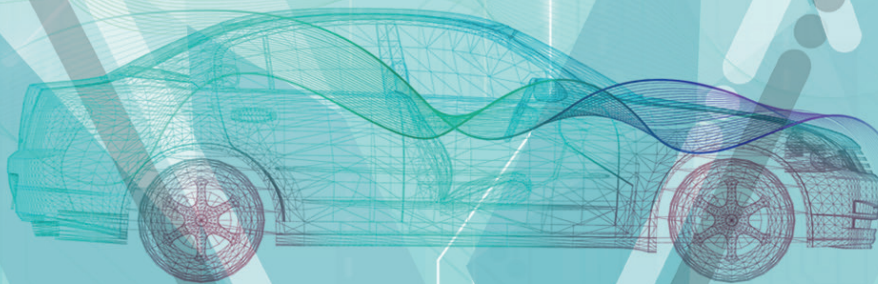


Automated Functional Data Exchange in the Automotive Industry



prostep ivip PSI 20/VDA 5550 Recommendation V1.1

Automated Functional Data Exchange in the Automotive Industry

FDX Working Group

Abstract

This prostep ivip / VDA Recommendation has been devised by the FDX Working Group. It defines a standardized format for machine-readable specifications of a data model/format for the exchange of functional data (e.g. arrays of characteristics, single characteristics, parameters) between OEMs and suppliers, in order to enable the parties to exchange highly structured data. This recommendation aims at facilitating consistent and efficient implementation of these processes in the automotive industry.

Objectives:

- Harmonization of exchange of functional data between OEMs and suppliers.
- Improved quality and availability of functional data for CAE/simulation purposes
- Elimination of discrepancies in functional data between ordered and delivered data
- Greater automation in data generation, exchange and processing

Note re. version 1.1.0

This prostep ivip / VDA Recommendation defines the FDX format. This version of the document describes the standardized rules for the development of compatible software tools. At the moment there are a number of development projects for specialized software tools that support the FDX data format in progress. After completion of a reference implementation, the required technical artefacts as well as parts 2 and 3-1 to 3-n will be added to this prostep ivip / VDA Recommendation in order to facilitate its practical implementation. In the course of the implementation of the software tool(s), it might become necessary to provide some clarifications regarding this document, which will be published as the need arises.

Disclaimer

prostep ivip / VDA Recommendations (VDA / PSI Recommendations) are recommendations that are available for general use. Anyone using these recommendations is responsible for ensuring that they are used correctly. This VDA / PSI Recommendation gives due consideration to the prevailing state-of-the-art at the time of publication. Anyone using VDA / PSI Recommendations must assume responsibility for his or her actions and acts at her/his own risk. The prostep ivip Association, the VDA and the parties involved in drawing up the VDA / PSI Recommendation assume no liability whatsoever. We request that anyone encountering an error or the possibility of an incorrect interpretation when using the VDA / PSI Recommendation should contact the prostep ivip Association (psiissues@prostep.com) immediately so that any errors can be rectified.

Copyright

- I. All rights to this VDA / PSI Recommendation, in particular the copyright rights of use, and sale such as the right to duplicate, distribute or publish the recommendation, remain exclusively with the prostep ivip Association and the VDA as well as their members.
- II. This VDA / PSI Recommendation may be duplicated and distributed unchanged, for instance for use in the context of creating software or services.
- III. It is not permitted to change or edit this VDA / PSI Recommendation.
- IV. A suitable notice indicating the copyright owner and the restrictions on use must always appear.

Table of Contents

1 Introduction	2
1.1 Background	2
1.2 Application	2
1.3 Objective	2
1.4 Disclaimer	3
1.5 The FDX Data format	3
2 About this recommendation	4
3 Fields of application	5
3.1 Use case application administrator: Customizing FA master (for component-specific additions)	7
3.2 Use case application administrator: Defining and editing FA-FDX for component	8
3.3 Use case application administrator: Instantiating and editing FA-OEM for component	9
3.4 Use case data requester: Creating, completing and transmitting order	10
3.5 Use case data supplier: Receiving, examining and accepting order	11
3.6 Use case data supplier: Entering, finalizing and submitting data	12
3.7 Use case data user: Storage of test data in ATFX format at OEM or supplier	13
3.8 Content and structure of functional data exchange file (example)	14
3.8.1 Structure and attribute categories	14
3.8.2 Measurement order	18
3.8.3 Order meta data and component-specific additional information	18
3.8.4 Dataset meta data	20
3.8.5 Unit under test	21
3.8.6 Test equipment setup	22
3.8.7 Test equipment parameters	23
3.8.8 Functional data	26
3.8.9 Derived characteristic values	27
4 Technical Basis	28
4.1 Steps from technical specification to description in ATFX exchange format	28
4.2 ASAM ODS	30
4.2.1 ASAM ODS base model	30
4.2.2 Application models	31
4.2.3 ASAM ODS ATFX data exchange format	33
4.2.4 More information on ASAM ODS and ATFX	33
4.3 openMDM	33
4.3.1 openMDM4 standard	33
4.3.2 Extensions to and deviations from the openMDM4 datamodel	33
4.3.3 More information on openMDM base standard	34
4.4 prostep ivip / VDA FDX extensions of openMDM4 and ASAM ODS	34
4.4.1 Extensions of openMDM application model	34
4.4.2 Relationship between ASAM ODS base model, openMDM application model and prostep ivip / VDA FDX application model - example of rubber mount	34
5 Structure of FDX functional data exchange file	35
5.1 Functional data exchange file as template (.fdtc)	35
5.2 Functional data exchange file as FDX file (.fdxc)	36
5.3 ATFX file (.atfx)	36
5.4 Referenced files	37
5.5 Validation certificate	37

Table of Contents

6 Minimum requirements for data exchange	38
6.1 Data security	38
6.2 Data exchange	38
7 Normative references	39
8 Appendix	40
8.1 Appendix A: Terms and definitions	40
8.2 Appendix B: openMDM application model	42
8.3 Appendix C: Extension of openMDM application with rules; including an extension that allows for the distinction between mandatory and optional attributes in the order files	43 46
8.4 Appendix D: Relationship between ASAM ODS base model, openMDM application model and prostep ivip / VDA FDX application model - example of rubber mount	51
8.5 Appendix E: Components of application elements UnitUnderTest, TestSequence and TestEquipment and their relationship with categories of the prostep ivip / VDA FDX application model	
8.6 Appendix F: Model-driven aspects of openMDM application model	57
8.7 Appendix G: Representation of model-driven aspects of openMDM4 application model in ATFX	68
8.8 Appendix H: XML schema of ASAM ATFX data exchange format	70
8.9 Appendix I: Conventions for UML diagrams and associated texts	75

Figures

Figure 2-1 Structure of prostep ivip / VDA Recommendation	5
Figure 3-1 Main roles in the use of the FDX data exchange format	5
Figure 3-2 Use case "Customizing FA master"	6
Figure 3-3 Use case "Customizing FA master"	7
Figure 3-4 Use case "Defining and editing FA-FDX for component"	8
Figure 3-5 Use case "Instantiating and editing FA-OEM for component"	9
Figure 3-6 Use case "Creating, finalizing and transmitting order"	10
Figure 3-7 Use case "Receiving, examining and accepting order"	11
Figure 3-8 Use case "Entering, finalizing and submitting data"	12
Figure 3-9 Excerpt from main category "Test equipment parameters" with various subcategories and attributes	14
Figure 3-10 Structure of applicable application model	15
Figure 3-11 Functional data exchange file: main categories, subcategories and contents	16

Figure 3-12	Examples of attributes, value ranges, block rules and simple rules	17
Figure 3-13	Main category "Measurement order" with associated attributes and sample content	18
Figure 3-14	Excerpt from main category "Order meta data" with associated attributes and sample content	19
Figure 3-15	Excerpt from main category "Component-specific additional information" with associated attributes and sample content	19
Figure 3-16	Main category "Dataset meta data" with associated attributes and sample content	20
Figure 3-17	Main category "Unit under test" with associated attributes and sample content	21
Figure 3-18	Main category "Test equipment setup" with associated attributes and sample content	22
Figure 3-19	Subcategory "Test type" with associated attributes and sample content	23
Figure 3-20	Subcategory "Measurement" with associated attributes and sample content	24
Figure 3-21	Subcategory "Test type" with associated attributes and sample content	25
Figure 3-22	Main category "Functional data" with associated attributes and sample content	26
Figure 3-23	Main category "Derived characteristic values" with associated attributes and sample content	27
Figure 4-1	Relationship of objects and representation rules	28
Figure 4-2	Relationship between ASAM ODS, openMDM and FDX (levels 1 and 2)	29
Figure 4-3	ASAM ODS base model	31
Figure 4-4	Excerpt from an application model and the associated part of the base model	32
Figure 5-1	Components of FDX exchange format	35
Figure 5-2	Structure of ATFX file	36
Figure 5-3	Data enrichment in ATFX file	37
Figure 8-1	prostep ivip / VDA FDX modifications to openMDM4 application model	44
Figure 8-2	Relationship between ASAM ODS base model, openMDM application model and prostep ivip / VDA FDX application model - example of rubber mount (overview)	47
Figure 8-3	Valid components of application element UnitUnderTest for rubber mount (example)	51
Figure 8-4	Relationships between main categories and subcategories of prostep ivip / VDA FDX data model and components of application element UnitUnderTest in prostep ivip / VDA FDX application model - example of rubber mount	52
Figure 8-5	Attributes of application element MeasurementOrder and possible values	52

Figure 8-6	Valid components of application element TestSequence for rubber mount (example)	53
Figure 8-7	Relationships between main categories and subcategories of prostep ivip / VDA FDX data model and components of application element TestSequence in openMDM application model - example of rubber mount	54
Figure 8-8	Attributes of application element MeasurementType and possible values	55
Figure 8-9	Valid components of application element TestEquipment for rubber mount (example)	55
Figure 8-10	Relationships between main categories and subcategories of prostep ivip / VDA FDX data model and components of application element TestEquipment in openMDM application model - example of rubber mount	56
Figure 8-11	Attributes of application elements DataOrigin, MeasurementSensors and Sensors and possible values	57
Figure 8-12	Model-driven aspects of openMDM application model and use of templates and catalogues in connection with the ASAM ODS base model and the prostep ivip / VDA FDX application model - example of rubber mount	59
Figure 8-13	Scenario for representation of prostep ivip / VDA FDX data model by prostep ivip / VDA FDX application model	65
Figure 8-14	Scenario of a specific openMDM application model - example of rubber mount	66
Figure 8-15	Model-driven aspects of openMDM application model	68
Figure 8-16	Excerpt from application_element section of an ATFX exchange file; example of prostep ivip / VDA FDX application model for the rubber mount, for application element UnitUnderTest	71
Figure 8-17	Excerpt from application_element section of an ATFX exchange file; example of prostep ivip / VDA FDX application model for rubber the mount, for application element MeasurementOrder	72
Figure 8-18	Excerpt from instance_data section of an ATFX exchange file; example of prostep ivip / VDA FDX application model for the rubber mount, for application element UnitUnderTest	73
Figure 8-19	Excerpt from instance_data section of an ATFX exchange file; example of prostep ivip / VDA FDX application model for the rubber mount, for application element MeasurementOrder	73
Figure 8-20	Excerpt from instance_data section of an ATFX exchange file; example of prostep ivip / VDA FDX application model for the rubber mount, for application element UnitUnderTest	74
Figure 8-21	Excerpt from instance_data section of an ATFX exchange file; example of prostep ivip / VDA FDX application model for the rubber mount, for application element MeasurementOrder	74

Tables

Table 2-1	Relevance of chapters for different reader groups	4
Table 3-1	Use case application administrator: "Creating and editing FA master"	7
Table 3-2	Use case "Defining and editing FA-FDX for component"	8
Table 3-3	Use case "Instantiating and editing FA-OEM"	10
Table 3-4	Use case data requester: Creating, finalizing and transmitting order	11
Table 3-5	Use case data supplier: Receiving, examing and accepting order	12
Table 3-6	Use case data supplier: Entering, finalizing and submitting data	13
Table 3-7	Use case data user: Storage of testing data in ATFX format at OEM or supplier	13
Table 8-1	Application elements Sensors, MeaQuantity, Quantity, Unit and PhysDimension for static force-displacement characteristic curve (CurveForceDisplacementStatic) measuring program for rubber mount	50
Table 8-2	Testing scenario with templates and catalogues for a measurement order - example of rubber mount	60

1 Introduction

1.1 Background

Reduced availability of hardware trial platforms, shorter development times and the increased complexity of products require operators to align their digital vehicle development processes with generally recognized system engineering standards. For the design of modern cars, various design, construction and simulation plants and teams work together across all stages of the project. They all require accurate information, including comprehensive and consistent functional component descriptions and specifications in the form of functional data.

Functional data includes measured, calculated or estimated scalable values, characteristics and array of characteristics that describe the properties of components and are essential for the development of a product.

For the digital transformation in product development, this information must be available in a standardized, electronic format. Automated data processing and model parameterization for the efficient definition of simulation models demand that both the content and the format of the data are standardized.

However, today data is still often exchanged in an unstructured manner and many different formats. Unless there is a standardized data exchange format in place, further automation cannot progress, so that achieving greater compatibility, higher process safety and improved quality is hampered.

1.2 Application

This prostep ivip / VDA Recommendation describes a data model and format for the standardized and fully traceable exchange of functional data and associated relevant meta data. It also outlines the applications that benefit from such standardization, for instance with regard to the exchange of functional data between car manufacturers and their suppliers, and between automotive suppliers and their subcontractors.

The focus thereby remains always on the efficient use of the relevant functional data for the virtual development process.

1.3 Objective

The new standard aims at streamlining the exchange and processing of functional data, by compiling all relevant information in a machine-readable format.

With this format, the data requester will be able to describe the requirements regarding the functional data, including division into mandatory and optional data.

The data supplier on the other hand will be able to fully understand these requirements, so that the relevant testing/simulation data can be generated. The format instructs the data supplier as regards mandatory and optional data and requirements for submission.

For quality assurance purposes, the format and the data structure incorporate simple, software-based checks for the identification of missing or incorrect entries. The results of these checks are recorded in the form of validation certificates. The end result is a complete dataset containing all requirements, associated results and validation certificates.

Descriptive attributes facilitate the transfer of the data to data management systems such as PLM, PDM or TDM. In addition, the format supports automated data reading and writing.

The format must be suitable for use in the global market, which means in particular that it must support different units of measurement.

The chosen format must be scalable and updatable to cater for future data volumes. OEMs and the suppliers are thus in a position to add specific blocks of information as required.

Changes and additions to the data model are detected and processed automatically without having to reconfigure the software tools.

The format allows for the file-based exchange of data between IT tools, and thus for greater automation, irrespective of the IT technology and IT platform used by the parties.

This prostep ivip / VDA Recommendation has been drawn up to explain the data model and the data format, as a basis for the development of bespoke applications.

1.4 Disclaimer

This prostep ivip / VDA Recommendation does not aim at standardizing the OEM-side requirements regarding the content of functional data from suppliers, as this would infringe too much on corporate secrets and go against the wish of OEMs to devise specific product features and to follow their own development strategies and approaches.

Given the examined fields of application and the expected data volumes, the storage of functional data in binary format is not envisaged, although the ASAM ATFX format supports the exchange of binary instance data.

1.5 The FDX Data format

The starting point for this recommendation is the ODS (Open Data Services) standard developed by ASAM e.V. (Association for Standardization of Automation and Measuring Systems), known as the ASAM ODS standard, the related ASAM ATFX data exchange format (XML) and the openMDM application model also based on the ASAM ODS standard.

The data format described in this document builds on this standard, and extends it in three areas:

1. Extension of current openMDM4 application model
2. Fleshing out of openMDM4 application model with application models for existing, concrete applications
3. Bundling of data in a container file that contains not only the XML (ATFX) file, but also referenced files and an optional validation certificate file.

We also provide a template of the recommended basic scope of exchangeable functional data at component level. Such component-specific templates for functional data cater for all information required for the standardized specifications for measurement orders as well as the exchange of data.

Links to Normative references and Glossary:

- [Normative references](#)
- [Appendix A: Terms and definitions](#)

2 About this recommendation

This document has been structured so that readers of various target groups and with different levels of technical expertise can find the information that is relevant to them.

It targets in particular the following three reader groups:

- Specialist users and decision-makers (component managers, system engineers, testing engineers, etc.)
- Technical managers (application administrator for templates, process optimization engineers, IT system architects, etc.)
- Software developers (tool designers, interface developers, system integrators, etc.)

Chapter	Specialist user	Technical managers	Software developers
1 Introduction (purpose and objectives)	X	X	X
2 About this recommendation / prostep ivip / VDA recommendation	X	X	X
3 Fields of application	X	X	X
4 Technical basis (ASAM ODS, openMDM)		X	X
5 Structure of FDX functional data exchange file	X	X	X
6 Minimum requirements for data exchange	X	X	X
7 Normative references		X	X
8 Appendix			X
Part 2	X	X	X
Part 3-n	X		

Table 2-1: Relevance of chapters for different reader groups

This prostep ivip / VDA Recommendation is structured as shown in Figure 2 1. It consists of a main document (part 1 of the recommendation) covering the topic in general, a second part with more detailed information regarding the attribute list and the data model, and parts 3-n with component-specific information provided in the form of pdf files and templates known as ATFX files that contain the component-specific application administration. The document comes with a number of appendixes providing more detailed information. For a specific component, readers are advised to read part 1, part 2 and the component-specific section in part 3-n.

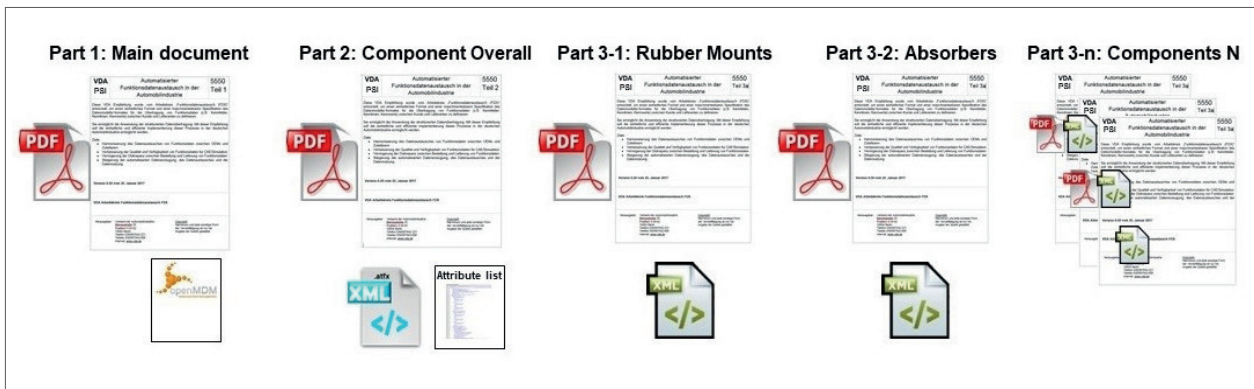


Figure 2-1: Structure of prostep ivip / VDA Recommendation

The component-specific templates will be made available in a FDX repository.

3 Fields of application

The following application examples describe typical scenarios for the data exchange format described in this document. These scenarios were used to identify the actual requirements for the data exchange format, which were then implemented in the proposed solution.

With regard to the four main roles, namely application administrator, data requester, data supplier and data user (Figure 3-1), we can distinguish the following areas of application:

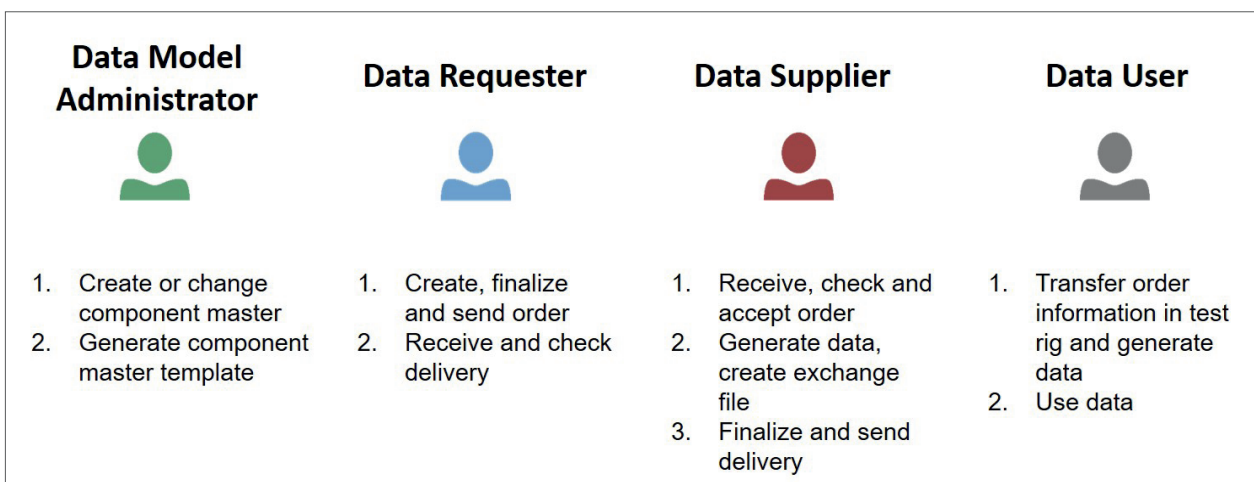


Figure 3-1: Main roles in the use of the FDX data exchange format

The application administrators are in charge of the definition of the attributes and their properties in the data model, and thus provide the basis for a standardized description of components and component types in the data model of the functional data exchange file. For data requesters and data suppliers, the template defines and standardizes the scope of information they can request and deliver.

The data requester uses the predefined attributes and properties in order to add concrete instructions, rules, setpoints, derived characteristic values, etc. to be submitted by the data supplier, whereby the FDX exchange format communicates these requirements in an unambiguous manner. Communication thus happens exclusively through the component-specific templates made available by the prostep ivip / VDA FDX Working Group.

The data supplier assigns actual values to the attributes and properties, following the rules laid down by the data requester. This assures the quality of the information. The data is then sent back to the data requester, using the FDX exchange format. The requester can now perform the relevant tests and process the data.

The data user transfers the functional data from the FDX exchange format to his processes and/or data management system. This can for instance happen through the automated transfer of the functional data to simulation models/ systems, or through the automatic application of test equipment parameters in testing systems/equipment. The FDX exchange format supports such automated processes as it contains comprehensive and standardized descriptions of all relevant data.

In the context of this document, the term "application administration" ("FA") is used in a number of constellations. Application administration is concerned with the technical data model, which serves as a template for further instantiations. Depending on the actual objectives, we distinguish between different application administration levels. The diagram below illustrates the various levels of implementation.

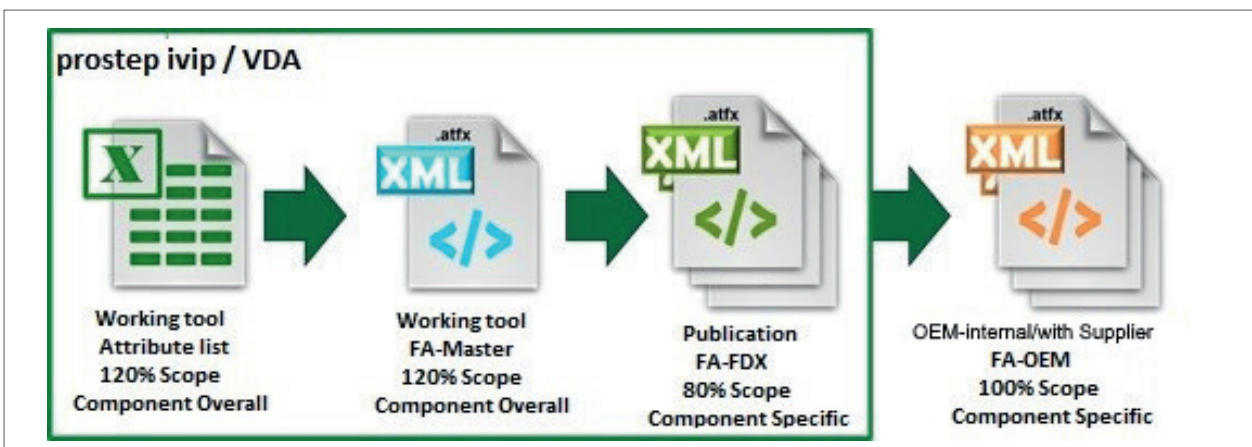


Figure 3-2: Use case "Customizing FA master"

Note:

When devising the templates, special attention was paid to the differentiation between mandatory and optional attributes. For concrete measurement orders, we recommend that the data requester does not change optional attributes to mandatory, unless there are compelling reasons to do so.

