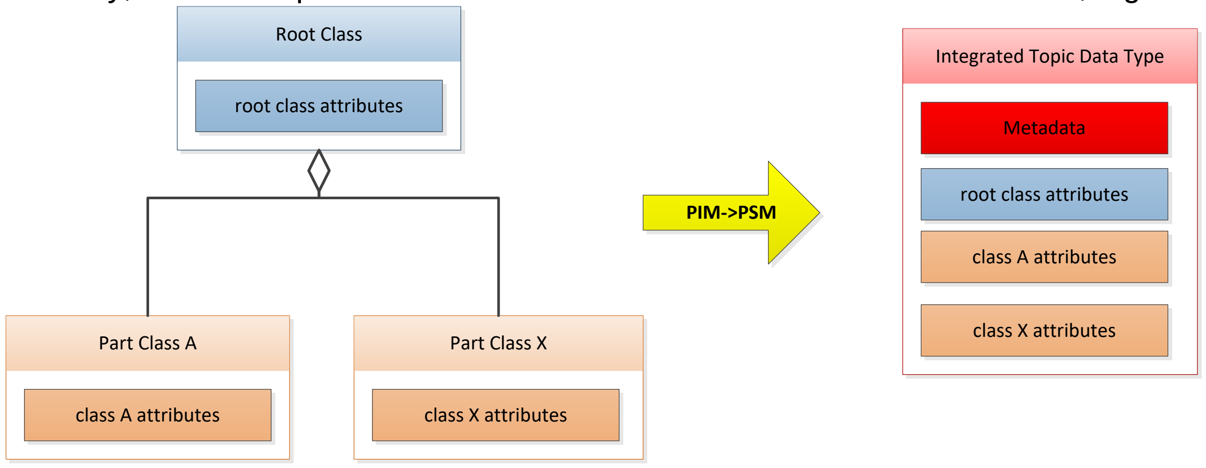


Figure 27: Translation of Aggregate IE to Integrated Topic Data Type

ANNEX B MODULE MATURITY LEVELS

Level	MML Description	Equivalent TRL Description
MML 9	Data interface model generated from module PIM has been embedded in multiple deployed designs and proven to operate.	Actual Technology system qualified through successful mission operations.
MML 8	Data interface model generated from module PIM has been embedded within an actual electronic architecture design which has passed all test and validation and is proven in-Service.	Actual technology system completed and qualified through test and demonstration.
MML 7	Data interface model generated from module PIM has been embedded within an actual electronic architecture design, and is ready for final test and demonstration.	Technology system prototype demonstration in an operational environment.
MML 6	Data interface model generated from module PIM has been embedded and implemented in a whole system context either on a systems integration rig or on an actual system using a majority of real components.	Technology system/subsystem model or prototype demonstration in a relevant environment.
MML 5	Data interface model generated from module PIM has undergone testing of the complete set of classes for that PIM on a development rig which includes the simulation of operating applications that use the Interface data structures.	Technology component and/or basic technology subsystem validation in relevant environment.
MML 4	Data interface model generated from module PIM has undergone initial lab tests by ensuring that all classes have been exercised by at least one write operation and at least one read operation, thereby demonstrating correct Data transport.	Technology component and/or basic technology subsystem validation in laboratory
MML 3	Module PIM has been subject to several reviews, agreed by an approved review body or working group and has been translated and compiled without errors.	Analytical and experimental critical function and/or characteristic proof-of concept.
MML 2	Module PIM has undergone a single review against relevant Use Cases at a stakeholder workshop session, module elements are fully documented and the module is compliant with LDM Methodology.	Technology concept and/or application formulated.

MML 1	Initial Use Cases and PIM created for Module.	Basic principles observed and reported

ANNEX C ABBREVIATIONS

CCB	Change Control Board
CI	Configuration Item
CM	Configuration Management
CR	Compulsory Requirement
DDSI	Data Distribution Service Interoperability
DDS-XML	DDS Consolidated XML Syntax
DDS	Data Distribution Service
DDSI	Data Distribution Service Interoperability
IE	Information Element
GVA	Generic Vehicle Architecture
IBM	International Business Machines
IDL	Interface Design Language
ITT	Invitation to Tender
MDA	Model Driven Architecture
MilVA	Military Vetronics Association
MOD	Ministry of Defence
NAAG	NATO Army Armaments Group
NATO	North Atlantic Treaty Organization
NGVA	NATO Generic Vehicle Architecture
NSA	NATO Standardisation Agency
OE	Optional Enhancement
OMG	Object Management Group
PIM	Platform Independent Model
PSM	Platform Specific Model
QoS	Quality of Service
RMDP	Reference Model Delivery Package
RTPS	Real Time Publish Subscribe
SS	System Specific
STANAG	Standardization Agreement
UML	Unified Modelling Language
XMI	XML Metadata Interchange
XML	Extensible Markup Language

AEP-4754 VOLV (B)(1)