3.1 Use case application administrator: Customizing FA master (for component-specific additions)

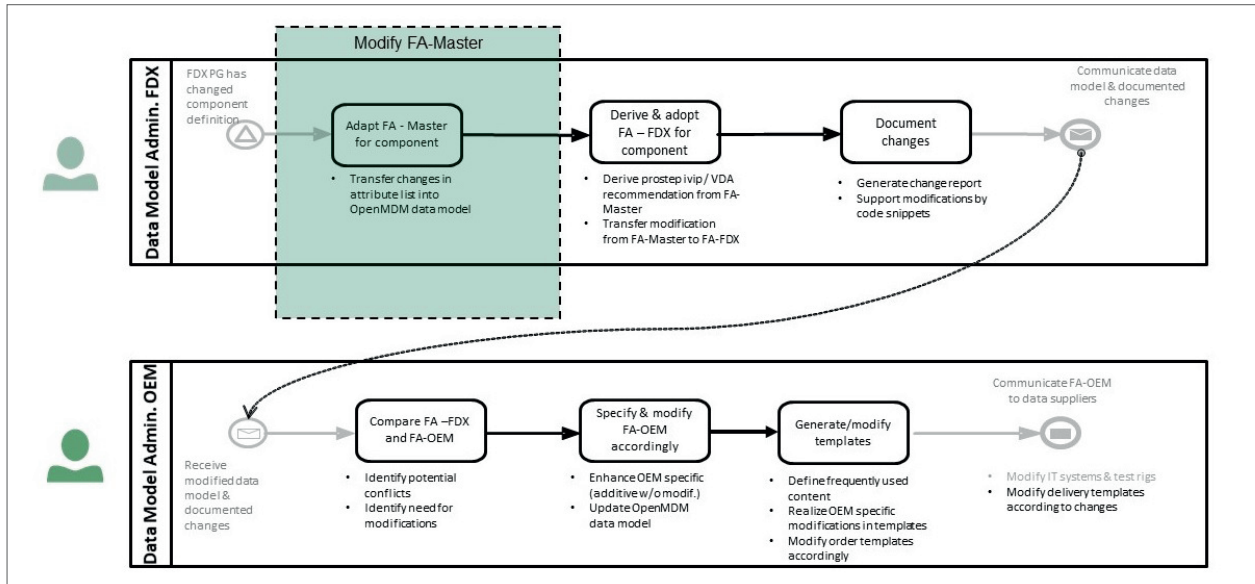


Figure 3-3: Use case "Customizing FA master"

Description/purpose	<p>Creation of application administration master (FA master), i.e. drafting of description attributes in categories "Unit under test" (component), "Operating parameters" (test performance) and "Setup" (testing setup), and definition of catalogue components (aggregates of description attributes).</p> <p>Definition of the attributes and their properties to provide a basis for the standardized description of various component types, so that data suppliers know exactly which information they must provide.</p> <p>The descriptions of the different component types are thus standardized, as they are based on a shared FA master.</p>
Scenario	<ol style="list-style-type: none"> 1. Creating component <ol style="list-style-type: none"> 1.1. The user defines catalogue components (attribute groups) according to the various categories and subcategories in the attribute list. 1.2. The user sets up the attributes within the catalogue component. In this process, the properties are defined, based on the attribute list. 2. Editing attributes and properties of existing catalogue components. <ol style="list-style-type: none"> 2.1. Deleting catalogue component 2.2. Deleting attribute 2.3. Creating new attribute 2.4. Edit attribute properties 3. Saving: The FA master is saved as an ATFX file on a suitable data storage medium, e.g. a hard disk.
Result	<p>An FA master in the form of an ATFX file conforming to ASAM ODS ATFX and the openMDM definition, containing the catalogue components and their attributes.</p>

Table 3-1: Use case application administrator: "Creating and editing FA master"

3.2 Use case application administrator: Defining and editing FA-FDX for component

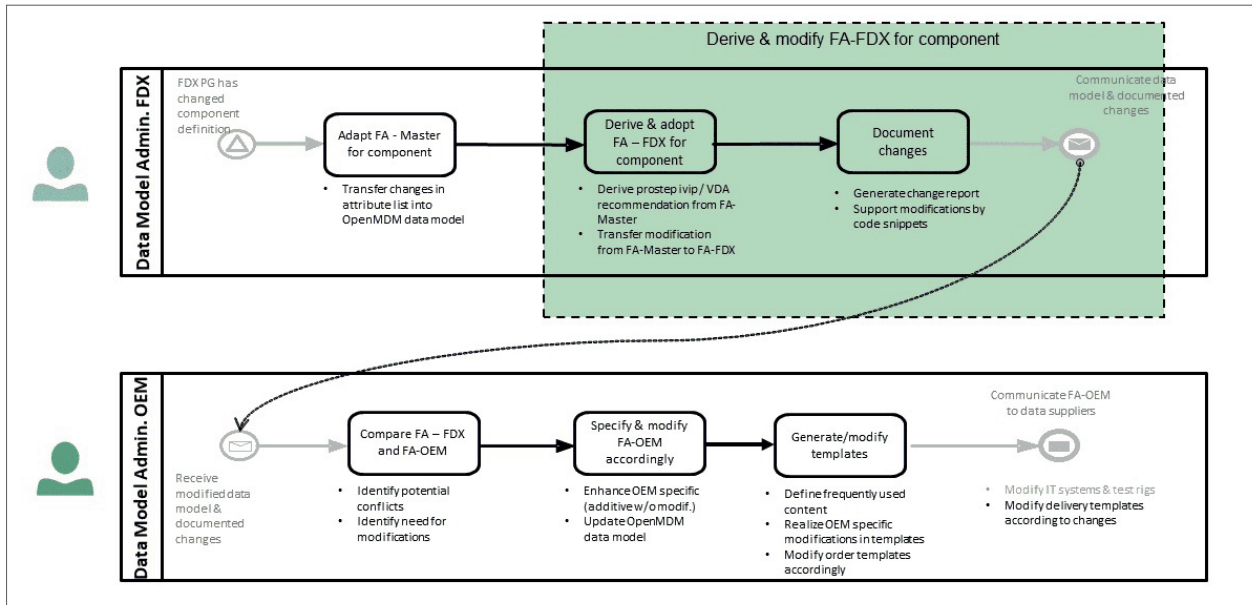


Figure 3-4: Use case "Defining and editing FA-FDX for component"

<p>Description/purpose</p>	<p>Creation/definition and editing of FDX application administration (FA-FDX), which is a template for a specific component type.</p> <p>By selecting catalogue components and attributes from the FA master, predefining certain attributes and choosing certain attribute combinations, an FA -FDX for future orders is created (serving as a template in which the attributes that must be submitted for a specific type of order are specified).</p>
<p>Scenario</p>	<ol style="list-style-type: none"> 1. Creating, editing and delete hierarchical groups (similar to directories) for greater transparency. 2. Creating, editing and deleting openMDM component templates, using the catalogue components from the FA master. 3. It is possible to set up sub-components of existing components, adding depth to the structure. 4. Defining whether a component is optional, active by default and/or variable depending on the test series. 5. Defining the attributes to be used for the component. 6. Defining attribute properties, i.e. optional, preset, write-protected, etc. 7. Creating, editing and deleting test step templates, including, if appropriate, selecting associated group. 8. Assigning and changing the assignment of unit under test templates, operating parameter templates and a setup template to the test step templates. 9. Creating and deleting FA-FDX (testing template), including, if appropriate, selecting associated group. 10. Assigning and changing the assignment of test step templates to the FA-FDX (testing template). 11. Saving: The FA-FDX master is saved as an ATFX file on a suitable data storage medium, e.g. a hard disk.
<p>Result</p>	<p>The FDX application administration for the component type has been created or changed and is now available with the correct syntax in the form of an FA-FDX.</p>

Table 3-2: Use case "Defining and editing FA-FDX for component"

3.3 Use case application administrator: Instantiating and editing FA-OEM for component

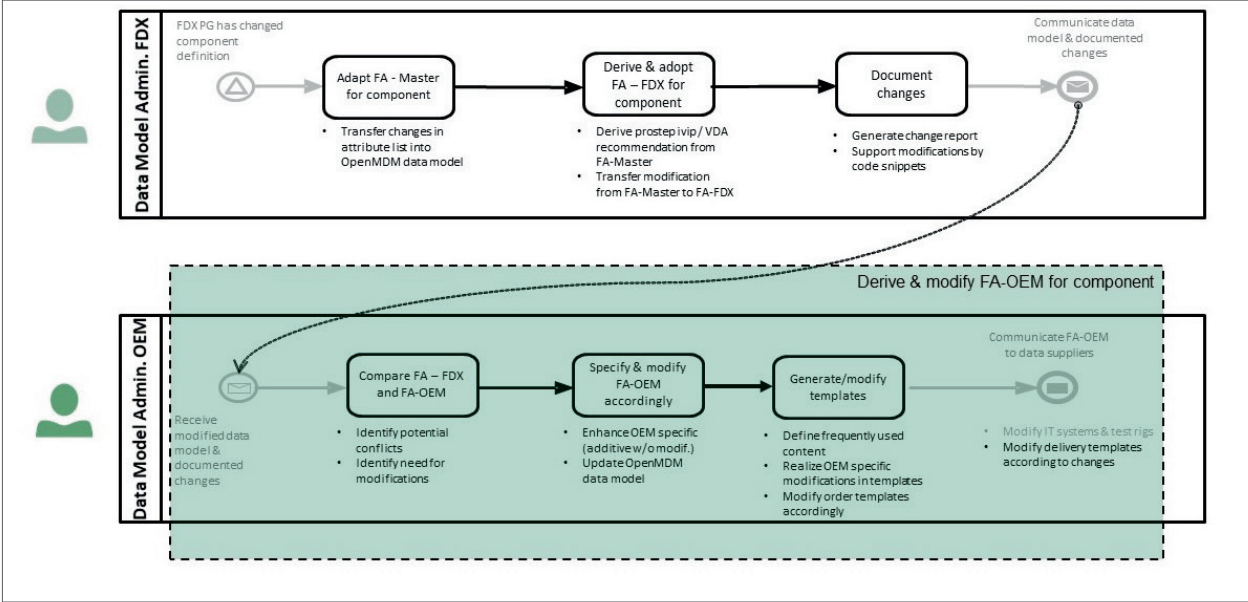


Figure 3-5: Use case "Instantiating and editing FA-OEM for component"

Description/purpose	Implementation of data model of the exchange format based on the FA-FDX with OEM-specific modifications. Updating of resulting format.
Scenario	<p>A: Analog to: FA-FDX</p> <ol style="list-style-type: none"> 1. Creating, editing and deleting hierarchical groups (similar to directories) for greater transparency. 2. Creating, editing and deleting openMDM component templates, using the catalogue components from the FA-FDX. 3. It is possible to set up sub-components of existing components, adding depth to the structure. 4. Defining whether a component is optional, active by default and/or variable depending on the test series. 5. Defining the attributes to be used for the component. 6. Defining attribute properties, i.e. optional, preset, write-protected, etc. 7. Creating, editing and deleting test step templates including, if appropriate, selecting associated group. 8. Assigning and changing the assignment of unit under test templates, operating parameter templates and a setup template to the test step templates. 9. Creating and deleting FA-OEM (testing template), including, if appropriate, selecting associated group. 10. Assigning and changing the assignment of test step templates to the FA-OEM (testing template). 11. Saving: The FA-OEM is saved on a suitable data storage medium, e.g. a hard disk. <p>Analog to: FA master:</p> <ol style="list-style-type: none"> 1. Creating component 2. The user defines the catalogue components (attribute groups).

Scenario	<ol style="list-style-type: none"> 3. The user creates the attributes within the catalogue component. 4. Editing attributes and properties of existing catalogue components. 5. Deleting catalogue component 6. Deleting attribute 7. Creating new attribute 8. Edit attribute properties
Result	<p>An FA-OEM in the form of an ATFX file conforming to ASAM ODS ATFX and the openMDM definition, containing the catalogue components and their attributes.</p> <p>OEM-specific application administration and templates for component type, based on FA-FDX.</p>

Table 3-3: Use case "Instantiating and editing FA-OEM"

3.4 Use case data requester: Creating, completing and transmitting order

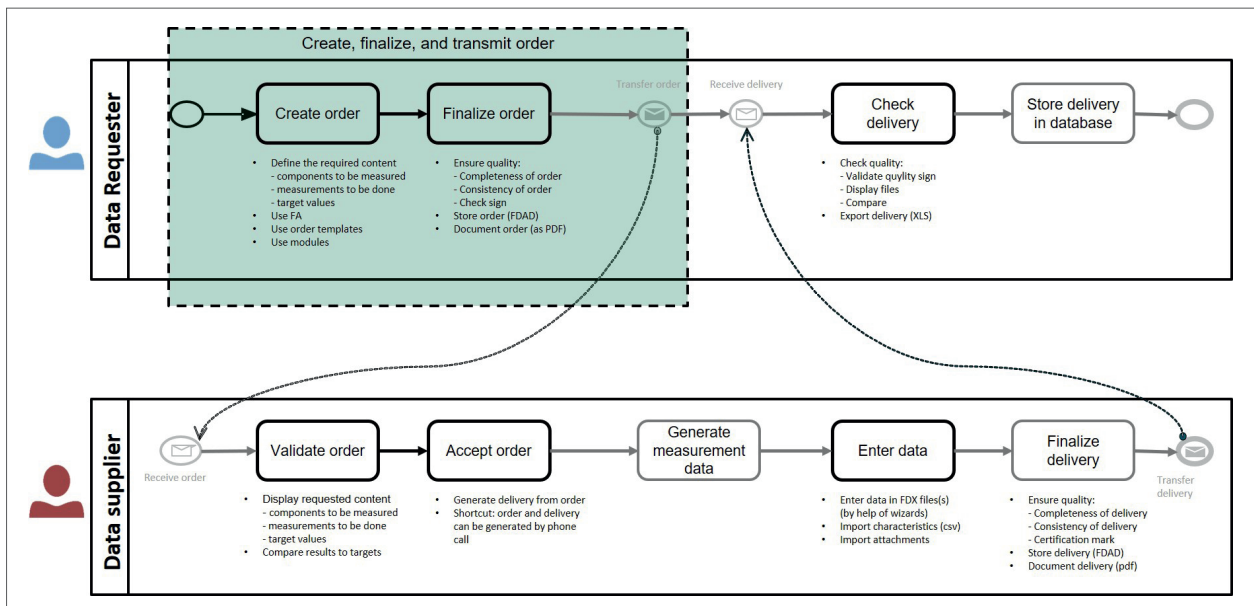


Figure 3-6: Use case "Creating, finalizing and transmitting order"

Description/purpose	<p>The data requester defines the required information in the ATFX file (by assigning the attribute value or selecting it from the value list).</p> <p>He then finalizes the order by documenting the quality of the ATFX file content by means of one or more validation certificates files, and bundling the ATFX file together with the validation certificates file(s) and, if required, additional referenced documents (pictures, pdf, etc.) in the container formed by the functional data exchange file.</p> <p>Subsequently, the order is transferred electronically to the data supplier.</p>
---------------------	---

Scenario	<ol style="list-style-type: none"> 1. Definition of the following parameters, using the application administration, order templates and elements, and through direct data input: <ol style="list-style-type: none"> 1.1. Measurement order data (e.g. order status, order number) 1.2. Order meta data (e.g. supplier, design version, part description, weight) 1.3. Dataset meta data (e.g. administrative information, dataset identification, information requirement specification) 1.4. Component-specific additional information (e.g. material, dimensions, version) 1.5. Unit under test meta data (component/quality status, part identification, part description, preliminary testing, component modification) 1.6. Test equipment setup: (e.g. data generation: calculation/estimation/test bench, test bench adaption, component-specific modifications) 1.7. Test equipment parameters: (e.g. administrative information, test type, preload, preconditioning, testing program, ambient conditions) 1.8. Functional setpoint data (e.g. physical dimensions, static measurement, dynamic measurement, scalable parameters, characteristic curves, derived characteristic values (e.g. pitch)) 2. Finalizing order: <ol style="list-style-type: none"> 2.1. Generating validation certificates for individual datasets 2.2. Generating validation certificates for complete file 2.3. Saving all information in functional data exchange file 3. Transmission of functional data exchange file to data supplier
Result	Verified, complete measurement order has been transmitted to the data supplier

Table 3-4: Use case data requester: Creating, finalizing and transmitting order

3.5 Use case data supplier: Receiving, examining and accepting order

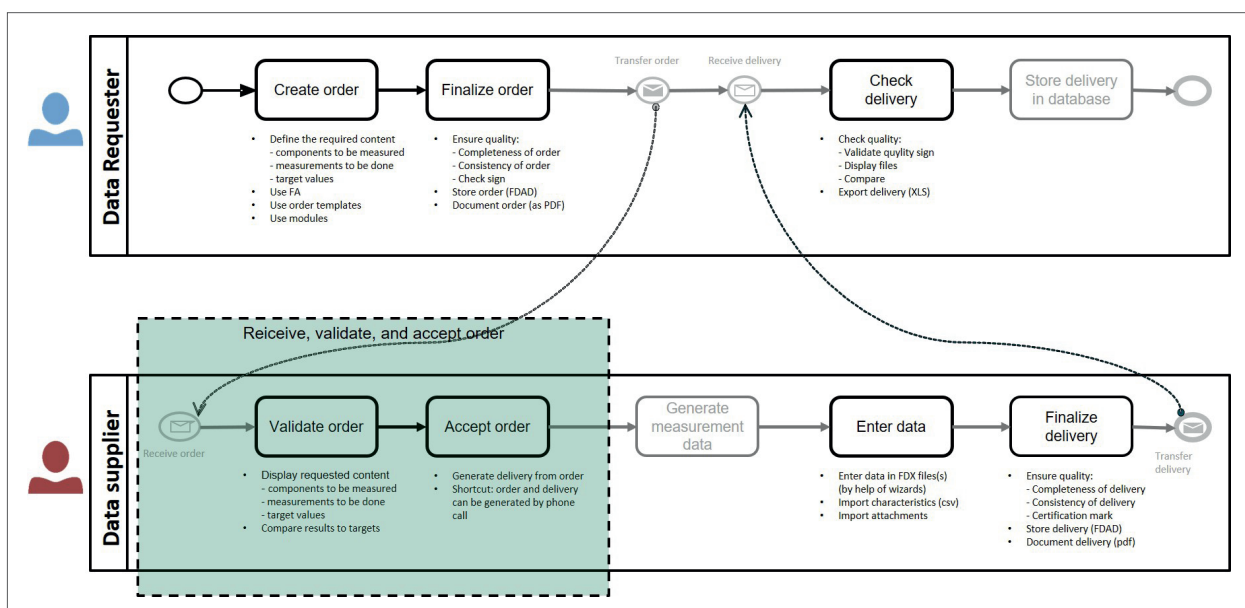


Figure 3-7: Use case "Receiving, examining and accepting order"

<p>Description/purpose</p>	<p>The data supplier receives the functional data exchange file containing the measurement order.</p> <p>The data supplier opens the file with a suitable software tool, checks the order and accepts it.</p> <p>He then generates a new file in which he adds the requested data associated with the order data.</p>
<p>Scenario</p>	<ol style="list-style-type: none"> 1. The data supplier receives and checks the order: <ol style="list-style-type: none"> 1.1. He loads and opens the functional data exchange file. 1.2. The data supplier validates the validation certificate to ensure that no changes have been made to the functional data exchange file during the last quality check. 1.3. He visualizes the data described in the above use case (data requester: "Creating, finalizing and transmitting order") to examine the current state of completeness of the dataset(s) and to evaluate the order. For this purpose, he can: Compile scalar/derived characteristic values in a table Generate characteristic curves Compare characteristic curves Compare scalar/derived characteristic values in tables 2. The data supplier accepts the order and uses the setpoint datasets to generate actual datasets that are prepopulated with functional meta data values from the setpoint dataset. <ol style="list-style-type: none"> 2.1. He ensures that the release/version of the data model behind his datasets is identical with that of the requester. 2.2. He applies the order information from the datasets of the requester 2.3. He sets the order status to 'Delivery' and "in progress"
<p>Result</p>	<p>The measurement order from the data requester has been examined, evaluated and accepted. The supplier datasets are ready to be filled with actual values.</p>

Table 3-5: Use case data supplier: Receiving, examining and accepting order

3.6 Use case data supplier: Entering, finalizing and submitting data

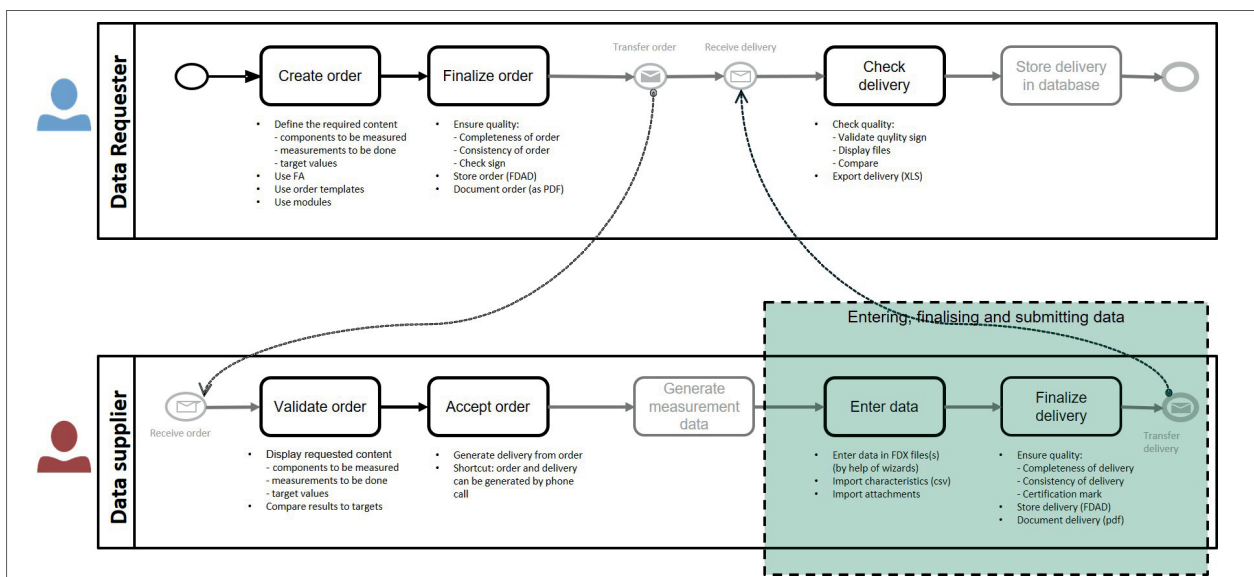


Figure 3-8: Use case "Entering, finalizing and submitting data"

Description/purpose	<p>The data supplier enters all relevant information as requested (completing the measured values for the requested values).</p> <p>He then validates the quality of the data by checking it for completeness and consistency (all mandatory attributes, requested datasets and preset value ranges included?) and by generating a validation certificate.</p> <p>The resulting data is saved in the functional data exchange file and sent to the data requester.</p>
Scenario	<ol style="list-style-type: none"> 1. Entering actual values, using templates, elements and direct data input: <ol style="list-style-type: none"> 1.1. Measurement order data (e.g. order status) 1.2. Order meta data (e.g. supplier, design version, part description) 1.3. Dataset meta data (e.g. administrative details, dataset identification, information re. requirement specification) 1.4. Component-specific additional information (e.g. material, dimensions, version) 1.5. Entering unit under test meta data (component/quality status, part identification, part description, preliminary tests, component modification) 1.6. Test equipment setup: (e.g. data generation: calculation/estimation/test bench, test bench adaption, component-specific modifications) 1.7. Test equipment parameters (e.g. administrative information, test type, preload, preconditioning, testing program, ambient conditions) 1.8. Actual functional data (e.g. physical dimensions, static measurement, dynamic measurement, derived characteristic values (e.g. pitch), characteristic curves) 1.9. Attachments: For documentation purposes, the user has the option to attach pictures, drawings, etc. to the order. 2. Finalizing the order: <ol style="list-style-type: none"> 2.1. Generating validation certificates for individual datasets 2.2. Generating validation certificate for complete file 2.3. Saving of all information in functional data exchange file 3. Transmission of functional data exchange file to data requester
Result	Completed functional data exchange file with validation certificates has been forwarded to the data requester.

Table 3-6: Use case data supplier: Entering, finalizing and submitting data

3.7 Use case data user: Storage of test data in ATFX format at OEM or supplier

Description/purpose	<p>Automated loading of data from functional data exchange file or ATFX file to OEM's or supplier's IT system</p> <p>Efficient, error-free transfer of data for use in internal processes.</p>
Scenario	<ol style="list-style-type: none"> 1. The data requester receives the functional data exchange file from the data supplier and checks the data quality. 2. The data requester saves the functional data exchange file in the data management system and releases it. 3. The data user loads the relevant information from the functional data exchange file to the data management system of his machine, system or model.
Result	The data required for the data user use case is loaded without errors from the functional data exchange/ATFX file to the data user's machine, system or model.

Table 3-7: Use case data user: Storage of testing data in ATFX format at OEM or supplier

3.8 Content and structure of functional data exchange file (example)

This chapter describes selected contents and their structures in the data model, referring to a concrete example. The contents and structures are presented in tables that might deviate significantly from the actual representation in the FDX XML exchange file.

The field and label names of the attributes as well as the associated values are examples only. Field and label names are subject to change and this document might not represent the latest valid version. Some of the terms, particularly in the rule expressions, are based on the openMDM and ASAM standards.

3.8.1 Structure and attribute categories

This chapter describes the structure of the FDX format. The attributes are grouped at three structural levels:

- Main categories
 - Subcategories
 - Attributes

The main categories normally contain several subcategories, which in turn contain several attributes.

Category	Sub category	Field name (Attribute name)	Range	Field type	Unit type	Block rules	
Test Equipment Setup (Messgerätesetup)	DataOrigin	Method	ValueList.Name.Generic.GenericDataOrigin	String	Dimensionless		
	Simulation (Simulation)	SoftwareName		String	Dimensionless	TstEquip.DataOrigin.Method == 'Simulation'	
		Version		String	Dimensionless		
		Remark		String	Dimensionless		
	Estimation	Designation			String	Dimensionless	TstEquip.DataOrigin.Method == 'Estimation'
		Remark				Dimensionless	

Figure 3-9: Excerpt from main category "Test equipment parameters" with various subcategories and attributes

The top level consists of the following nine main categories:

- Measurement order
- Order meta data
- Component-specific additional information
- Dataset meta data
- Unit under test
- Test equipment setup
- Test equipment parameters
- Function data
- Derived characteristic values

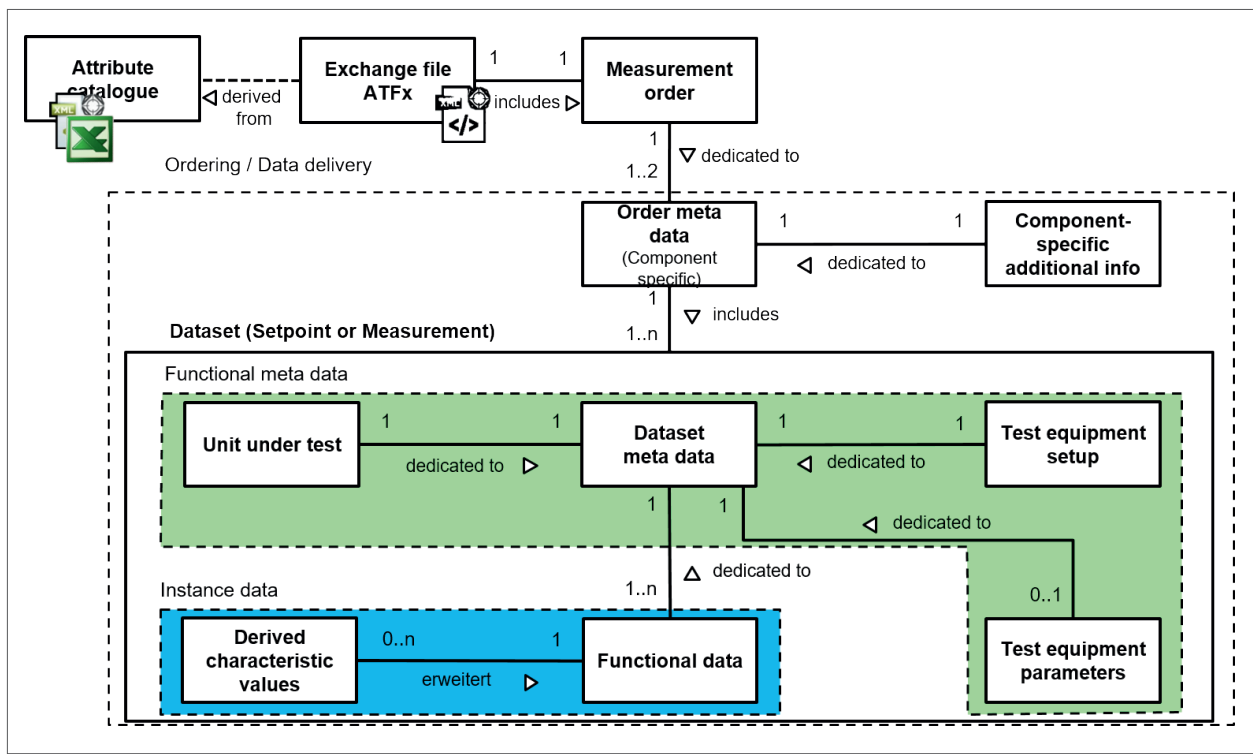


Figure 3-10: Structure of applicable application model

A *Measurement order* is generated on the basis of a generic attribute catalogue that contains all possible attribute categories and contents. The person setting up the measurement order only selects those attributes that are relevant for the actual measurement order. A complete list of all available attributes is included in parts 2 and 3-x of this prostep ivip / VDA Recommendation.

Depending on its status, the *Measurement order* contains only requested data or both requested and delivered data. This is indicated in **Figure 3-10** with notation "1..2" for the relationship between *Measurement order* and *Order meta data*. "1" means that there is only one dataset in the order; "2" means that there is one dataset for the request and one dataset for the data delivery.

Both the data request and the data delivery contain the *order meta data*, with optional *Component-specific additional information*, at the top level.

Depending on the requirements, the data request can include one or more setpoint datasets or setpoint and measurement datasets (**Figure 3-10**, notation "1..n" for the relationship between *Order meta data* and *Dataset (Setpoint or Measurement)*).

A setpoint or measurement dataset consists of function meta data and *instance data*. The function meta data contains *Dataset meta data* regarding the *Unit under test*, *Test equipment setup* and *Test equipment parameters*. The *Test equipment parameters* are optional and might therefore be omitted. *Unit under test* and *Test equipment setup* are mandatory.

Depending on the actual requirements, the *Dataset meta data* can be assigned multiple instances of data. The instance data consists of the *Functional data*, which can be extended by one or more *Derived characteristic values*.

Figure 3-11 describes the main information exchanged with the FDX format.

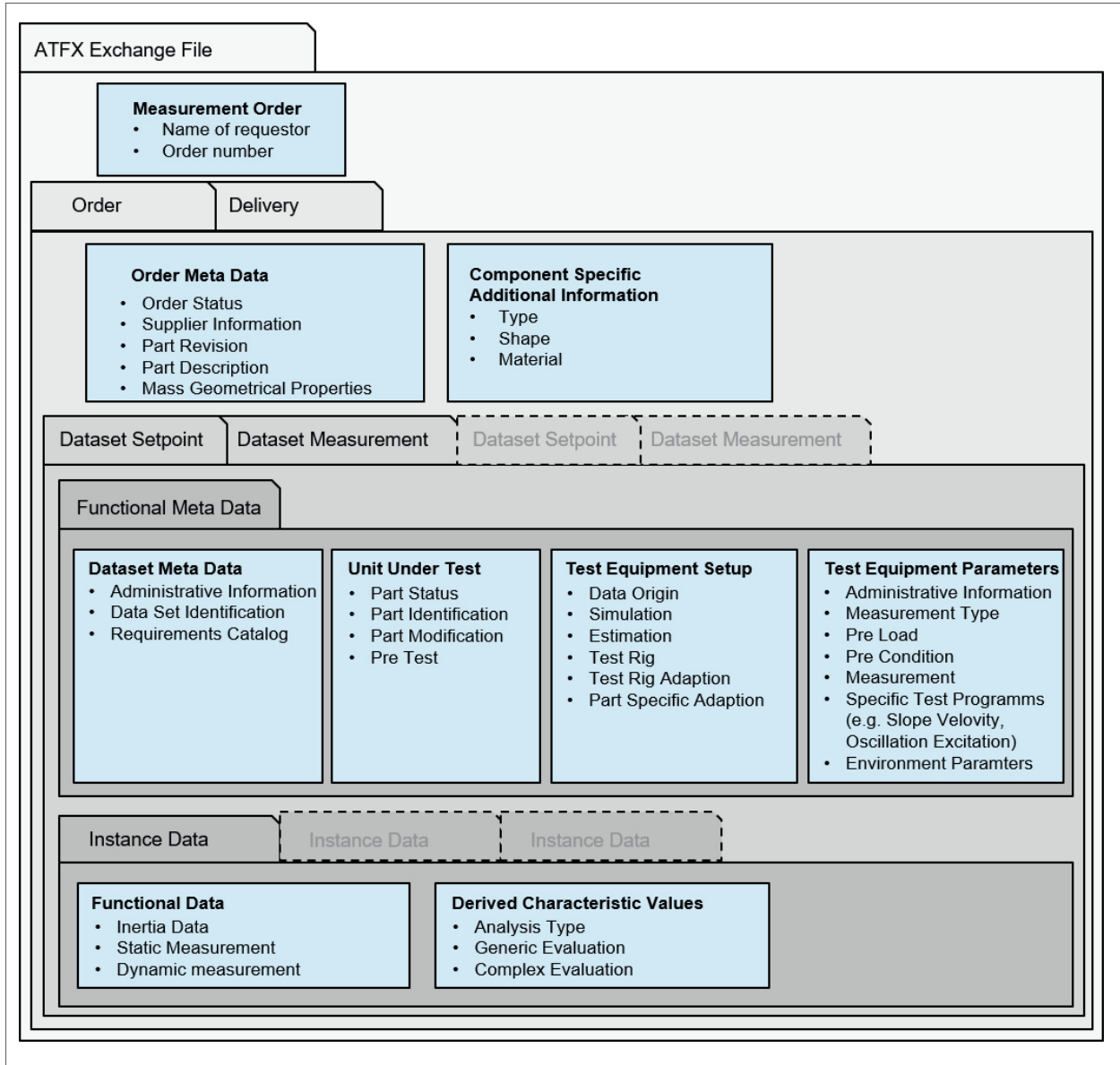


Figure 3-11: Functional data exchange file: main categories, subcategories and contents

Figure 3-12 shows the logic behind the FDX format. It aims at providing templates that can be configured and filled with data to suit specific requirements.

The level below the subcategories is the attribute level. As a rule, all attributes are defined as mandatory or optional attributes, depending on whether the data supplier is obliged to submit the information or not. There are also controlling and simple attributes.

Category	Sub category	Field name (Attribute name)	Range	Field type	Unit type	Block rules	Rule	Optional - Optional / Mandatory in text	Optional - Optional / Mandatory in text	
Test Equipment Parameters	Admin Info	Identification		String	Dimensionless			optional	optional	
	MeasurementType (Messungstyp)	MotionDirection	ValueList.Name.Generic.GenericMotionDirection	String	Dimensionless				mandatory	mandatory
		SpatialDirection	ValueList.Name.Generic.GenericSpatialDirection	String	Dimensionless				mandatory	mandatory
		DesignDirectionTrans	ValueList.Name.Generic.GenericIdentification	String	Dimensionless			TstSq.MeasurementType.MotionDirection == 'Translational'	mandatory	mandatory
		DesignDirectionRot	ValueList.Name.Generic.GenericDesignDirectionR	String	Dimensionless			TstSq.MeasurementType.MotionDirection == 'Rotational'	mandatory	mandatory
		Preload	ValueList.Name.Generic.GenericYesNo	String	Dimensionless				mandatory	mandatory
		Preload2	ValueList.Name.Generic.	String	Dimensionless				mandatory	mandatory
		Preconditioning	ValueList.Name.Generic.GenericYesNo	String	Dimensionless				mandatory	mandatory
		TimeChannelType	ValueList.Name.Generic.GenericTimeChannelType	String	Dimensionless				mandatory	mandatory
		TimeIncrement			Decimal	Time		TstSq.MeasurementType.TimeCh	mandatory	mandatory
		Frequency			Decimal	Frequency		TstSq.MeasurementType.TimeCh	mandatory	mandatory
		MeasurementDynamics	ValueList.Name.Generic.GenericMeasurementDyn	String	Dimensionless				mandatory	mandatory
		Preload (Vorlast)	MotionDirection	ValueList.Name.Generic.GenericMotionDirection	String	Dimensionless		TstSq.MeasurementType.Preload == 'Yes'		mandatory
	SpatialDirection		ValueList.Name.Generic.GenericSpatialDirection	String	Dimensionless				mandatory	mandatory
	ControlType		ValueList.Name.Generic.GenericControlType	String	Dimensionless				mandatory	mandatory
	Displacement				Decimal	Length		TstSq.Preload.ControlType == 'Dis	mandatory	mandatory
	Angle				Decimal	Planeangle		TstSq.Preload.ControlType == 'Displacement' && TstSq.Preload.MotionDirection == 'Rotational'	mandatory	mandatory
	Force				Decimal	Force		TstSq.Preload.ControlType == 'For	mandatory	mandatory
	Moment				Decimal	Moment		TstSq.Preload.ControlType == 'For	mandatory	mandatory
	Preload2 (Vorlast 2)	MotionDirection	ValueList.Name.Generic.	String	Dimensionless		TstSq.Measurement		mandatory	mandatory
		SpatialDirection	ValueList.Name.Generic.	String	Dimensionless				mandatory	mandatory
		ControlType	ValueList.Name.Generic.	String	Dimensionless				mandatory	mandatory
		Displacement			Decimal	Length		TstSq.Preload2.ControlType == 'Di	mandatory	mandatory
		Angle			Decimal	Planeangle		TstSq.Preload2.ControlType == 'Di	mandatory	mandatory

Figure 3-12: Examples of attributes, value ranges, block rules and simple rules

Rules are a means to add dynamic relations between different attributes in the template. With them it is possible to add or remove attributes or whole components depending on a specific other attribute value when it is set by the user. Whether an attribute's value is mandatory or optional to fill can be dynamically changed in the same manner using rules.

In the figures, control attributes are shown in blue. They form an important part of the template concept and are prefilled with values defined by the prostep ivip / VDA FDX Working Group. The data requester/supplier must select values from the predefined list and is prevented from entering any other values.

The control attributes are assigned rules that determine the content and structure of the data to be submitted. These rules define whether an attribute is mandatory or optional, and disables or enables attributes, based on previously made selections. There are simple rules that control individual attributes, and block rules that disable/enable multiple attributes or all attributes in a subcategory.

3.8.2 Measurement order

Main category Measurement order contains the following attributes relating to general information regarding the ordering party, the requester and the order:

- CustomerIdentificationNumber (D-U-N-S number)
- CustomerName
- RequesterDepartment
- RequesterName
- RequesterPhone
- SubOrder (multiple partial orders)
- OrderNumber
- SubOrder

In the example shown here, the ordering party is "MusterOEM AG" and the *CustomerIdentificationNumber* is "123456789" (nine-digit D-U-N-S format). The next three optional attributes have been left blank. The order is not to be divided into partial orders, and the *OrderNumber* is "9876-2017" (**Figure 3-13**).

Category	Sub category	Field name (Attribute name)	Range	Field type	Unit type	Block rules	Rule	Order		Delivery	
								Optional - Optional / Mandatory in text	Optional - Optional / Mandatory in text	Unit	Value
Measurement Order	OrderInfo	CustomerIdentificationNumber		String	Dimensionless			optional	optional	-	123456789
		CustomerName		String	Dimensionless			mandatory	mandatory	-	MusterOEM AG
		RequesterDepartment		String	Dimensionless			optional	irrelevant	-	
		RequesterName		String	Dimensionless			optional	irrelevant	-	
		RequesterPhone		String	Dimensionless			optional	irrelevant	-	
		SubOrder	ValueList.Name.Generic.GenericYesNo	String	Dimensionless			optional	optional	-	No
		OrderNumber		String	Dimensionless			mandatory	mandatory	-	9876-2017
		SubOrderNumber		String	Dimensionless			M\$Order.OrderInfo.SubOrder == 'Yes'	mandatory	mandatory	-

Figure 3-13: Main category "Measurement order" with associated attributes and sample content

Each ATFX file contains exactly one measurement order. The main category Measurement Order has the subcategory OrderInfo. All associated attributes are thus directly assigned to the subcategory.

3.8.3 Order meta data and component-specific additional information

Assigned to the *measurement order* is the actual order and, depending on the order status (e.g. if the order is to be edited and completed by the supplier), the data submission. Information regarding the supplier and the component as well as component-specific information for the request and delivery of data are filed in the main categories *Order Meta Data* and *Component Specific Additional Information*.

Category	Sub category	Field name (Attribute name)	Range	Field type	Unit type	Block rules	Rule	Order	Delivery
								Optional - Optional / Mandatory in text	Optional - Optional / Mandatory in text
Order Meta Data	BundleStat	BundleType	ValueList.Name.Generic.	String	Dimensionless			mandatory	mandatory
	Supplier	SupplierIdentification Number		String	Dimensionless			optional	mandatory
	(Lieferant)	SupplierName		String	Dimensionless			mandatory	mandatory
	PartRev	CustomerPartDescrip		String	Dimensionless			mandatory	optional
	(Konstruktionsstand)	CustomerPartNumbe		String	Dimensionless			mandatory	mandatory
		CustomerPartNumbe		String	Dimensionless			mandatory	mandatory
		StageRemark	CustomerDevelopme ntStageRemark		String	Dimensionless			optional
	PartDesc	Component		String	Dimensionless			mandatory	mandatory

Figure 3-14: Excerpt from main category "Order meta data" with associated attributes and sample content

In the example shown here, order meta data attributes *BundleType*, *SupplierIdentificationNumber*, *SupplierName*, *CustomerPartDescription*, *CustomerPartNumber* are set to the values shown in the last column in **Figure 3-14**.

In the component specific additional information, the design type is a rubber mount "Without axial stop" (**Figure 3-15**). The main direction of the component is to be translational in x-direction. As soon as attribute *DirectionTranslationXPrincipal* is assigned value "yes", the associated rule for the activation of attribute *StiffnessTranslationalXNominal* is applied and a decimal value of unit type "LocalStiffness" must be entered. At the same time, the attributes for main direction *y* and *z* translational, and *x*, *y* and *z* rotational are disabled.

Category	Sub category	Field name (Attribute name)	Range	Field type	Unit type	Block rules	Rule	Order	Delivery	Unit	Value
								Optional - Optional / Mandatory in text	Optional - Optional / Mandatory in text		
Component Specific Additional Information	AddInfo	Material		String	Dimensionless			optional	optional	-	Without axial stop
		HardnessShore		Decimal	Hardness			optional	mandatory	-	
		DirectionTranslationXPrincipal	ValueList.Name.Generic.G enericYesNo	String	Dimensionless			optional	optional	-	Yes
		StiffnessTranslationXNominal		Decimal	LocalStiffness		AddInfoPartSpec.Ad	mandatory	mandatory	N/mm	8500
		DirectionTranslationYPrincipal	ValueList.Name.Generic.G enericYesNo	String	Dimensionless			optional	optional	-	No
		DirectionTranslationZPrincipal	ValueList.Name.Generic.G enericYesNo	String	Dimensionless			mandatory	mandatory	-	
		StiffnessRotationXNominal		Decimal	LocalRotational Stiffness		AddInfoPartSpec.Ad	mandatory	mandatory	-	
		StiffnessRotationZNominal		Decimal	LocalRotational Stiffness		AddInfoPartSpec.Ad	mandatory	mandatory	-	
		Characteristic		String	Dimensionless			optional	optional	-	GF-X1
		LengthFree		Decimal	Length			optional	mandatory	mm	75

Figure 3-15: Excerpt from main category "Component-specific additional information" with associated attributes and sample content

3.8.4 Dataset meta data

In subcategory *DataSetID*, the *DataSetName*, *Version*, and *TimeStampCreation* as well as the Originator are specified (see *Figure 3-16*).

Category	Sub category	Field name (Attribute name)	Range	Field type	Unit type	Block rules	Rule	Order		Delivery	
								Optional - Optional / Mandatory in text	Optional - Optional / Mandatory in text	Unit	Value
Data Set Meta Data	DataSetID	DataSetName		String	Dimensionless			mandatory	mandatory	-	Rubber mount 001
		Version		Integer	Dimensionless			mandatory	mandatory	-	002
		TimeStampCreation		Date	Dimensionless			mandatory	mandatory	-	20170120-085831
		TimeStampModification		Date	Dimensionless			mandatory	mandatory		
		DataSetSourceFile		String	Dimensionless			mandatory	mandatory		
		TimeStampMeasurement		Date	Dimensionless			irrelevant	optional		
		MeasuredBy		String	Dimensionless			irrelevant	optional		
		ModifiedBy		String	Dimensionless			mandatory	mandatory		
		Originator		String	Dimensionless			mandatory	mandatory	-	Max Muster
	Remark		String	Dimensionless			optional	optional			
	ReqCat	Description		String	Dimensionless			optional	optional		
		Version		String	Dimensionless			optional	optional		
		Conformity	ValueList.Name.Generic.GenericYesNo	String	Dimensionless			irrelevant	mandatory		
		ConformityRemark		String	Dimensionless		DataSetMetaDat.ReqCat.Conformity == 'No'	irrelevant	mandatory		

Figure 3-16: Main category "Dataset meta data" with associated attributes and sample content

3.8.5 Unit under test

Subcategory *Unit under test* caters for information that is specific for the unit under test. In this example, the unit under test has undergone preliminary test "Wave endurance test with ozone chamber", features a modification and a bore for temperature measurement in the mount core (*Figure 3-17*).

By selecting value "yes" for the control attributes *PreTest* and *PartModification*, subcategories *PreTest* and *PartModification* as well as the associated attributes are enabled and the relevant value can be entered.

Category	Sub category	Field name (Attribute name)	Range	Field type	Unit type	Block rules	Rule	Order	Delivery	Unit	Value
								Optional - Optional / Mandatory in text	Optional - Optional / Mandatory in text		
Unit UnderTest	PartStatus	SampleStatus		String	Dimensionless			optional	optional		
		QualityStatus		String	Dimensionless			optional	optional		
		VulcanisationStatus		String	Dimensionless			optional	optional		
		PreTest	ValueList.Name.Generic.GenericYesNo	String	Dimensionless			mandatory	mandatory	-	Yes
	PartModification	ValueList.Name.Generic.GenericYesNo	String	Dimensionless			mandatory	mandatory	-	Yes	
	PartID	SupplierID		String	Dimensionless			optional	mandatory		
		CustomerID		String	Dimensionless			mandatory	mandatory		
		BatchNumber		String	Dimensionless			optional	optional		
		ProductionDate		Date	Dimensionless			optional	optional		
	PreTest	Remark		String	Dimensionless	PartUnderTest.PartStatus.PreTest == 'Yes'		optional	mandatory		Wave endurance test with ozone chamber
		Attachment		File	Dimensionless			optional	optional		20170120-0923-Pretest-WET.pdf
	PartModif	Remark		String	Dimensionless	PartUnderTest.PartStatus.PartModification == 'Yes'		optional	mandatory		Borehole in mount core for temperature measurement
		Attachment		File	Dimensionless			optional	optional		20170120-0923-UUT-Mod-BHinMC.pdf

Figure 3-17: Main category "Unit under test" with associated attributes and sample content

3.8.6 Test equipment setup

Main category test equipment setup contains information relating to the *DataOrigin*. The data normally originates from tests/measurements, but the system also caters for data determined in simulations and for estimated values (Figure 3-18).

In our example, the data originates from a measurement, and attribute *data origin* is set to "measurement". Through the block rules, subcategories *test bench* and *test bench adaptation* are activated and the attributes for the designation of the *TestRig* and for the *TestRigAdaptation*, as well as references to the respective *Attachment* can be entered.

Category	Sub category	Field name (Attribute name)	Range	Field type	Unit type	Block rules	Rule	Order	Delivery	Unit	Value
								Optional - Optional / Mandatory in text	Optional - Optional / Mandatory in text		
Test Equipment Setup	DataOrigin	Method	ValueList.Name.Generic.GenericDataOrigin	String	Dimensionless			mandatory	mandatory	-	Measurement
	Simulation	SoftwareName		String	Dimensionless	TstEquip.DataOrigin.Method == 'Simulation'		optional	mandatory		
		Version		String	Dimensionless			optional	optional		
		Remark		String	Dimensionless			optional	optional		
	Estimation	Designation		String	Dimensionless	TstEquip.DataOrigin.Method == 'Estimation'		optional	mandatory		
		Remark		String	Dimensionless			optional	optional		
	TestRig	Identification		String	Dimensionless	TstEquip.DataOrigin.Method == 'Measurement'		optional	mandatory	-	Servohydraulic testrig.003
		SoftwareName		String	Dimensionless			optional	mandatory	-	Test software
		Remark		String	Dimensionless			optional	optional	-	-
		Attachment		File	Dimensionless			optional	optional		20170120-100421-ServoHydraulicTestrig.003.pdf
	TestRigAdap	Identification		String	Dimensionless	TstEquip.DataOrigin.Method == 'Measurement'		optional	optional	-	Big proving ring with measuring box at ring side
		PartNumber		String	Dimensionless			optional	optional		
		PartNumberVersion		String	Dimensionless			optional	optional		
		DrawingNumber		String	Dimensionless			optional	optional		
		DrawingNumberVersion		String	Dimensionless			optional	optional		
		Remark		String	Dimensionless			optional	optional		
Attachment		File	Dimensionless		optional	optional		20170120-10063f-BPR-MB.pdf			

Figure 3-18: Main category "Test equipment setup" with associated attributes and sample content

3.8.7 Test equipment parameters

This chapter describes the definition of *test equipment parameters* (example: rubber mount).

- The test is a static test.
- The movement is translational in x-direction.
- The component is not to be preloaded.
- The max. test speed is 10 mm/min.
- The excitation is to be sinusoidal.
- The test is to be performed with force control from +10 to -10kN.
- SubOrder

The relevant attributes are found in main category *test equipment parameters*, subcategory *test type*.

Subcategory *test type* contains four attributes that are relevant for the definition of the above test. These four attributes are control attributes. As described in 3.8.1, **Figure 3-12**, this means that the user must select from a list of predefined values:

- Attribute: *MotionDirection* - values: "translational", "rotational"
- Attribute: *SpatialDirection* - values: "x", "y" or "z"
- Attribute: *Preload* - values: "yes" or "no"
- Attribute: *Preload2* - values: "yes" or "no"
- Attribute: *Preconditioning* - values: "yes" or "no"

Category	Sub category	Field name (Attribute name)	Range	Field type	Unit type	Block rules	Rule	Order		Delivery		Value
								Optional - Optional / Mandatory in text	Optional - Optional / Mandatory in text	Unit	Value	
Test Equipment Parameters	Admin Info	Identification		String	Dimensionless			optional	optional	-		
	Measurement type	MotionDirection	ValueList.Name.Generic.GenericMotionDirection	String	Dimensionless			mandatory	mandatory	-	Translational	
		SpatialDirection	ValueList.Name.Generic.GenericSpatialDirection	String	Dimensionless			mandatory	mandatory	-	X	
		Preload	ValueList.Name.Generic.GenericYesNo	String	Dimensionless			mandatory	mandatory	-	No	
		Preload2	ValueList.Name.Generic.GenericYesNo	String	Dimensionless			mandatory	mandatory	-	No	
		Preconditioning	ValueList.Name.Generic.GenericYesNo	String	Dimensionless			mandatory	mandatory	-	No	
	Preload	MotionDirection	ValueList.Name.Generic.GenericMotionDirection	String	Dimensionless		TstSq.MeasurementType.Preload == 'Yes'	mandatory	mandatory	-		
		SpatialDirection	ValueList.Name.Generic.GenericSpatialDirection	String	Dimensionless			mandatory	mandatory	-		
		ControlType	ValueList.Name.Generic.GenericControlType	String	Dimensionless			mandatory	mandatory	-		

Figure 3-19: Subcategory "Test type" with associated attributes and sample content

For the above example, the following values must be selected: For *MotionDirection* "translational"; for *SpatialDirection*: "x"; for *Preload*, *Preload2* and for *Preconditioning*: "no".

As attributes *With Preload*, *Preload2* and *Preconditioning* have been assigned value "no", subcategories *Preload*, *Proeload2* and *Preconditioning* are disabled through block rules and are thus skipped (if *Preload* is set to "no", subcategory *Preload* is not admissible; if *Preload* is set to "yes", subcategory *Preload* is mandatory).

The next category, i.e. *Measurement* contains the following four control attributes:

- ProgramType
- VariedQuantity
- SlopeType

First select the value for attribute *ProgramType*. As the excitation is to be sinusoidal and over multiple cycles, value "periodic" must be selected. With "periodic", attributes *VariedQuantity* and *SlopeType* are not relevant and therefore remain automatically disabled.

Category	Sub category	Field name (Attribute name)	Range	Field type	Unit type	Block rules	Rule	Order		Delivery	
								Optional - Optional / Mandatory in text	Optional - Optional / Mandatory in text	Unit	Value
Test Equipment Parameters	Admin Info	Identification		String	Dimensionless			optional	optional	-	
	Measurement	MotionDirection	ValueList.Name.Generic.GenericMotionDirection	String	Dimensionless			mandatory	mandatory	-	Translational
Measurement	ProgramType	ValueList.Name.Generic.GenericProgramType		String	Dimensionless			mandatory	mandatory	-	Periodic
	VariedQuantity	ValueList.Name.Generic.GenericVariedQuantity		String	Dimensionless		TstSq.N	mandatory	mandatory	-	
	SlopeType	ValueList.Name.Generic.GenericSlopeType		String	Dimensionless		TstSq.N	mandatory	mandatory	-	
	Symmetry	ValueList.Name.Generic.GenericSymmetry		String	Dimensionless		(TstSq.PreConditioning)	mandatory	mandatory	-	

Figure 3-20: Subcategory "Measurement" with associated attributes and sample content

As the test involves sinusoidal excitation, additional attribute values must be entered in subcategory *TPmOscillatExcitMS*. As the max. test speed is specified, attribute *VelocityType* is assigned value "speed". Attribute *VelocityTranslational* is assigned value "10", and the unit is "mm/min".

Attribute *SsignalShape* is set to "sinus". Control element test *ControlType* is assigned value "force", and attributes *ForceStart* and *ForceStop* are assigned value "10" and unit "kN", as the force-controlled measurement must be performed within range +/-10 kN.

Category	Sub category	Field name (Attribute name)	Range	Field type	Unit type	Block rules	Rule	Order		Unit	Value	
								Optional - Optional / Mandatory in text	Optional - Optional / Mandatory in text			
Test Equipment Parameters	Admin Info	Identification		String	Dimensionless			optional	optional	-		
	MeasurementType	MotionDirection	ValueList.Name.Generic.GenericMotionDirection	String	Dimensionless			mandatory	mandatory	-	Translational	
		SpatialDirection	ValueList.Name.Generic.GenericSpatialDirection	String	Dimensionless			mandatory	mandatory	-	X	
TPmOscillatExcitMS		Symmetry	ValueList.Name.Generic.GenericSymmetry		Dimensionless	TstSq.Measurement.ProgramType == 'Periodic' TstSq.Measurement.ProgramType == 'SingleCycle'	TstSq.Measurement.ProgramType == 'SingleCycle'	mandatory				
		Hysteresis	ValueList.Name.Generic.GenericHysteresis	String	Dimensionless		TstSq.Measurement.ProgramType == 'SingleCycle'	mandatory	mandatory			
		VelocityType	ValueList.Name.Generic.GenericVelocityType	String	Dimensionless			TstSq.Measurement.ProgramType == 'Periodic'	mandatory	mandatory	-	Speed
		VelocityTranslation			Decimal	Velocity		(TstSq.Measurement.ProgramType == 'SingleCycle' TstSq.TPmOscillatExcitMS.VelocityType == 'Velocity') && TstSq.MeasurementType.MotionDirection == 'Translational'	mandatory	mandatory	mm/min	10
		VelocityRotation			Decimal	RotationalVelocity		(TstSq.Measurement.ProgramType == 'SingleCycle' TstSq.TPmOscillatExcitMS.VelocityType == 'Velocity') && TstSq.MeasurementType.MotionDirection == 'Rotational'	mandatory	mandatory		
		SignalShape	ValueList.Name.Generic.GenericSignalShape	String	Dimensionless			TstSq.Measurement.ProgramType == 'Periodic' && (TstSq.TPmOscillatExcitMS.VelocityType == 'Velocity' TstSq.TPmOscillatExcitMS.VelocityType == 'Frequency')	optional	mandatory	-	Sinus
		CycleNumber			Integer	Dimensionless		TstSq.Measurement.ProgramType == 'Periodic' && (TstSq.TPmOscillatExcitMS.VelocityType == 'Velocity' TstSq.TPmOscillatExcitMS.VelocityType == 'Frequency')	optional	mandatory		
		ControlType	ValueList.Name	String	Dimensionless			TstSq.TPmOscillatExcitMS.ControlType == 'Displacement' && TstSq.MeasurementType.MotionDirection == 'Translational'	mandatory	mandatory	-	Force
		DisplacementStart			Decimal	Length		TstSq.TPmOscillatExcitMS.ControlType == 'Displacement' && TstSq.MeasurementType.MotionDirection == 'Translational'	mandatory	mandatory		
		ForceStart			Decimal	Force		TstSq.TPmOscillatExcitMS.ControlType == 'Force' && TstSq.MeasurementType.MotionDirection == 'Translational'	mandatory	mandatory	kN	10
ForceStop			Decimal	Force		TstSq.TPmOscillatExcitMS.ControlType == 'Force' && TstSq.MeasurementType.MotionDirection == 'Translational'	mandatory	mandatory	kN	10		
MomentStart			Decimal	Moment		TstSq.TPmOscillatExcitMS.ControlType == 'Moment' && TstSq.MeasurementType.MotionDirection == 'Rotational'	mandatory	mandatory				

Figure 3-21: Subcategory "Test type" with associated attributes and sample content

All information for the above measurement is now predefined by attributes.

3.8.8 Functional data

In the end, the attributes of main category *Function Data* contain the actual values that need to be measured or have been measured. This main category is divided into subcategories *InertiaData* and *CurveForceDisplacementStatic*. Our example concerns a static measurement with force and displacement measurements. These are stored in attributes Displacement and Force. As the measurements are of a vector nature, field types are set to vectors.

Category	Sub category	Field name (Attribute name)	Range	Field type	Unit type	Block rules	Rule	Order	Delivery	Unit	Value
Function Data	InertiaData	Mass		Decimal	Mass			irrelevant	optional	-	
		COGX		Decimal	Length			irrelevant	optional	-	
		COGY		Decimal	Length			irrelevant	optional		
		COGZ		Decimal	Length			irrelevant	optional		
		COGdInertiaX		Decimal	RotationalInertia			irrelevant	optional		
CurveForceDisplacementStatic		Time		Decimal			TstSq.MeasurementType == 'Explicit'				
		Velocity		Vector of Decimal	Velocity			optional	optional		
		Displacement		Vector of Decimal	Length			optional	mandatory	mm	0,0,42,0,73;0,95;1,36;1,77;1,98;2,33;2,52;2,95
		Force		Vector of Decimal	Force			optional	mandatory	N	0,1,10;2,53;3,84;4,99;6,67;8,67;9,68;11,
		Temperature		Vector of Decimal	Temperature			optional	optional		
		Current		Vector of Decimal	ElectricCurrent			optional	mandatory		
		HysteresisLoop		Vector of Integer	Dimensionless			optional	optional		
				Decimal	LocalStiffness			optional			

Figure 3-22: Main category "Functional data" with associated attributes and sample content

3.8.9 Derived characteristic values

The *Derived Characteristic Values* are attributes/values that are not measured directly but calculated based on the actual measurements. In our example, the translational stiffness of the rubber mount in x-direction within a range of +/-0.5 mm is calculated from the measured values as "8563 N/mm".

Category	Sub category	Field name (Attribute name)	Range	Field type	Unit type	Block rules	Rule	Order		Unit	Value	
								Optional - Optional / Mandatory in text	Optional - Optional / Mandatory in text			
Derived Characteristic Values	AnalysisType	AnalysisDirection	ValueList.Name.Generic.GenericAnalysisDirection	String	Dimensionless			optional	optional	-	x	
		RangeXMin		Decimal	<mult.>		DeriCharact.AnalysisType.AnalysisDirection == 'X' DeriCharact.AnalysisType.AnalysisDirection == 'XY'	optional	optional	mm	-0.5	
		RangeXMax		Decimal	<mult.>		DeriCharact.AnalysisType.AnalysisDirection == 'X' DeriCharact.AnalysisType.AnalysisDirection == 'XY'	optional	optional	mm	0.5	
		RangeYMin		Decimal	<mult.>		DeriCharact.AnalysisType.AnalysisDirection == 'Y' DeriCharact.AnalysisType.AnalysisDirection == 'XY'	optional	optional	-		
		RangeYMax		Decimal	<mult.>		DeriCharact.AnalysisType.AnalysisDirection == 'Y' DeriCharact.AnalysisType.AnalysisDirection == 'XY'	optional	optional	-		
	Complex	GasForce			Decimal	Force			optional	mandatory		
		FrictionForce			Decimal	Force			optional	mandatory		
		StiffnessTranslation			Decimal	LocalStiffness			optional	optional	N/mm	8563
		StiffnessTranslationUpper			Decimal	LocalStiffness			optional	optional		
		StiffnessTranslationLower			Decimal	LocalStiffness			optional	optional		
		ForceHysteresis			Decimal	Force			optional	optional		
		DisplacementHysteresis			Decimal	Length			optional	optional		
		StiffnessRotation			Decimal	LocalRotationalStiffness			optional	optional		
		StiffnessRotationUpper			Decimal	LocalRotationalStiffness			optional	optional		
		StiffnessRotationLower			Decimal	LocalRotationalStiffness			optional	optional		
		MomentHysteresis			Decimal	Moment			optional	optional		
		AngleHysteresis			Decimal	Planeangle			optional	optional		
		LossAngle			Decimal	Planeangle			optional	optional		
		Stiffness			Decimal	LocalStiffness			optional	optional		

Figure 3-23: Main category "Derived characteristic values" with associated attributes and sample content

4 Technical Basis

This specification allows for the definition of descriptive attributes, derived characteristic values and characteristic curves. It can therefore be used for any type of component (rubber mounts, shock absorbers, tires, etc.) and supports data transfer to databases.

4.1 Steps from technical specification to description in ATFX exchange format

The steps required to translate a technical requirement specification into the ATFX exchange format are shown in Figure 4-1.

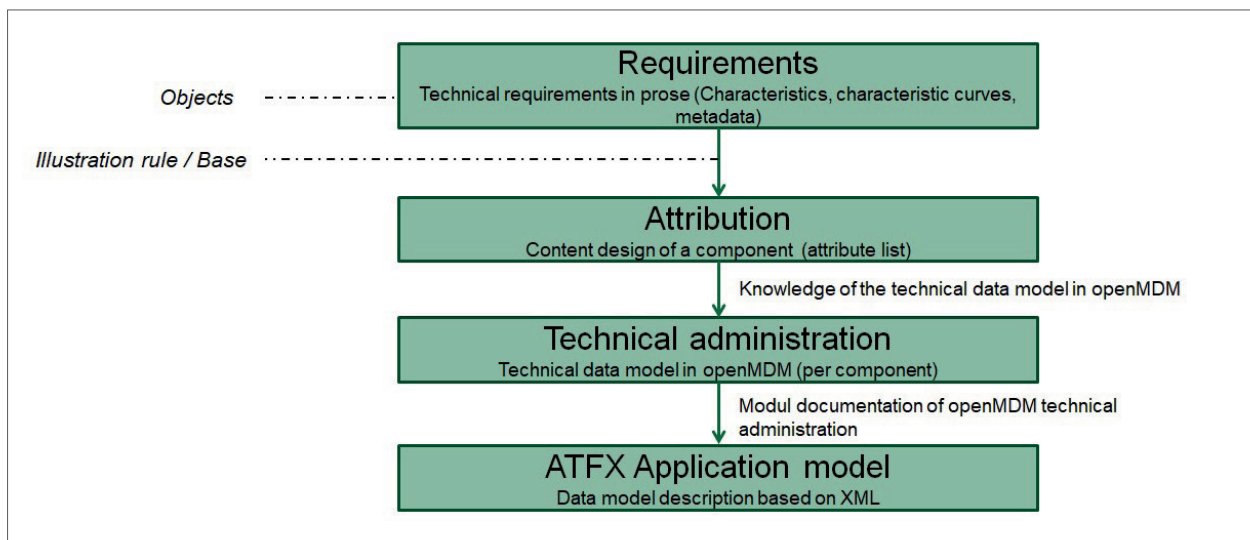


Figure 4-1: Relationship of objects and representation rules

The technical basis of the FDX data exchange format has been devised on the basis of the application concept and the data model. The FDX data format has been developed by the prostep ivip / VDA FDX Working Group, with reference to the ASAM ODS standard, openMDM and openMDM extensions (*Figure 4-2*).

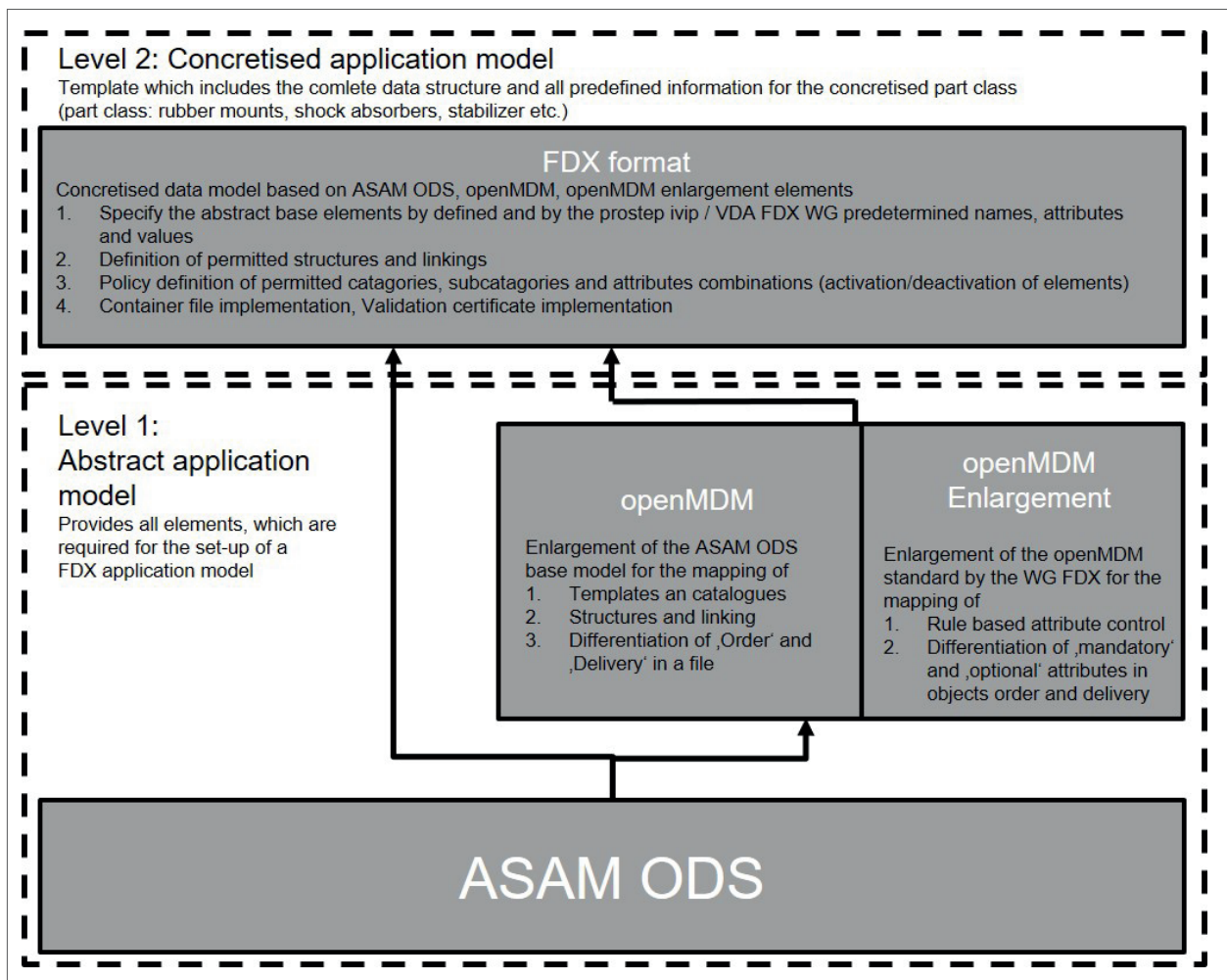


Figure 4-2: Relationship between ASAM ODS, openMDM and FDX (levels 1 and 2)

The application specification consists of three levels. Fig. 4 2 shows levels 1 and 2.

1. Abstract application model: a number of abstract model elements (ASAM ODS base elements and openMDM application elements) that can be used for the specification of concrete application models. This abstract application model is specified in ASAM ODS, openMDM and special openMDM extensions, which have been incorporated into the model by the prostep ivip / VDA FDX Working Group.
2. Concrete application model in FDX format: combination of model elements (application elements) relevant for and applicable to specific component class. These elements are based on the abstract model elements and filled with concrete data including relationship information by the prostep ivip / VDA FDX Working Group. The application models are being developed by specialist project groups within the Working Group to cater from various component classes (rubber mount, shock absorber, stabilizer, etc.) and published in the form of templates (FA-FDX).
3. Instances that apply the application model to a concrete component: depending on the order status, these instances contain the order and test data in the defined data structure. The instances thus contain the data, which is contained in the ATEX file for data request or data supply.

4.2 ASAM ODS

ASAM ODS (Association for Standardization of Automation and Measuring Systems Open Data Services) is a standard on the persistent storage and retrieval of testing data in the wider automotive industry. It governs testing data, meta data (test bench design, parameters) and file attachments. ASAM ODS covers the following aspects:

- Base model and base elements
- Framework for the definition of application models
- File formats (in particular ATFX) for the exchange of data between different systems
- Programming interface for access to data
- Physical storage for long-term archiving

The first three aspects are relevant for this recommendation. Advantages of the adoption of ASAM ODS standard:

- Established format for testing data and similar data types
- Format can be read/written by various existing software systems

4.2.1 ASAM ODS base model

The ASAM ODS data model distinguishes between the base model and the application models. Both types of models describe only the structure of the data to be stored, based on predefined base elements and their relationships. To store actual values, instances of these application elements are generated, and the actual values are stored in these instantiations.

The base model is a general data model for the description of tests and functional data (**Figure 4-3**). The base model defines and describes base elements such as tests, test data, test bench setup and their relationships (basic relationships) that can be stored in electronic format.

Each base element represents one type of information. Example: *AoUnit* is the base element representing a physical unit such as Newton or millimeter; *AoMeasurementQuantity* is the base element representing a measured physical parameter such as force or length.

Each basic relationship represents a relationship of a specified type between two base elements. *AoMeasurementQuantity* refers for example to *AoUnit*, and this relationship indicates which one of the available units is the unit to be used for *AoMeasurementQuantity*. The base model forms the basis for all application models that use ASAM ODS.

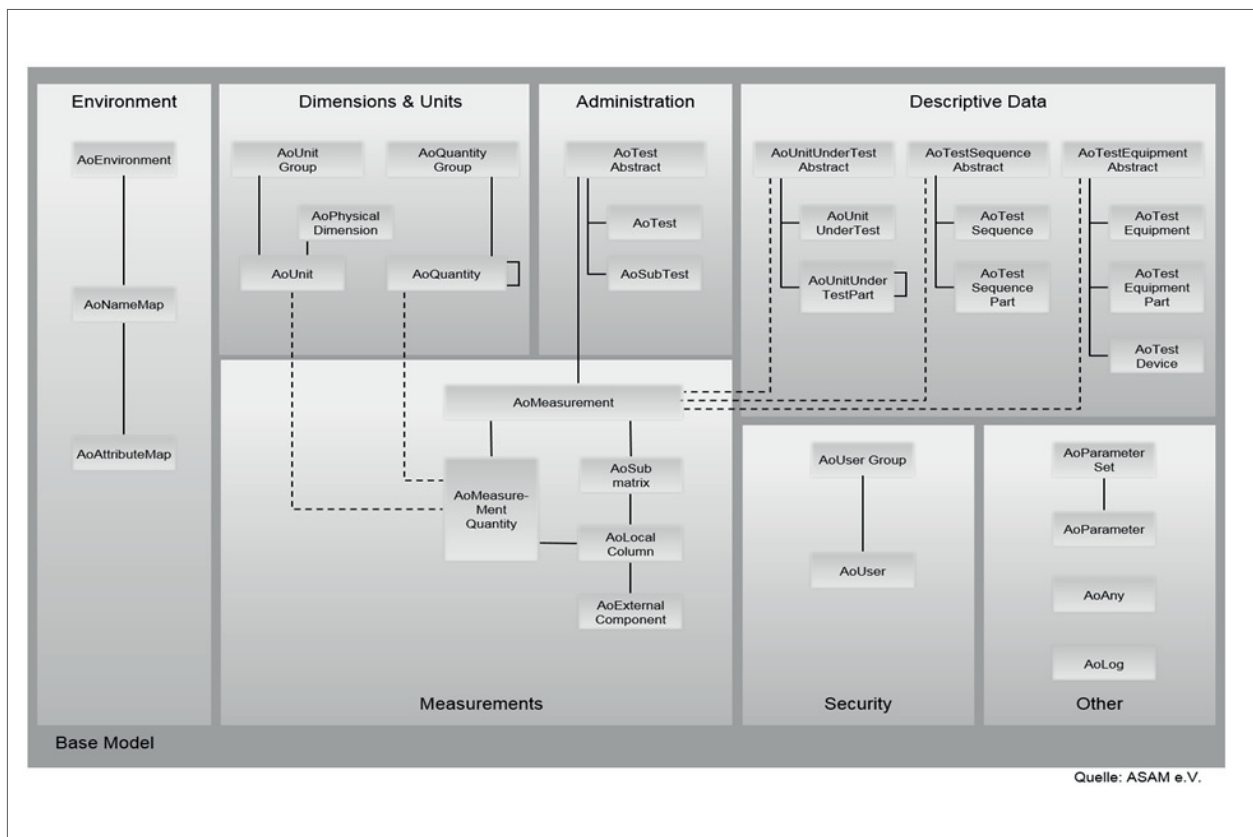


Figure 4-3: ASAM ODS base model

The technical basis of the FDX data exchange format has been devised on the basis of the application concept and the data model. The FDX data format has been developed by the prostep ivip / VDA FDX Working Group, with reference to the ASAM ODS standard, openMDM and openMDM extensions (Figure 4-2).

4.2.2 Application models

Application models cater for specific applications, e.g. the measurement of dampers or rubber mounts. The general elements of the base model are instantiated in the application model. The application model defines the base elements to be used and their names. Every element of an application model refers to an element of the base model, its content and its significance (**Figure 4-4**).

The application model must use the base elements, in order to model the information and its structure for subsequent processing and storage of real data according to the ASAM ODS standard. By creating an application model, a list of application models must be specified, so that each unit of information to be stored can be captured in a single application element. The application elements inherit all mandatory attributes and all basic relationships from the corresponding base elements.

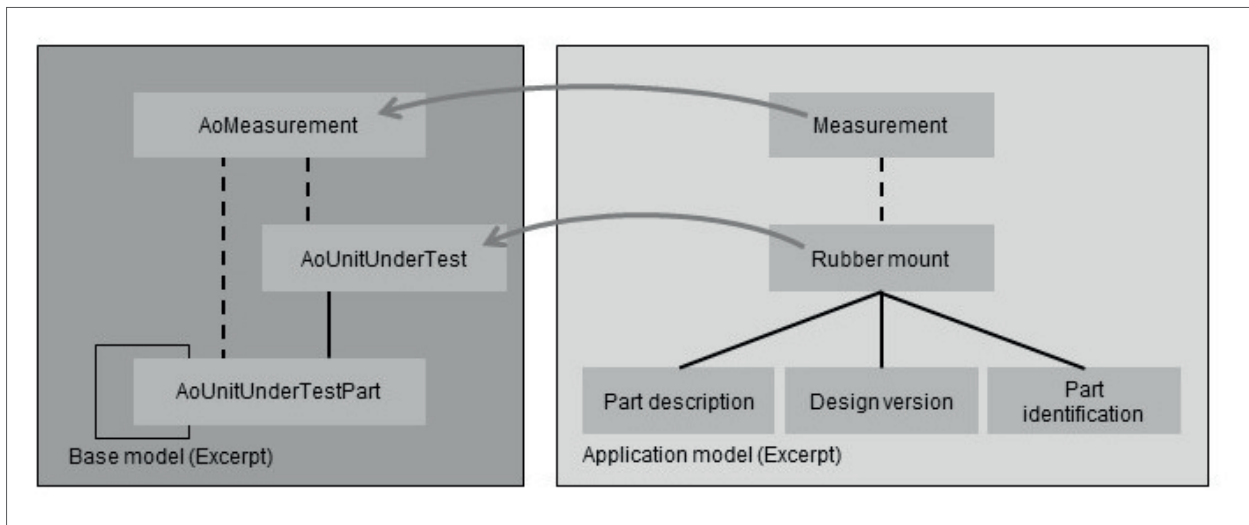


Figure 4-4: Excerpt from an application model and the associated part of the base model

The excerpt from the base model shows base elements *AoMeasurement*, *AoUnitUnderTest*, and *AoUnitUnderTestPart*. The lines between the base elements represent the basic relationships. Continuous lines indicate a direct relationship along the element hierarchy; a dashed line indicates a reference between base elements in different categories (e.g. measurements and descriptive data, see **Figure 4-3**).

Multiplicity of the relationships (not shown in the figure):

- 0..* for all relationships involving *AoMeasurement*.
- One *AoUnitUnderTest* and *AoUnitUnderTestPart* can be assigned 0..* *AoUnitUnderTestPart*.
- One *AoUnitUnderTestPart* is assigned to one *AoUnitUnderTest* or to one *AoUnitUnderTestPart* along the component tree.

The above excerpt from the application model represents a possible structure for the filing of specific details of a rubber mount for the performance of measurements. The designer of this application model has made the following choices:

- Each measurement is called "measurement". Therefore, application element "measurement" is included in his application model. Its basic type is *AoMeasurement*.
- He decided that the focus is to be on the measurement of a rubber mount. That is why application element "rubber mount" is the root elements of his test object description. Its basic type is *AoMeasurement*.
- He further decided that additional information such as a description of the part, its version and a part ID are required for the unique identification of the rubber mount. That is why he included three application elements of type *AoUnitUnderTestPart* in his application model. All relate directly to "rubber mount".
- The application designer decided that each measuring action is to be assigned to the rubber mount in which the measurements were determined. The measurement must therefore include information on the rubber mount on the test bench, and the rubber mount data must include a reference to the actual measurements. In **Figure 4-4**, this is specified in the relationship between *AoMeasurement* and *AoUnitUnderTestAbstract*, which is inherited by *AoUnitUnderTest* and *AoUnitUnderTestPart*, and to the derived application elements "rubber mount", "part description", "version" and "part identification". (In **Figure 4-4**, this is represented by the dotted lines between *AoMeasurement*, *AoUnitUnderTest* and *AoUnitUnderTestPart*.)

There are several ways to devise an application model for the above purpose. By defining a framework for standardization this prostep ivip / VDA Recommendation provides the predefined application models for certain components such as rubber mounts and shock absorbers. These models are not based on the ASAM ODS standard only, but also use the openMDM framework (see chapter 4.2). The openMDM framework complements the ASAM

ODS standard by offering additional functions, thus providing a meta model that can be used for the specification of application models.

4.2.3 ASAM ODS ATFX data exchange format

ATFX (for "ASAM Transport Format / XML") is the XML-based data exchange format used with ASAM ODS. Apart from the actual instance data, ATFX files contain the application model, i.e. a description of the data, types and relationships, so that the instance data can be checked as regards formal and content requirements. To evaluate an ATFX file, only the actual file and the ASAM ODS base model are required. This ensures that future extensions can be made without causing any problems, as changes to the application model are always stored in the file.

4.2.4 More information on ASAM ODS and ATFX

For more information, in particular for the development of own applications, refer to the ASAM website, following the link to the description of the base model (see also chapter 7 "Normative references").

4.3 openMDM

The openMDM® Eclipse Working Group develops open-code concepts and software components for the management of measurement based on the ASAM ODS standard.

4.3.1 openMDM4 standard

This recommendation describes the FDX format based on version 4 of the openMDM framework's datamodel. For the openMDM datamodel, the ASAM ODS base model has been extended, for example to allow for the creation of templates and catalogues. These templates and catalogues can be grouped and structured hierarchically. This approach allows for the representation of complex relationships (hierarchies, reference) through a type of a kit or module system. An overview of the openMDM application model is included in appendix B.

4.3.2 Extensions to and deviations from the openMDM4 datamodel

The FDX format makes specific extensions and changes to the openMDM4 datamodel and data handling to better accommodate its use cases. These include the following:

- Rules for dynamic adjustments of FDX order templates (see appendix C for details)
- Connection between elements Tpl*Attr and ValueList for better handling of FDX order templates
- Addition of attribute ObligatoryResult to Tpl*Attr elements to specify attributes obligatory only for measurements, not the order
- Usage of FDX-specific translation mechanism
- Data handling for descriptive data deviates from openMDM4.
 - o FDX enables different descriptions at TestSteps and each of their MeaResults, which is restricted in openMDM to the same template for all of them.
 - o FDX enables an own description for each MeaResult, whereas openMDM4 demands the same description for each MeaResult under the same TestStep
- While openMDM4 test orders consist only of TestStep descriptions of a test to perform, an FDX test order already contains "measurement data" (results and channels), which are used to describe target or reference curves in the Functional data part. For the order description of these measurements the measurement description is used and also TplParameterSet, TplParameter and TplSensor instances are used to describe the channels of these target or reference measurements.

4.4.3 More information on openMDM base standard

For more information, in particular for the development of own applications, refer to appendix B and the openMDM website (see also chapter 7 "Normative references").

4.4 prostep ivip / VDA FDX extensions of openMDM4 and ASAM ODS

The FDX format uses specific application models based on the options offered by the ASAM ODS base model and the openMDM application model. It complements the standard formats with structured, pre-defined attributes and scopes that are based on the actual purpose, and thus makes them more concrete.

4.4.1 Extensions of openMDM application model

In order to meet two key requirements for the FDX format as described in the objectives for this recommendation, the currently available openMDM4 application model has been extended.

1. On the one hand, there is the requirement that order specifications are to be taken into account by means of the control attributes and rules described in 3.8.1. This concern rules for the enabling/disabling of components (block rules) and for individual attributes (simple rules) as well as complete subcategories.
2. On the other Hand, it must be possible to distinguish between mandatory and optional attributes for order and delivery.

In this context, the term "component" refers to application elements of the openMDM application model (see appendix D). The order templates must not be confused with templates of the openMDM application model, which serve as templates for model components. The order templates are rather specifications of the overall order description whereas the openMDM templates are the technical means to store and handle the details of such a description without too much redundancy in the data.

The extensions made to the openMDM application model are described in detail in appendix C.

The conventions for UML diagrams are described in appendix I.

4.4.2 Relationship between ASAM ODS base model, openMDM application model and prostep ivip / VDA FDX application model – example of rubber mount

The model-driven administrative mechanisms for the instantiation of openMDM application models based on the ASAM ODS base model are described in more detail in appendixes D, E and F.

The technical representation of the data in the ASAM ATFX data exchange format is explained in detail in appendixes G and H.

The conventions for UML diagrams are described in appendix I.

5 Structure of FDX functional data exchange file

The data is exchanged in the form of functional data exchange files that contain the ATFX file and optional reference files. The functional data exchange files are containers that can be equipped with an optional validation certificate file (Figure 5-1).

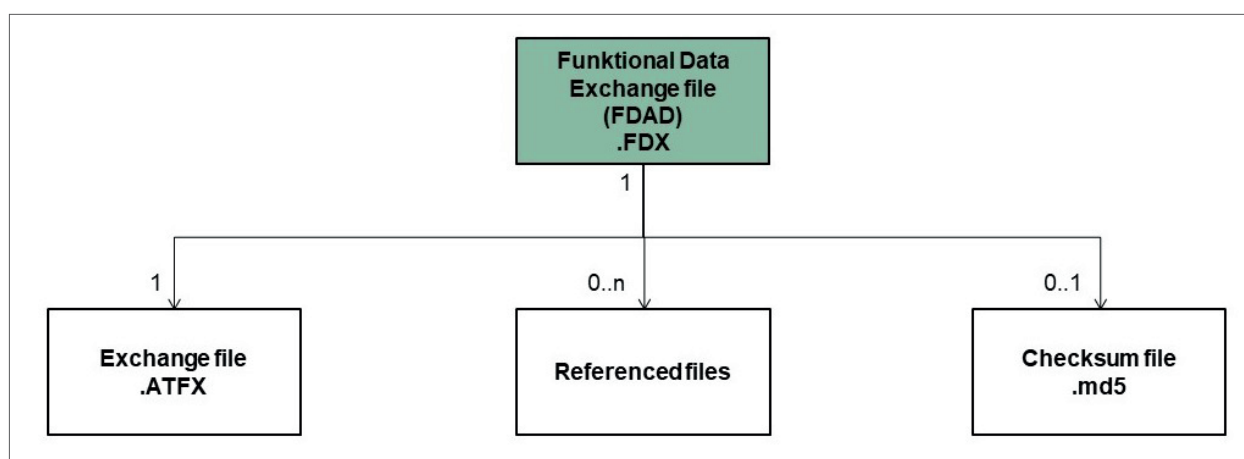


Figure 5-1: Components of FDX exchange format

The functional data exchange file is a container (ZIP) file with extension .fdxc or .fdtc. Extension .fdxc denotes a "normal file", while extension .fdtc is used for templates.

Depending on the data enhancement status of the functional data exchange file, it might include a number of additional files:

- ATFX file with order template, order datasets and result datasets
- Reference files for order template, order datasets and result datasets
- validation certificates

Note:

- Each functional data exchange file contains one ATFX file.
- Each functional data exchange file is assigned a status for every data enhancement stage.
- The existence of reference files and a validation certificate file depend on the data enhancement status and the data model on which the functional data exchange file is based

5.1 Functional data exchange file as template (.fdtc)

A functional data exchange file with data enhancement status "Template" is provided by the application administrator for the creation of new orders. If the application administrator has set up documents for attribute default values, the functional data exchange files of this data enhancement level might already contain files. Initially, functional data exchange files do not necessarily feature a validation certificate, as the application administrator might need to make further changes and the file is therefore only temporarily stored.

5.2 Functional data exchange file as FDX file (.fdxc)

The exchange of functional data between the requester and the supplier is done through the functional data exchange file. It results from the use cases "Data requester - Creating, finalizing and transmitting order", "Data supplier - Receiving, examining and accepting order" and "Data supplier - Entering, finalizing and submitting data" described in chapter 3.

5.3 ATFX file (.atfx)

The ATFX file of the FDX format is based on ASAM ATFX, a XML-based industrial standard that includes a data model for the storage of measurements. The FDX format is self-descriptive, as it contains the requirements, the measuring conditions and the results and that can be evaluated without the need for any additional descriptions. Everything needed to interpret the data in the file is included in the ATFX file, as the application model and its application elements, attributes and values contain all rules for the measurement order for a specific component. The ATFX file therefore contains a complete description for the measurement and evaluation of functional data, and its storage and transfer based on XML.

ATFX is a flexible format that can be configured for any type of component and measuring/simulation method.

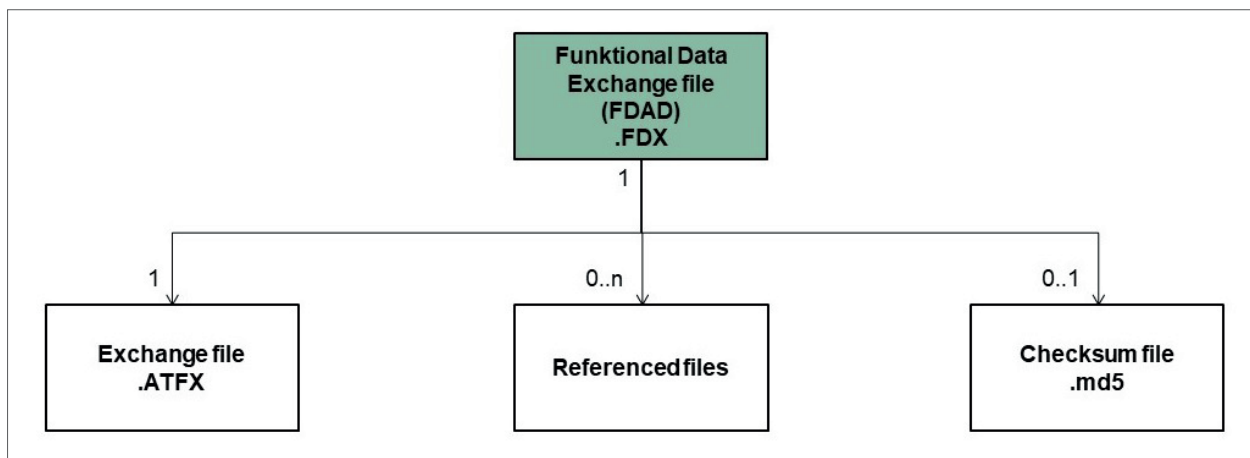


Figure 5-2: Structure of ATFX file

The ATFX file serves as a central file in which categories, attributes and their properties are defined and saved (**Figure 5-2**). ATFX files are created and populated with the relevant information as described in chapter 3 for use cases FA master, FA-FDX and FA-OEM. (**Figure 5-3**).

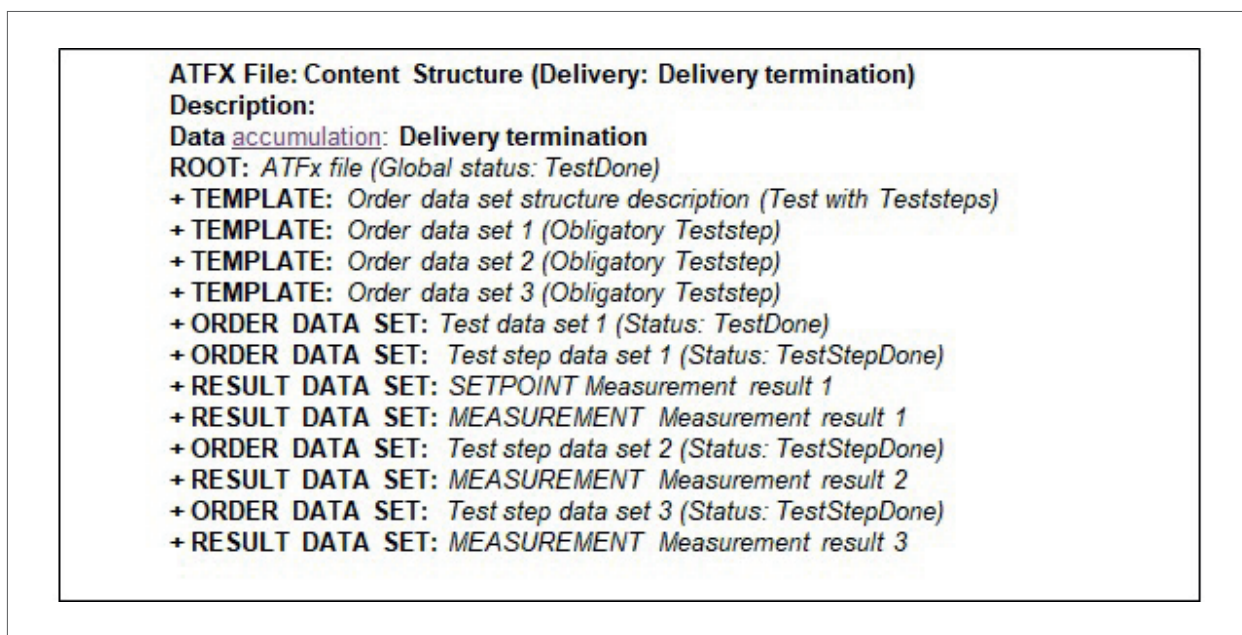


Figure 5-3: Data enrichment in ATFX file

5.4 Referenced files

ATFX files can also contain other documents, specified in attributes of type DT_EXTERNALREFERENCE or DS_EXTERNALREFERENCE. The relative file paths are thereby saved in the attributes in the ATFX file. The specified documents (e.g. pictures, installation layouts, drawings, etc.) are included as referenced documents in the functional data exchange file. The actual measured data should however not be included in the form of a referenced file.

5.5 Validation certificate

The FDX format support the use of validation certificates. The functional data exchange file contains a placeholder for a validation certificate (see **Figure 5-1**). The validation certificate is used to let the partner user know that the functional data exchange file meets his or her requirements. It certifies:

- Compliance with ATFX syntax
- Completeness of data
- Minimum scope of mandatory attributes
- Compliance with prescribed data types
- Compliance with prescribed value ranges

Version 1.0 of prostep ivip / VDA Recommendation 5550 does not include specifications or instructions for the structure, generation and verification of validation certificates.

6 Minimum requirements for data exchange

Product-related data require additional protection. The safe handling and exchange of product data is therefore a major concern.

We wish to make it clear that data exchanged in FDX format (i.e. in a zip container), the ATRX file and the reference files are not encrypted. We therefore strongly recommend following the general data security instructions in this chapter.

6.1 Data security

Product-related data is normally deemed confidential information and must therefore be protected against unauthorized access.

Such data must be stored in a structured and secure format. For local or portable storage device, we recommend encrypting the confidential information, using suitable encryption mechanisms.

6.2 Data exchange

The data is exchanged between contractual partners. This normally involves a manual process in which the files are uploaded by authorized, registered users to a dedicated customer portal, from where they are then downloaded. Secure transfer is normally ensured by https (TLS).

Another option is the use of a secure automated data exchange method that is widely used in the automotive industry:

1. OFTP2 over the Internet: data transmission is secured with minimum TLS; data can be encrypted for desk-to-desk security. This protocol is the most commonly data exchange protocol in the automotive industry.
2. OFTP1 over ENX: use of managed virtual private network of the automotive industry for the secure point-to-point transmission of data.
3. Other VPN solutions, which normally require separate registration and configuration.

In all cases, the exchange of data is recorded by tickets. Files can also be uploaded and downloaded by e-mail. With OFTP, end-to-end response should be used.

The specific requirements must be defined between the parties prior to the project start.

7 Normative references

ASAM ODS

ASAM ODS is a standard for the persistent storage of testing data in the wider automotive industry. It covers testing data, meta data (test bench setup, parameters) and file attachments.

This prostep ivip / VDA Recommendation is based on the ASAM ODS standard.

<https://wiki.asam.net/display/STANDARDS/ASAM+ODS>

openMDM4

openMDM4 is an extension of the ODS standard for the representation of templates and similar files.

This prostep ivip / VDA Recommendation is based on the ASAM ODS standard and the openMDM4 standard.

<http://www.openmdm.org/>

Uniform Resource Identifiers (URI)

Uniform Resource Identifiers (URI): Generic Syntax, T. Berners-Lee, R. Fielding, L. Masinter, IETF RFC 2396, August 1998 <http://www.ietf.org/rfc/rfc2396.txt>

URLs are for instance used to identify namespaces in XML schema definitions (XSD).

UTF-8

ISO 10646 Character Transformation Format

<https://tools.ietf.org/html/rfc3629>

ATFX uses the UTF-8 character set, thus supporting most common languages.

XML 1.0 (Fifth Edition)

Extensible Markup Language (XML) 1.0, Fifth Edition, Tim Bray et al., eds., W3C, 26 November 2008

<http://www.w3.org/TR/REC-xml>

The XML syntax is the syntax used in ATFX files.

XML namespaces

Namespaces in XML 1.0 , Third Edition W3C, Tim Bray et al., eds., 8 December 2009

<http://www.w3.org/TR/REC-xml-names>.

Namespaces are used for providing uniquely named elements and attributes in an XML document of a specific content domain.

XML schema

XML Schema Part 1: Structures, Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn, W3C, 2 May 2001 <http://www.w3.org/TR/xmlschema-1/> • XML Schema Part 2: Datatypes, Paul V. Biron and Ashok Malhotra, eds.,

W3C, 2 May 2001 <http://www.w3.org/TR/xmlschema-2/>

XML schemas describe the vocabulary (elements and attributes) and the structure of XML documents (instances).

They are used to validate the syntax of XML documents.

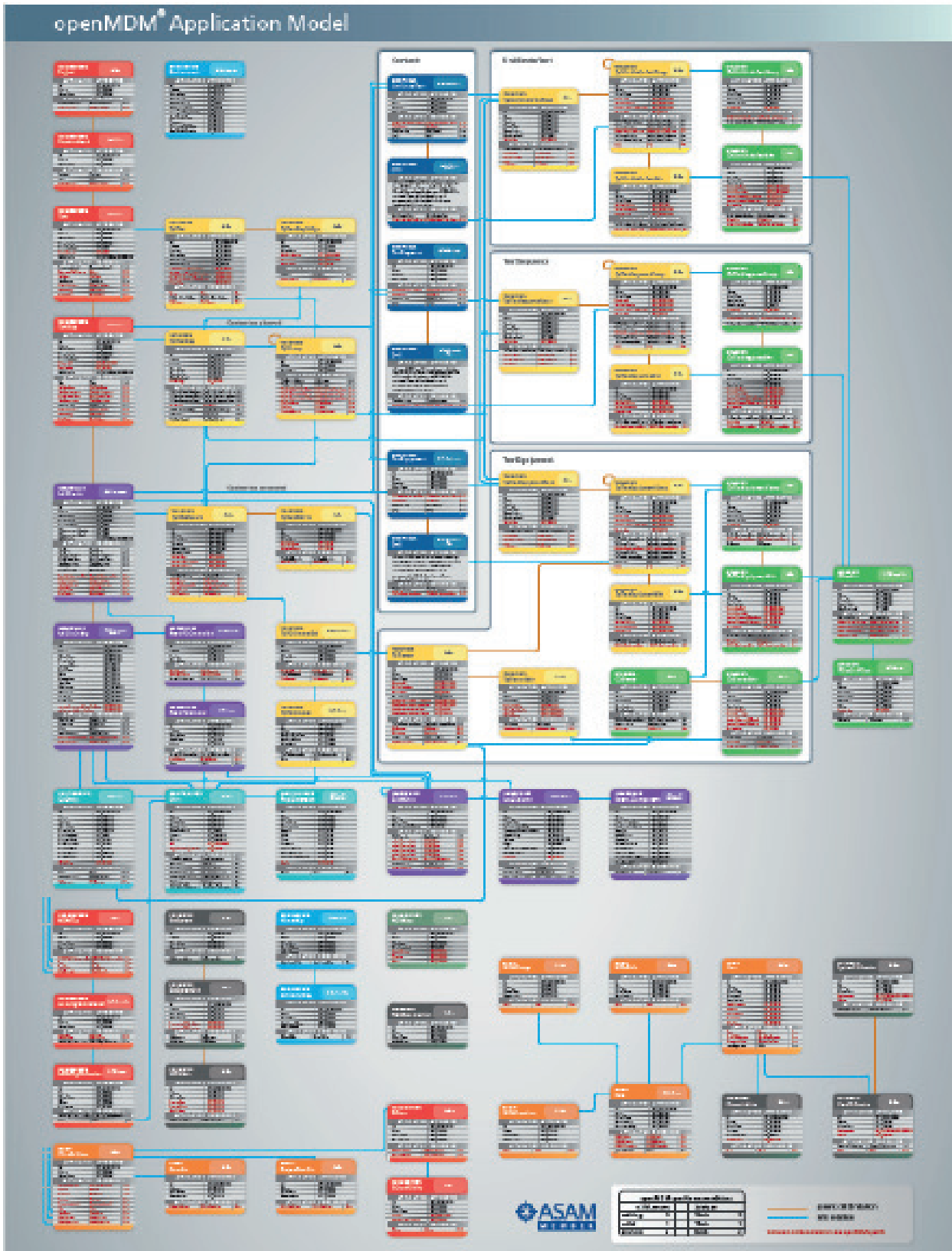
8 Appendix

8.1 Appendix A: Terms and definitions

Term	Definition / explanation
Application model	The application model specifies the elements from the base model that are used in an actual application for a clearly defined task.
ASAM	Association for Standardization of Automation and Measuring Systems
ASAM ODS	ASAM ODS (Open Data Services) is a standard focused on the persistent storage and retrieval of testing data in the wider automotive industry. It covers testing data, meta data (test bench setup, parameters) and file attachments.
ATFX file	ASAM transport format in XML: the ATFX file is the actual XML file used for the exchange of data (measurement order, measurement data, etc.).
Attribute	An attribute is a defined property of an object to be tested. The testing data describe the actual attributes such the property of a physical or virtual object.
Base model	Set of defined base elements and the relationships between them.
Operating parameter	Here: a concrete physical parameter used in a specific test of an object (e.g. temperature, rotary speed, etc.).
Data requester	Partner requesting the test / testing data.
Data supplier	Partner performing the test / sending the testing data.
Data model	Model of the data and semantic structures of information that is stored in exchange files (instances) or recorded in a database.
D-U-N-S number	Unique ID number, issued by Dun & Bradstreet, assigned to a business or a business location.
Application administrator	Person responsible for the development, management and publication of template specifications for standardized measurement orders of specific assemblies.
FA master	The application administration ('Fach-Administration') master is a template specification for standardized measurement orders for all currently defined assemblies.
FA-OEM	The OEM application administration is a template specification for component-specific measurement orders. It is based on the FA-FDX template and contains OEM-specific data. FA-OEM files are used as the working templates for data exchange between the OEM and the supplier.
FA-FDX	The FDX application administration is a template specification for component-specific measurement orders. It has been devised by the prostep ivip / VDA FDX Working Group
FDAD	Functional data exchange file: Container format with ATFX file, and additional reference files and validation certificate file; file extension: .fdx
.fdtc file	Functional data exchange file template created by the application administrator and made available to the data requester and the data supplier completion. By using the template, i.e. through editing by data requester and filling by data supplier, the .fdtc file becomes a .fdxc file (.fdtc = template; .fdxc = completed data file).

Term	Definition / explanation
.fdxc file	Normal functional data exchange file used for the exchange of information between data requesters and data suppliers (.fdtc = template; .fdxc = completed data file).
Functional data exchange file template	A functional data exchange file template is created by the application administrator in order to trigger concrete measurement orders for specific components or parts.
FDX	Functional data exchange; file format; name of prostep ivip / VDA Working Group.
Functional data	All data captured by measurement, estimation or calculation under defined conditions and according to a measurement order for a specific (real or virtual) unit under test.
Catalogue component	Aggregation of description attributes that form a functional unit and can/must be re-used in various application models.
Array of characteristic curves	Multiple characteristic curves forming a family of characteristic curves or displayed in a three-dimensional coordinate system, based on additional input parameters.
Characteristic curve	Chart depicting two interdependent physical parameters in the form of a curve.
Measurement order	Order for the performance of tests/measurements on an object; includes the parameters to be tested/measured and specifications regarding the testing/measurement equipment.
OMG	Object Management Group; international, open membership, not-for-profit technology standards consortium for the modelling and integration of software systems. One of its standards concerns the Unified Modelling Language® (UML).
openMDM®4	A collection of components and concepts designed for the development applications for testing data management. Devised by the openMDM® Eclipse Working Group.
XHTML	XML-conforming version of Hypertext Markup Language (HTML).
XML document	Data structure consisting of one or more files that share a common content.

8.2 Appendix B: openMDM application model



8.3 Appendix C: Extension of openMDM application with rules; including an extension that allows for the distinction between mandatory and optional attributes in the order files

This appendix is an addendum to chapter "Extensions of openMDM application model" of prostep ivip / VDA Recommendation PSI 20.

References to other appendices

As seen in the class diagram in **Figure 8-1**, the current openMDM4 application model is extended with the class *Rule* of ASAM ODS base type *AoAny*, and the existing classes *TplUnitUnderTestAttr*, *TplTestSequenceAttr* and *TplTestEquipmentAttr* contain each the new attribute *ObligatoryResult*:

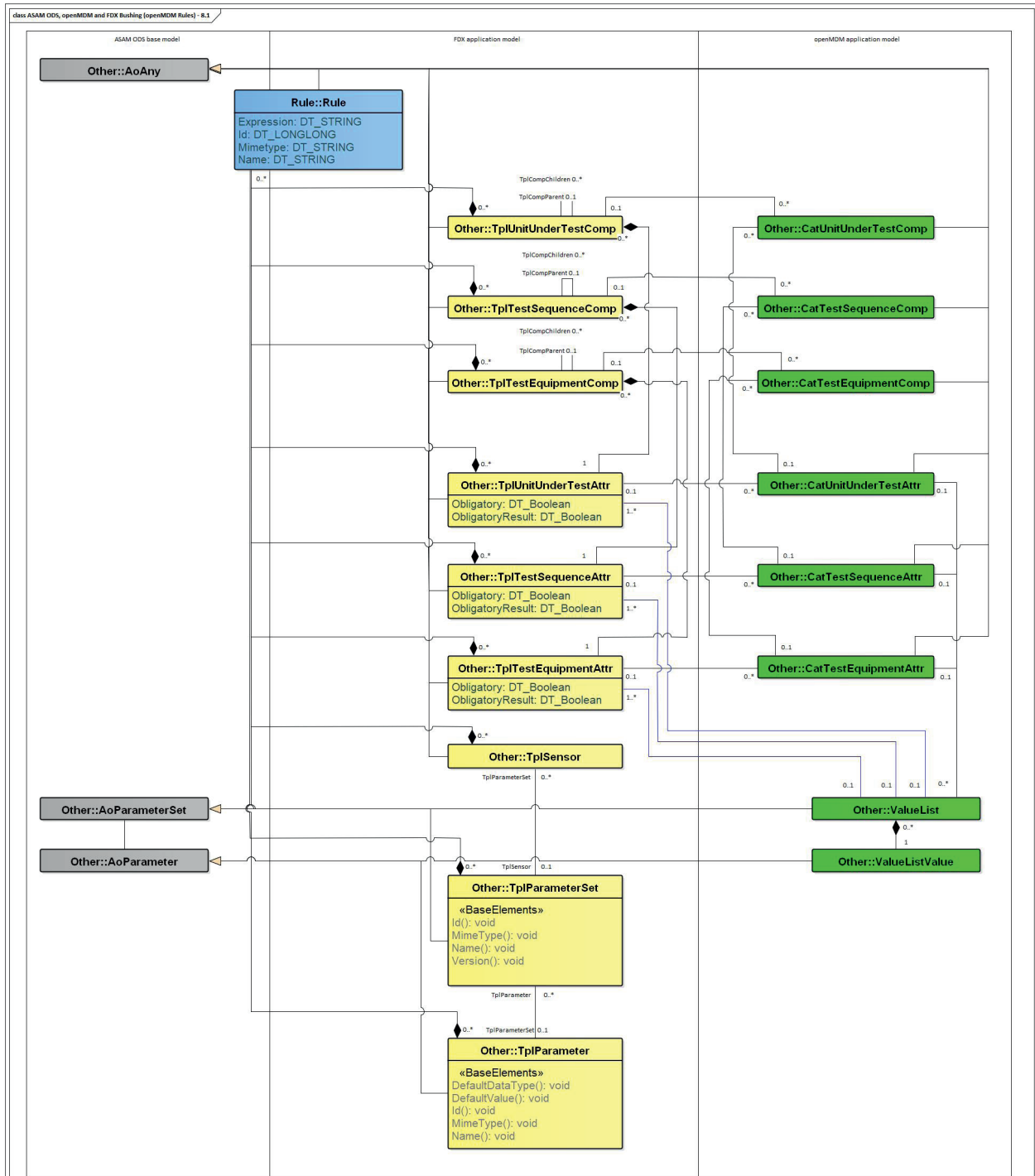


Figure 8-1: prostep ivip / VDA FDX modifications to openMDM4 application model

Addition of rules to dynamically adjusted template

- Application element Rule has been added to the model.
- A Rule instance specifies an expression, defined in Java-EL (Expression Language).
- The expression contains the specification of the control attribute/s and its/their value/s, which is/are triggering the action of this rule.
- The mimetype of the rule specifies the action connected with this rule. Currently the following mimetypes and actions are specified, but more could be added in the future:
 - o application/x-asam.aoany.rule.active

- o A rule with this mimetype causes the target attribute/s to be set active in the data description if the expression evaluates to true and their values have to be filled. If the expression evaluates to false the target attribute/s is/are removed from the description.

- o application/x-asam.aoany.rule.mandatory

- o A rule with this mimetype causes the target attribute's/' values to be mandatory to fill in the data description if the expression evaluates to true. If the expression evaluates to false the value/s is/are optional to fill.

It is possible to devise block rules for components (relations to Tpl...Comp) and simple rules for attributes (relations to Tpl...Attr):

- A block rule makes sure that, upon selection of a specific attribute value, components (with all their attributes) related to this attribute by that rule are affected.
- A simple rule makes sure that, upon selection of a specific attribute value, attributes of the same component that are related to this attribute by that rule are affected.

Rules can also be connected to other elements of the openMDM datamodel:

- TplSensor
- TplResultParameter
- TplResultParameterSet.

Example for an "active" block rule:

With reference to the lines marked with "2" and "3" in **Figure 3-12**, if attribute *TestType.MeasurementWithpreload* is set to "yes", the attributes of subcategory *Preload* must be assigned values. With value "no", subcategory *Preload* is not valid. This is a so-called block rule, since it is applied to all the attributes of the component *Preload*.

To define this, a *Rule* instance with mimetype "...active" is created and connected to the application element *TplTestSequenceComp*, which in turn refers to application element *CatTestSequenceComp* that corresponds to the class of application element *Preload*. The expression of the rule is *TestingEquipmentParameters.MeasurementType.Preload=='Yes'*, checking whether that attribute's value is "Yes". If the condition is fulfilled, the specific application model must contain application element *Preload* as a component of application element *TestSequence*.

Example for a "mandatory" simple rule:

With reference to the line marked with "4" in **Figure 3-12**, and under condition that attribute *Preload.ControlType* is assigned value "force-controlled", and attribute *TestingEquipmentParameters.Preload.DirectionOfMovement* is set to "translational", attribute *Preload.Force* would become a mandatory attribute while attribute *Preload.Torque* would not.

To define this, a *Rule* instance with mimetype "...mandatory" is created and connected to the application element *TplTestSequenceAttr*, which in turn refers to application element *CatTestSequenceAttr* that corresponds to the attribute *Preload.Force*. The expression of the rule specifies two attributes to check for specific values and if both checks evaluate to true, the target attribute is set to mandatory. According rules connected to the other attributes (*Displacement*, *Angle*, *Torque*) will make them optional in turn.

Distinction between mandatory and optional attributes for order and data delivery

The current openMDM4 application model does not support modelling of different mandatory attributes for order data and testing data in one order template. To make such a distinction, it has up to now been necessary to set up multiple order templates.

The following extension will make this obsolete:

- Application elements *TplUnitUnderTestAttr*, *TplTestSequenceAttr* and *TplTestEquipmentAttr* are extended by attribute *ObligatoryResult*. This value determines whether an attribute for testing data is a mandatory or an optional attribute. The previously defined attribute *Obligatory* is therefore only valid for order data. With this extension, a distinction is made between the order context and the measurement context.

8.4 Appendix D: Relationship between ASAM ODS base model, openMDM application model and prostep ivip / VDA FDX application model - example of rubber mount

This appendix is an addendum to chapter "Relationship between ASAM ODS base model, openMDM application model and prostep ivip / VDA FDX application model - example of rubber mount" of prostep ivip / VDA Recommendation PSI 20.

The class diagram in Figure 8 2 shows the relationship between the ASAM ODS base model, the openMDM application model and the prostep ivip / VDA FDX application model, using the example of a rubber mount. It contains a selection of important classes from the ASAM ODS base model and from the openMDM application model as well as from the classes of the prostep ivip / VDA FDX application model for rubber mounts. The classes shown in the class diagram represent the selected sections of the order, and of the data delivery for measurement orders. The vertical arrangement from top to bottom corresponds roughly to the sequence in which the data is submitted, i.e. the sequence in which the data requester and the data supplier fill the attributes with values.

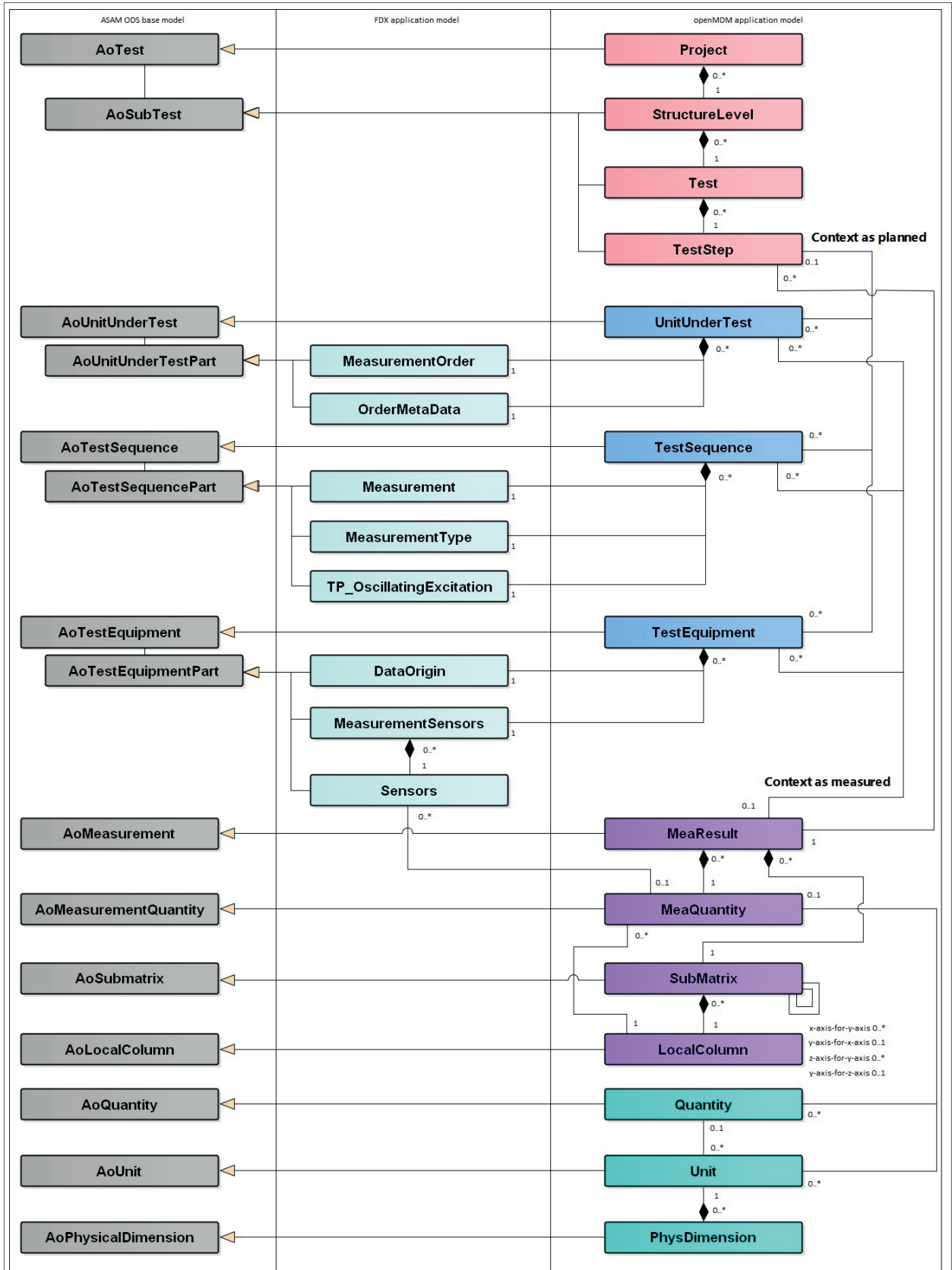



Figure 8-2: Relationship between ASAM ODS base model, openMDM application model and prostep ivip / VDA FDX application model - example of rubber mount (overview)

Organization of measurement orders by means of application elements *Project*, *StructureLevel*, *Test*, *TestStep*

Classes *Project*, *StructureLevel*, *Test* and *TestStep* allow for the organization of measurement orders and have the following associations:






- One ATFX file contains 0..* application elements *Project* of ASAM ODS base type *AoTest*, which form the root for the organization of measurement orders (application elements shown in red ). An ATFX file with 0 application elements *Project* therefore does not contain any measurement orders.
- An application element *Project* contains 0..* application elements *StructureLevel* of ASAM ODS base type *AoSubTest*. Application elements *Project* and *StructureLevel* organize the test project.
- An application element *StructureLevel* contains 0..* application elements *Test* of ASAM ODS base type *AoSubTest*. Application element *Test* represents the entire test project.
- An application element *Test* contains 0..* application elements *TestStep* of ASAM ODS base type *AoSubTest*. Application element *TestStep* represents one *test step* of a test project.

The definition and use of application elements *Project* and *StructureLevel* are not covered in this recommendation, as the prostep ivip / VDA FDX data model does not make specific reference to them. If required, the data requester and the data supplier might need to define the respective rules between them. Otherwise, these application elements are not taken into account, and all application elements *TestStep* that are accessed through application elements *Project* and *StructureLevel* must be accessed as described below.

Class *TestStep* has associations with classes *UnitUnderTest*, *TestSequence* and *TestEquipment*. These associations correspond to the associations labelled with "Context as planned" in the openMDM overview and are therefore relevant for the order.








Data for data request and delivery in application elements *UnitUnderTest*, *TestSequence*, *TestEquipment*

Through their components, classes *UnitUnderTest*, *TestSequence*, *TestEquipment* specify all order-related information, in particular the unit under test, the test equipment and the test sequence.

- An application element *TestStep* refers to 0..1 application element *UnitUnderTest* of ASAM ODS base type *AoUnitUnderTest*, to 0..1 application element *TestSequence* of ASAM ODS base type *AoTestSequence* and to 0..1 application element *TestEquipment* of ASAM ODS base type *AoTestEquipment* (shown in blue ). Application element *TestStep* therefore provides the link between the unit under test, the test sequence and the test method.
- An application element *UnitUnderTest* contains 0..* components of ASAM ODS base type *AoUnitUnderTestPart* that are valid for the specific openMDM application model (rubber mount, shock absorber, etc.). For the rubber mount, this is shown in **Figure 8-2** with the classes *MeasurementOrder* and *OrderMetaData* (excerpt, shown in light blue ). All components of *UnitUnderTest* and their functions valid for the rubber mount are listed in **Figure 8-3**.
- Application element *TestSequence* contains 0..* components of ASAM ODS base type *AoTestSequencePart* that are valid for the specific openMDM application model. For the rubber mount, this is shown in **Figure 8-2** with the classes *Measurement*, *MeasurementType* and *TP_OscillatingExcitation* (excerpt, shown in light blue ). All components of *TestSequence* and their functions valid for the rubber mount are listed in **Figure 8-6**.
- The application element of class *TP_OscillatingExcitation* stands for the concrete Measuring program with oscillating excitation.
- An application element *TestEquipment* contains 0..* components of ASAM ODS base type *AoTestEquipmentPart* that are valid for the specific openMDM application model. For the rubber mount, this is shown in **Figure 8-2** with the classes *DataOrigin* and *MeasurementSensors* (excerpt, shown in light blue ). All components of *TestEquipment* and their functions valid for the rubber mount are listed in **Figure 8-9**.
- An application element *MeasurementSensors* contains 0..* components *Sensors* of ASAM ODS base type *AoTestEquipmentPart*, whereby class *MeasurementSensors* represents the measuring program and class *Sensors* represents a measurement along the measuring curve (shown in light blue .

Functional data for data request and data delivery in application elements *MeaResult*, *MeaQuantity*, *SubMatrix* and *LocalColumn*

Classes *MeaResult*, *MeaQuantity*, *SubMatrix* and *LocalColumn* specify the measuring data:

- An application element *TestStep* contains 0..* application elements *MeaResult* of ASAM ODS base type *AoMeasurement*, whereby class *MeaResult* represents an entire measuring result (shown in purple ).
- An application element *MeaResult* contains 0..* application elements *MeaQuantity* of ASAM ODS base type *AoMeasurementQuantity*, whereby class *MeaQuantity* represents one specific value of the measuring result only - *MeaQuantity* does not contain any measurements (shown in purple ).
- An application element *MeaQuantity* refers to 0..1 specific application element *Sensors* (time, speed, displacement, force, angle, frequency, etc.) used or to be used for the measurement. In *Sensors*, the sensor type as well as the axis along which the measured value is to be represented (x, y) are defined.
- An application element *MeaQuantity* refers to 0..1 specific application element *Quantity* of ASAM ODS base type *AoQuantity*, specifying the measured parameter (time, speed, displacement, force, frequency, etc.) (shown in turquoise ). In *Quantity*, the measured value as well as the associated data type (e.g. floating) are defined.
- An application element *MeaQuantity* refers to 0..1 specific application element *Unit* of ASAM ODS base type *AoUnit*, specifying the unit of the *measurement* (Hz, kHz, N, kN, m, mm, m/s, m/s² (shown in turquoise ).
- An application element *Unit* refers to 1 specific application element *PhysDimension* of ASAM ODS base type *AoPhysicalDimension*, specifying the physical dimension to be measured (time, speed, distance, force, frequency, no dimension, etc.) (shown in turquoise ).
- An application element *MeaResult* contains 0..* application elements *SubMatrix* of ASAM ODS base type *AoSubmatrix*, structuring the measurements (shown in purple ). This allows for the representation of a multidimensional measuring result through various parameters, e.g. where period excitation occurs in both x and y direction.
- An application element *SubMatrix* contains 0..* application elements *LocalColumn* of ASAM ODS base type *AoLocalColumn*, whereby an application element *LocalColumn* contains the actual measured data of a specific parameter (shown in purple ), as *LocalColumn* refer to exactly one application element *MeaQuantity*.
- Application element *SubMatrix* also has two reflexive associations, i.e. associations with itself. This allows for the further structuring of the measurement, for instance in a multi-dimensional space. This feature is however outside the scope of this document.

Note: The up-to-date overview of the openMDM application model included in appendix B explains the relationship between the characteristic curves and the associated measurements. These relationships are reflected in the classes *MeasurementSensors*, *Sensors* and *MeaQuantity* and their associations.

Table 8-1 shows the specific information for the 5 application elements *Sensors*, *MeaQuantity*, *Quantity*, *Unit* and *PhysDimension* for the *static force-displacement characteristic curve measuring program* for a rubber mount. For this measuring program, five sensor elements are used: a time sensor, a sensor counting the hysteresis loops, a force sensor, a displacement sensor and a speed sensor. There are thus 5 sensor instantiations. Through the corresponding 5 instances of *MeaQuantity* and their relationships, 5 data types (5 instance of *Quantity*), 5 units (5 instances of *Unit*) and the physical dimensions (5 instances of *PhysDimension* (through *Unit* for the definition of the measurement (request) and the actual measurement (delivery) are assigned to the sensors.

NO.	Sensors	MeaQuantity	Quantity	Unit	PhysDimension
1	=> t	=> t	=> t	s	time
2	=> HysteresisLoopNumber	=> HysteresisLoopNumber	=> HysteresisLoopNumber	-	dimensionless
3	=> F	=> F	=> F	N	force
4	=> u	=> u	=> u	mm	length
5	=> v	=> v	=> v	m/s	velocity

Table 8-1: Application elements Sensors, MeaQuantity, Quantity, Unit and PhysDimension for static force-displacement characteristic curve (CurveForceDisplacementStatic) measuring program for rubber mount

The associations shown in **Figure 8-2** between classes *MeaResult* and *UnitUnderTest*, *TestSequence* and *TestEquipment* correspond to the associations labelled "Context as measured" in the openMDM overview (see appendix), matching those in the data delivery.

These associations specify the actual unit under test (*UnitUnderTest*), the actually performed measuring sequence and the actually used test program (*TestSequence*), the actual test method with the actually recorded characteristic curve (*TestEquipment*) as well as the current dataset. For the subordinate components of *UnitUnderTest*, *TestSequence* and *TestEquipment*, the same rules as described above for the order apply.

Should there be a discrepancy between the data delivery and the data request, application elements *UnitUnderTest*, *TestSequence* and *TestEquipment* must be set up, even if only one of their subordinate components is changed.

For a more detailed discussion of "Relationship between ASAM ODS base model, openMDM application model and prostep ivip / VDA FDX application model - example of rubber mount", refer to *Appendix E: Components of application elements UnitUnderTest, TestSequence and TestEquipment and their relationship with categories of the prostep ivip / VDA FDX application model* and *Appendix F: Model-driven aspects of openMDM application model*.

For more technical details regarding "Relationship between ASAM ODS base model, openMDM application model and prostep ivip / VDA FDX application model - example of rubber mount", see *Appendix G: Representation of model-driven aspects of openMDM4 application model in ATFX* and *Appendix H: XML schema of ASAM ATFX data exchange format*.

8.5 Appendix E: Components of application elements *UnitUnderTest*, *TestSequence* and *TestEquipment* and their relationship with categories of the prostep ivip / VDA FDX application model

This appendix is an addendum to chapter "Relationship between ASAM ODS base model, openMDM application model and prostep ivip / VDA FDX application model - example of rubber mount" of prostep ivip / VDA Recommendation PSI 20.

Components of application element *UnitUnderTest*

The class diagram in **Figure 8-3** shows all components of application element *UnitUnderTest* that are valid for the rubber mount. These components represent all instantiations of ASAM ODS base type *AoUnitUnderTestPart*.

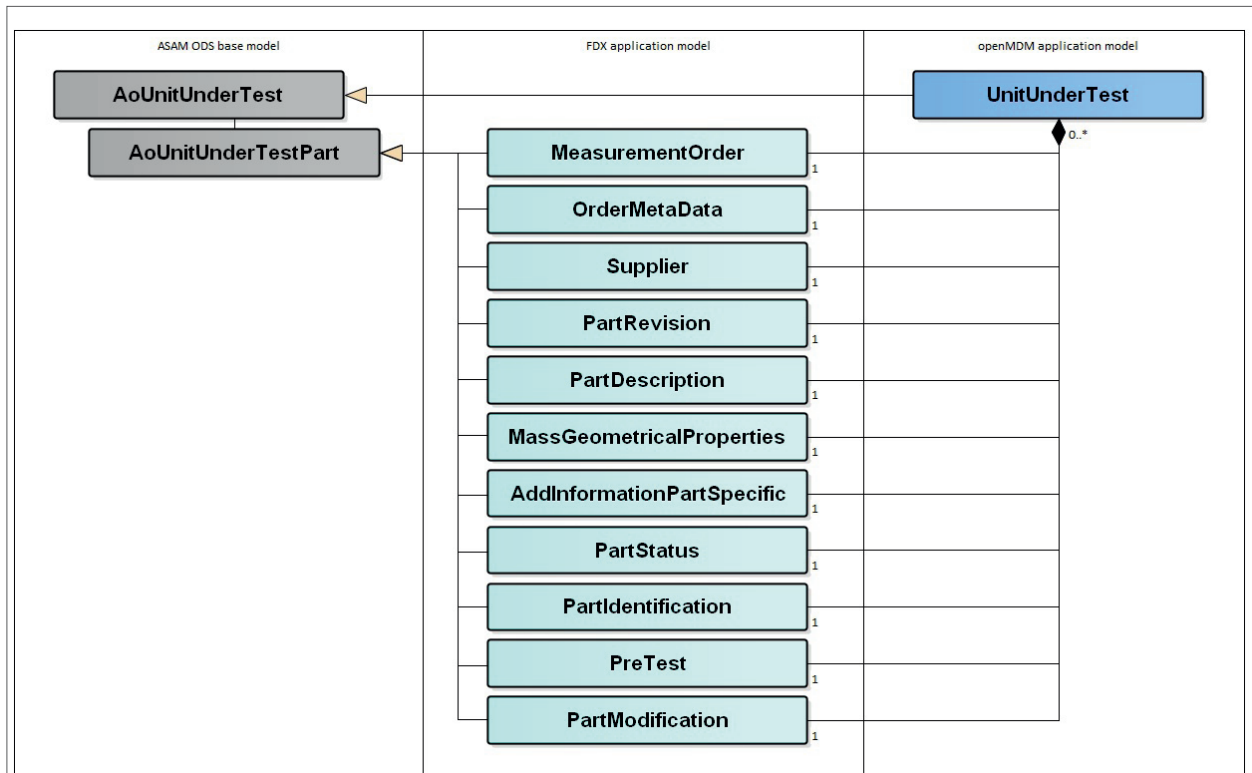


Figure 8-3: Valid components of application element *UnitUnderTest* for rubber mount (example)

Through its components, class *UnitUnderTest* contains information about the main categories measurement order, order meta data, part-specific additional information and unit under test. The class diagram in **Figure 8-4** shows the main categories and subcategories of the prostep ivip / VDA FDX data model (shown in beige) in comparison to the components of *UnitUnderTest* in the openMDM application model (shown in light blue). Dotted lines link a main category or subcategory of the prostep ivip / VDA FDX data model (see chapter 3.8) to the corresponding component of application element *UnitUnderTest* in the openMDM application model. The diagram also shows that there is no hierarchical structure with main categories and subcategories in the openMDM model.

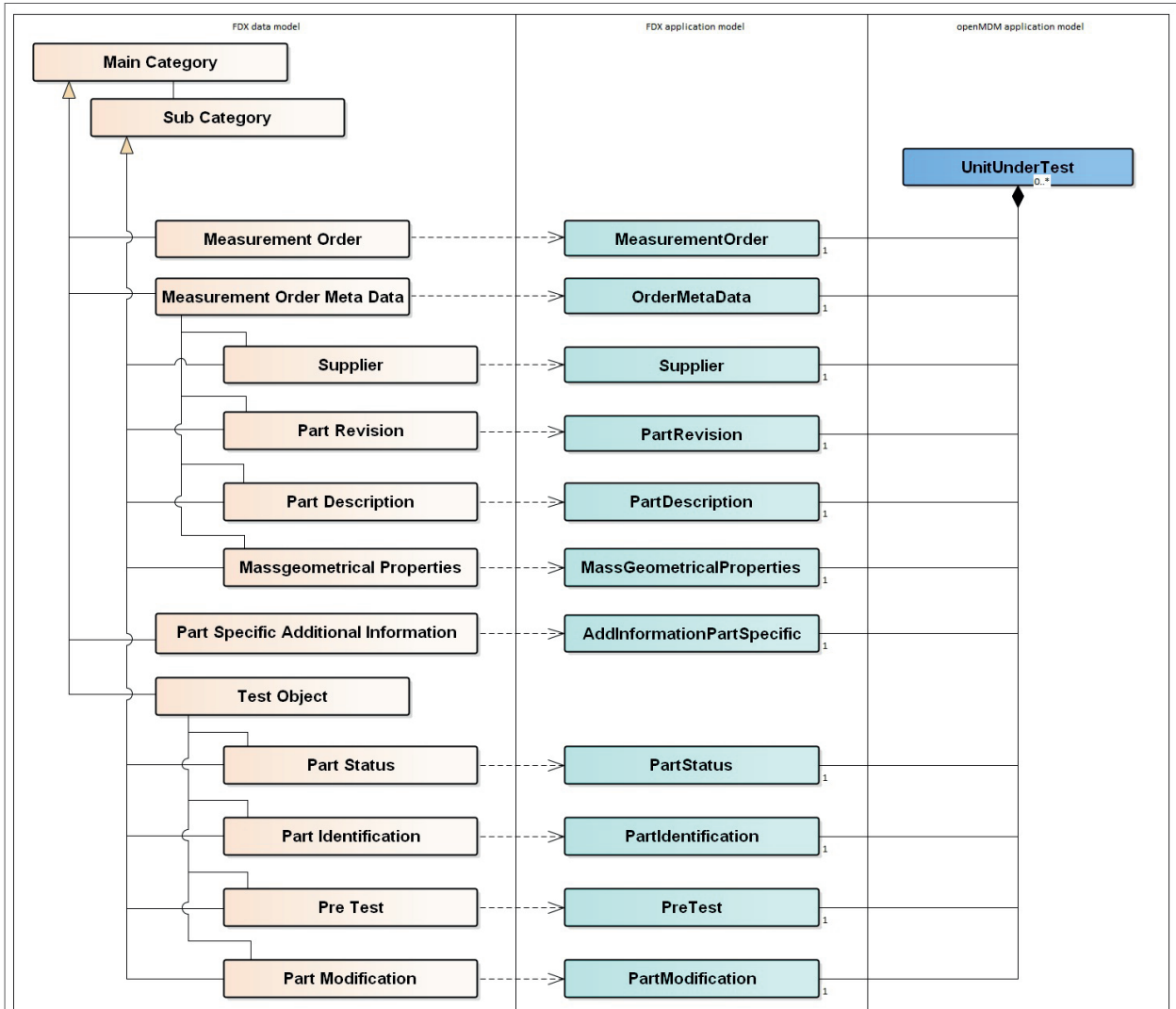


Figure 8-4: Relationships between main categories and subcategories of prostep ivip / VDA FDX data model and components of application element UnitUnderTest in prostep ivip / VDA FDX application model - example of rubber mount

The attributes of the main categories and subcategories are also found in the application elements of the openMDM application model. In the class diagram in **Figure 8-5**, for subcategory *Measurement order*, represented by FDX application element *MeasurementOrder*, these are for instance the attributes *CustomerIdentificationNumber*, *OrderNumber*, *RequesterName*, *Splittability* and *SubOrderNumber*.

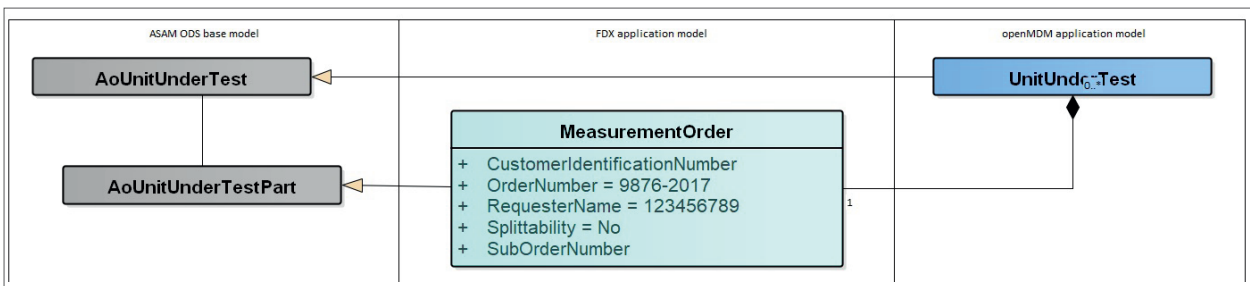


Figure 8-5: Attributes of application element MeasurementOrder and possible values

Note: In **Figure 8-5**, the possible values that application element *MeasurementOrder* are interpreted as default values in UML. While this notation does not strictly conform to UML, it has been chosen to simplify matters.

mes within the same class (multiple instantiations of one class). This happens for example if Precondition is selected, with the same test program as the actual *Measurement*. To distinguish between these two test programs, which are both under the same *TestStep* or *TestSequence*, it is necessary to label them accordingly in attributes *Measurement.TestProgram* or *PreCondition.TestProgram* respectively.

Components of application element *TestSequence*

The class diagram in **Figure 8-6** shows all components of application element *TestSequence* that are valid for the rubber mount. These components represent all instantiations of ASAM ODS base type *AoTestSequencePart*.

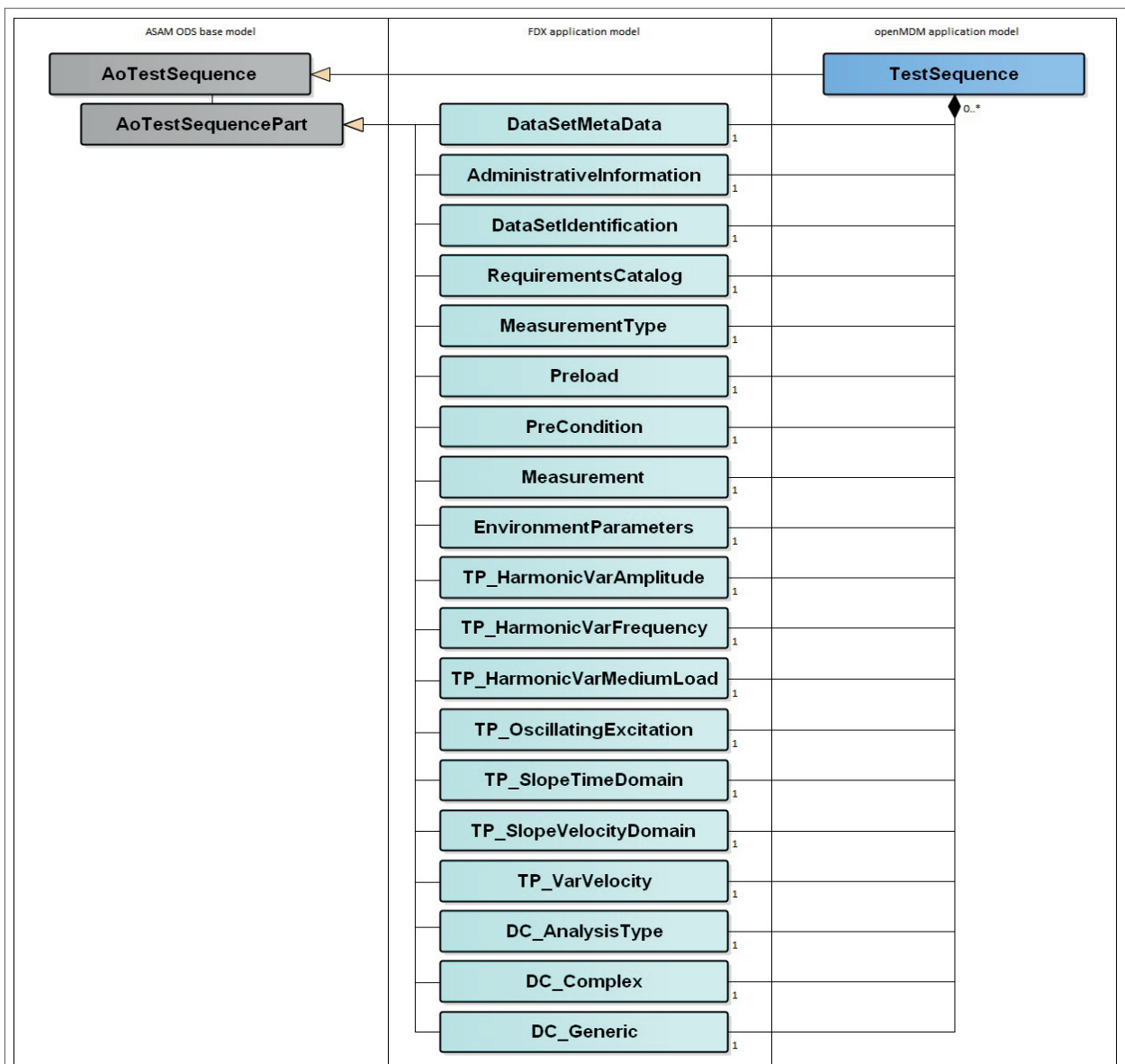


Figure 8-6: Valid components of application element *TestSequence* for rubber mount (example)

Through its components (shown in light blue), class *TestSequence* contains information about the main categories *Dataset meta data*, *Test equipment parameters* and *Derived parameters*. Similar to **Figure 8-4**, the class diagram shown in **Figure 8-7** shows the main categories and subcategories of the prostep ivip / VDA FDX data model (shown in beige) in comparison to the components of application element *TestSequence* in the open MDM application model (shown in light blue). The diagram shows again that there is no hierarchical structure with main categories and subcategories in the openMDM model.

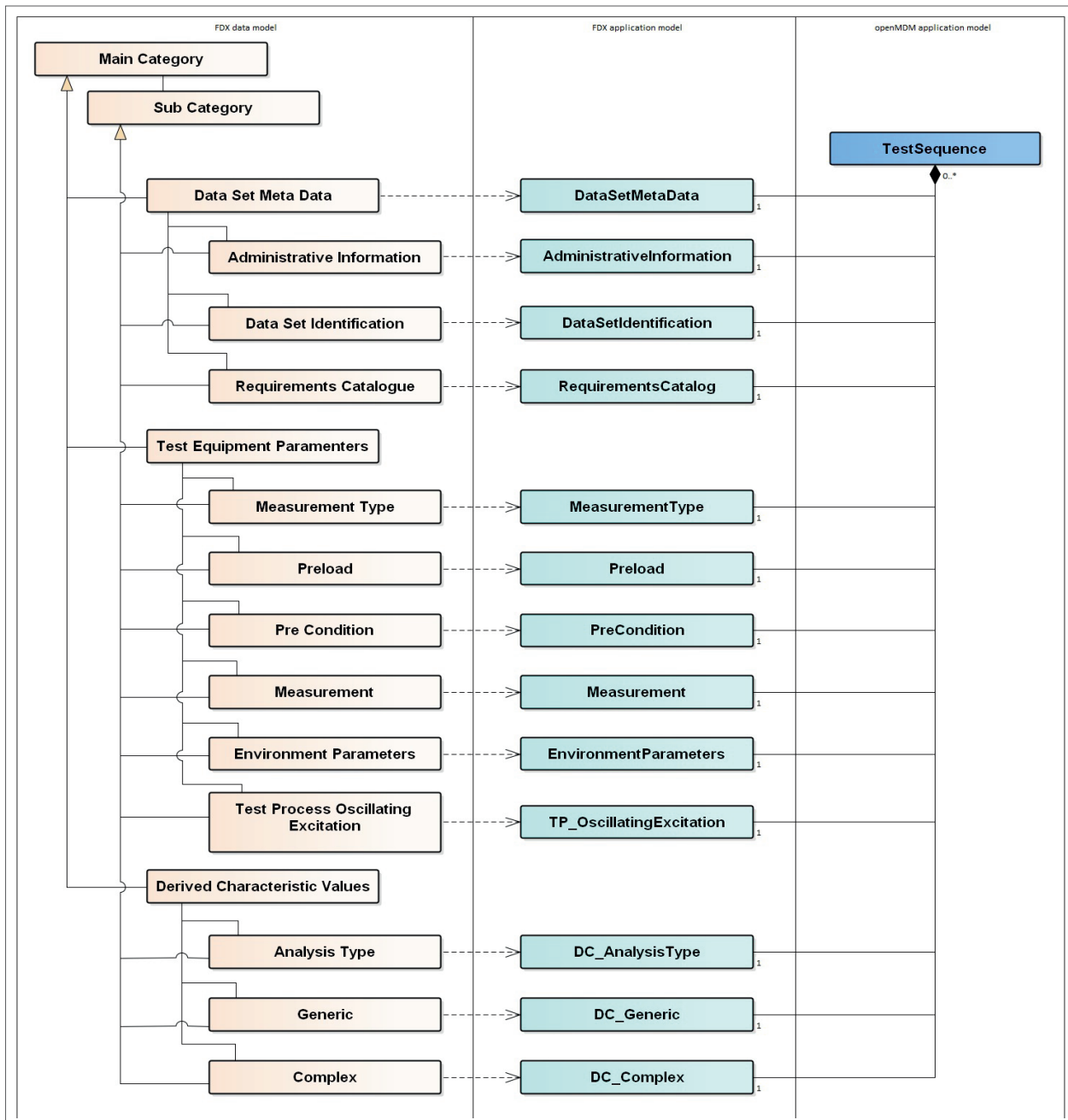


Figure 8-7: Relationships between main categories and subcategories of prostep ivip / VDA FDX data model and components of application element *TestSequence* in openMDM application model - example of rubber mount

The attributes of the main categories and subcategories are also found in the application elements of the openMDM application model. In the class diagram shown in **Figure 8-8**, subcategory *MeasurementType* corresponding to openMDM application element *MeasurementType*, these are for example attributes *DirectionOfMotion*, *MeasurementDynamics*, *Preconditioning*, *Preload*, *SamplingRate*, *SpatialDirection*, *TimeChannelType* and *TimeIncrement*.

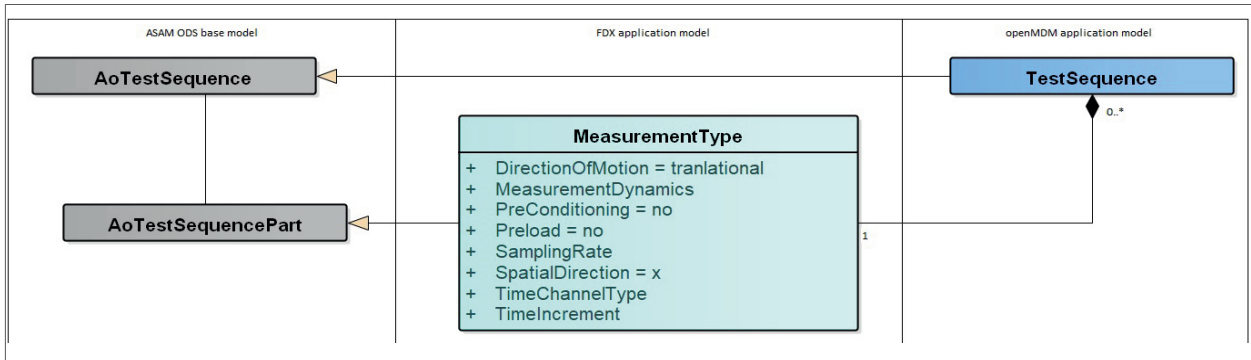


Figure 8-8: Attributes of application element *MeasurementType* and possible values

Note: The note regarding UML default values to **Figure 8-5** also applies to **Figure 8-8**.

Components of application element *TestEquipment*

The class diagram in **Figure 8-9** shows all components of application element *TestEquipment* that are valid for the rubber mount. These components represent all instantiations of ASAM ODS base type *AoTestEquipmentPart*.

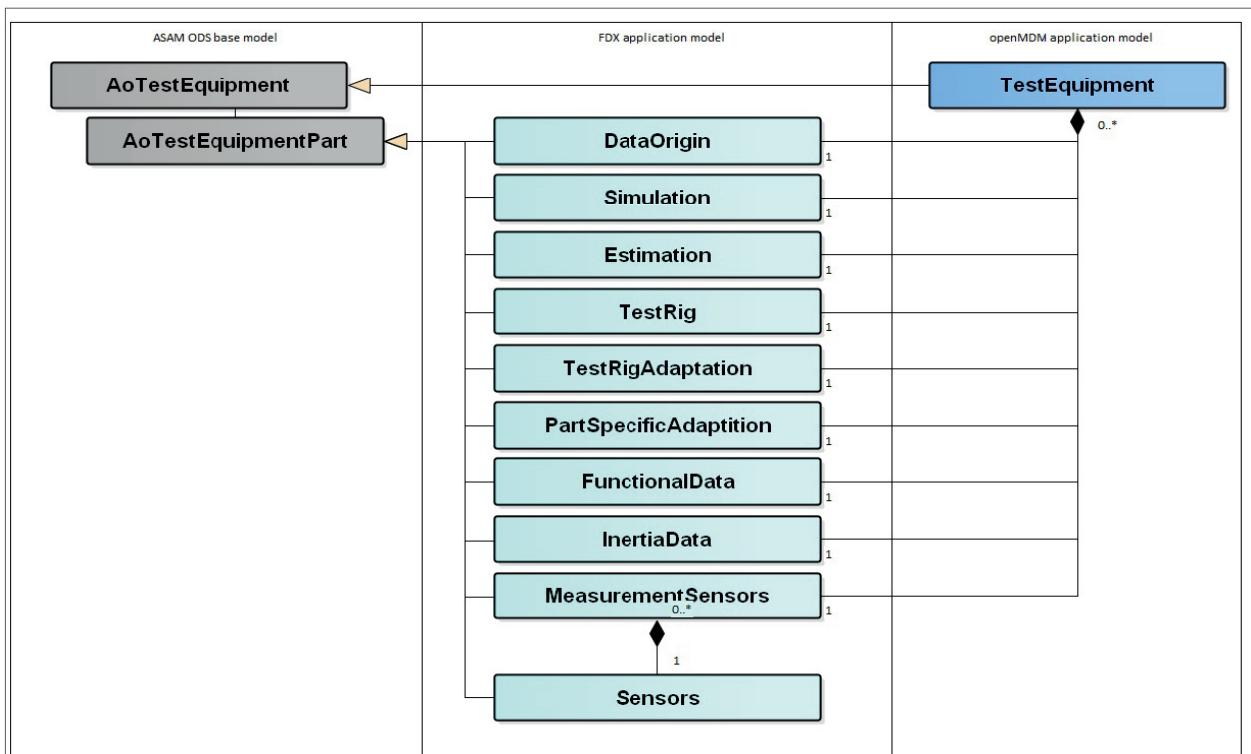


Figure 8-9: Valid components of application element *TestEquipment* for rubber mount (example)

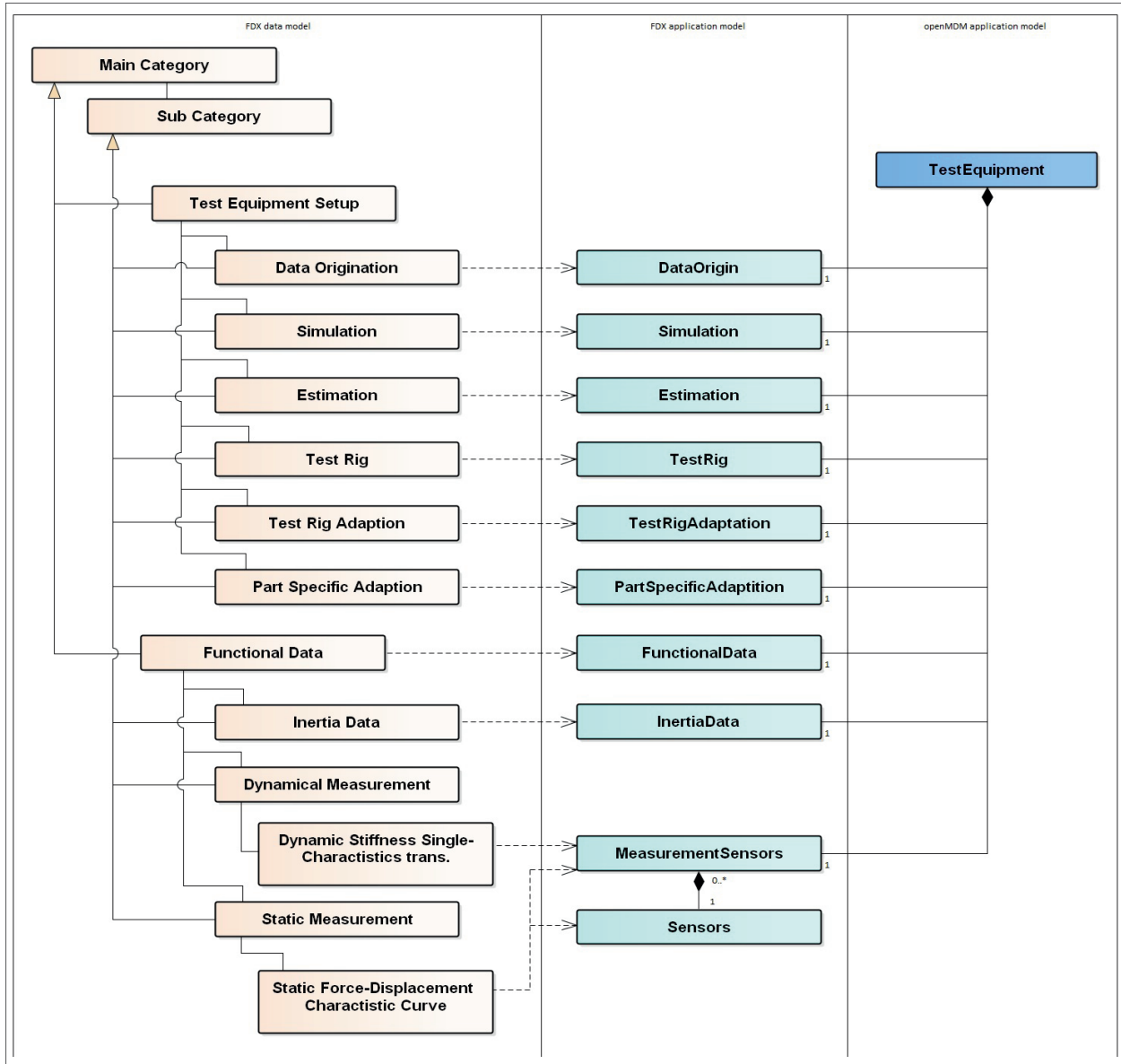


Figure 8-10: Relationships between main categories and subcategories of prostep ivip / VDA FDX data model and components of application element TestEquipment in openMDM application model - example of rubber mount

The attributes of the main categories and subcategories are also found in the application elements of the openMDM application model. In the class diagram shown in **Figure 8-11**, subcategory *DataOrigin* corresponding to application element *DataOrigin*, this is for instance attribute *DataGeneration*.

For subcategories *Static measurement* and *Dynamic measurement* in main category *Functional data*, a separate application element is set up for each attribute in the prostep ivip / VDA FDX application model. In the class diagram shown in **Figure 8-11**, subcategory *static measurement*, the *measurement curve static force-displacement characteristic curve* is for example represented in application element *MeasurementSensors*, and the measurements are represented in application element *Sensors*.

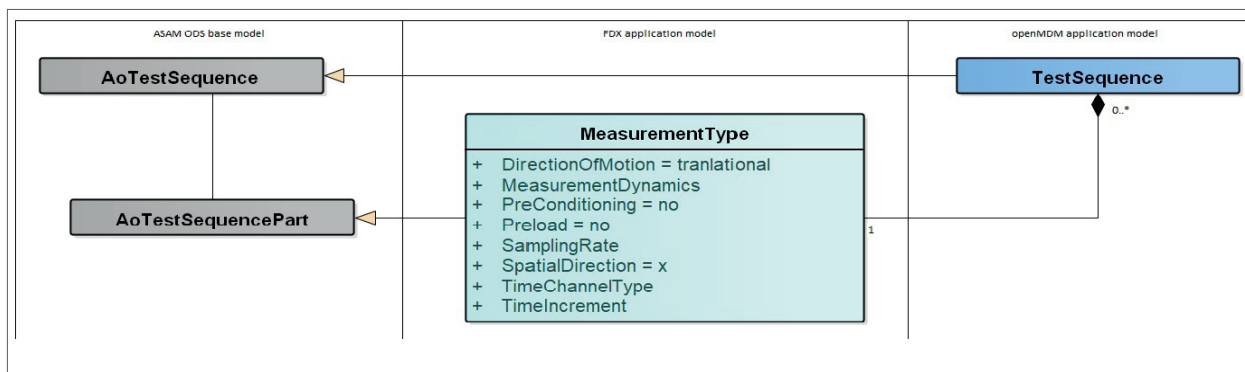


Figure 8-11: Attributes of application elements DataOrigin, MeasurementSensors and Sensors and possible values

Note: The note regarding UML default values to **Figure 8-5** also applies to **Figure 8-11**.

Note: In **Figure 8-11**, the measurement curve is instantiated as attribute "CurveForceDisplacementStatic" of application element *MeasurementSensors*. Following a similar logic, the measurements are instantiated as attributes "=> HysteresisLoopNumber", "=> t", "=> F", "=> u" and "=> v" of application element *Sensors*. While this representation does not strictly conform to the UML standard, it has been chosen for reasons of transparency and with reference to **Figure 8-8** and **Figure 8-5**. For more detailed information regarding the instantiation of application elements, see *Appendix F: Model-driven aspects of openMDM application model*, subchapters **Table 8-2**, **Figure 8-13** and **Figure 8-14**.

8.6 Appendix F: Model-driven aspects of openMDM application model


This appendix is an addendum to chapter "Relationship between ASAM ODS base model, openMDM application model and prostep ivip / VDA FDX application model - example of rubber mount" of prostep ivip / VDA Recommendation PSI 20.

Model-driven aspects of openMDM application model

The features at model level of the openMDM application model are primarily relevant for the application administrator and to a lesser extent for the data requester, but not for the data supplier, as he works only with a specific openMDM application model.

The application elements described above can be classified into two groups.

The first group contains all application elements that are shared across all specific openMDM application models, and typically includes classes *UnitUnderTest*, *TestSequence*, *TestEquipment*, *MeaResult*, *MeaQuantity*, *SubMatrix* and *LocalColumn*. These application elements are visualized in lanes labelled "openMDM application model" (and shown in light blue or purple).

The second group contains application elements for a specific openMDM application model. These classes are components of classes *UnitUnderTest*, *TestSequence* and *TestEquipment*. For the rubber mount, class *MeasurementOrder* is a component of *UnitUnderTest*, class *MeasurementType* is a component of *TestSequence* and class *DataOrigin* is a component of *TestEquipment*. These application elements are visualized in lanes labelled "openMDM application model" (and shown in light blue .

Application elements of the second group might appear in different specific openMDM application models, as is the case with the above classes *MeasurementOrder*, *MeasurementType* and *DataOrigin* that originate from the prostep ivip / VDA FDX application models for shock absorbers, support bearings, overload springs, rubber mounts, suspension springs, decoupling elements and stabilizers. This is not unusual, considering that these and other classes describe general information that applies to all models.



The second group however also contains application elements that only appear in specific openMDM application models, such as for instance class *PartSpecificAdaption* as a component of *TestEquipment* that is found in the two openMDM application models for shock absorbers and overload springs, or class *TP_VarVelocity* as a component of *TestSequence*, which is currently only included the openMDM application model for shock absorbers.

For the second group of application elements, it is quite common for certain attributes of application elements to appear only in specific openMDM application models. Value => *CurrentFeed* of *Test program oscillating excitation* is for example only found in the prostep ivip / VDA FDX application model for shock absorbers.

The two facts that certain application elements only appear in certain specific openMDM application models and that these application elements might only have specific attributes can best be summarized with the term "component applicability".

"Component applicability" must not be confused with the control mechanisms for the visibility of components and attributes described in chapter 4.4.1 "Extensions of openMDM application model". The visibility control mechanisms allow for the definition of conditions that determine the display of templates.

The chapters below describe the main model-driven aspects of the openMDM application model that are relevant for the design of a specific openMDM application model. These aspects are significant for the definition of the test steps that need to be performed in a single test sequence, and the selection of the application elements of the second group and their attributes for the capture of data.

The class diagram in **Figure 8-12** illustrates the relationships between the model-driven aspects of the openMDM application model. The application elements and their attributes that are valid in a specific openMDM application model, for instance for the rubber mount, are defined in templates and catalogues. The classes introduced here represent these templates and catalogues and are shown in yellow  or green  in the respective class diagrams.

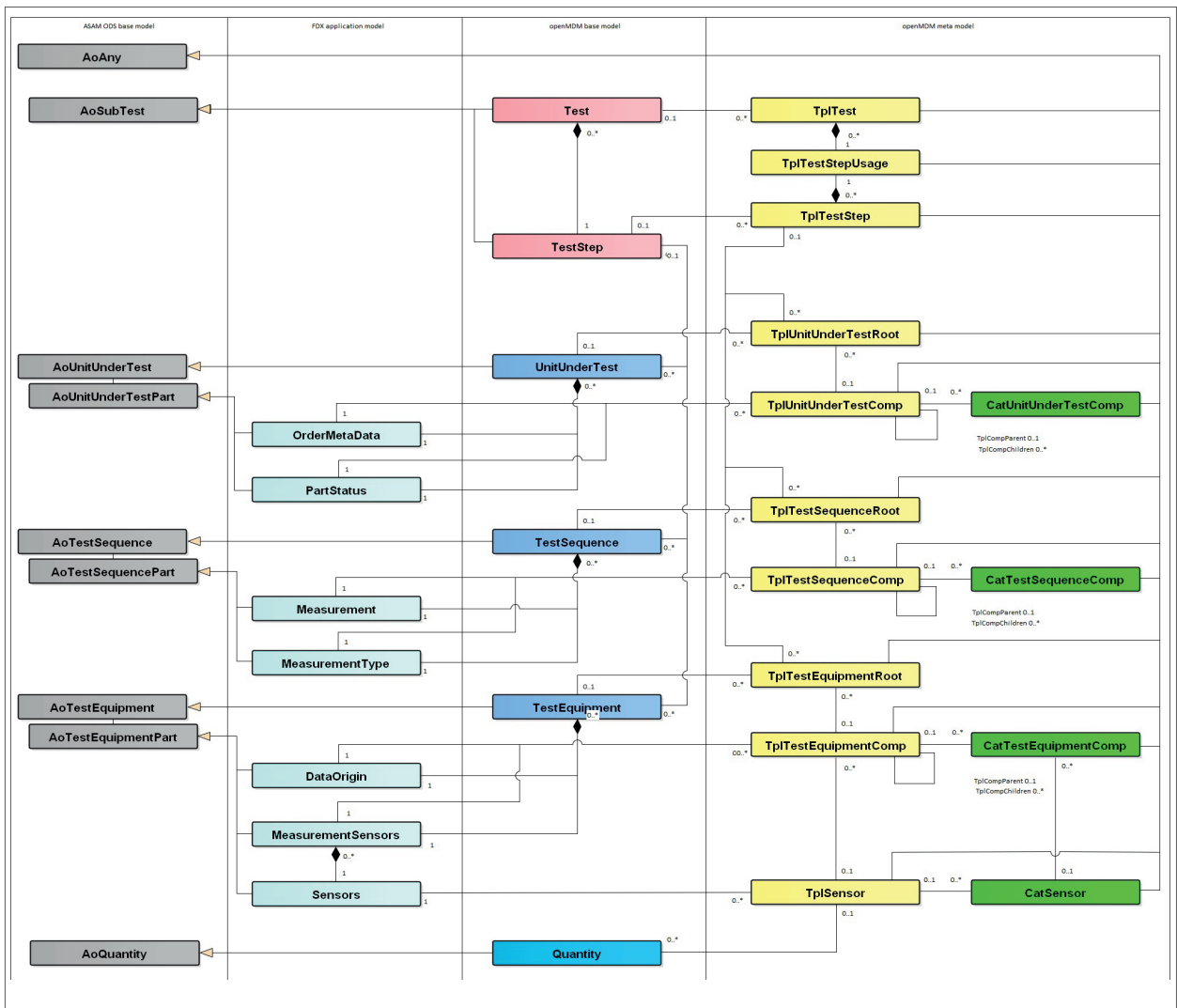


Figure 8-12: Model-driven aspects of openMDM application model and use of templates and catalogues in connection with the ASAM ODS base model and the prostep ivip / VDA FDX application model - example of rubber mount

All classes introduced in this chapter for templates and catalogues are of ASAM ODS base type *AoAny*.

In order to clearly distinguish between application elements that are exclusively designed to represent the model-driven aspects of the openMDM application model and the application elements of the first group, which are shared between all specific openMDM application models, the first group is shown in the "ASAM ODS base model" lane in **Figure 8-12**, while the second group is depicted in the "openMDM meta model" lane.

Table 8-2 below shows an object diagram for the rubber mount, representing a testing scenario with the templates and catalogues for a measurement order for two dynamic and one quasi-static measurement:

Test Instance	TestStep Instance	<> Instance	Cat->Comp Instance	CatSensor Instance	TpiTest TpiTestStep TpiTestStepUsage	Tpi->Root	Tpi->Comp Level 1	Tpi->Comp Level 2	Tpi->Comp Level 3	TpiSensor	Cat->Comp Class	CatSensor Class
Rubber mount					Rubber mount							
	Dynam. 0,05				Dynam. 1							
<>: UnitUnderTest	Rubber mount 001					Rubber mount						
	PartIdentification PartStatus PartModification OrderMetaData MeasurementOrder Supplier PartRevision PartDescription Mass. Geom.Prop. AddInform.PartSpec.					PartIdentification PartStatus PartTest PartModification OrderMetaData MeasurementOrder Supplier PartRevision PartDescription Mass. Geom.Prop. AddInform.PartSpec.					PartIdentification PartStatus PartTest PartModification OrderMetaData MeasurementOrder Supplier PartRevision PartDescription Mass. Geom.Prop. AddInform.PartSpec.	
<>: TestSequence	Dynam. 0,05					Dynam.Harm.Var.Freq.						
	MeasurementType Measurement TP_Harm.Var.Freq. 0,01 Environm.Param. DataSetMetaData Admin.Inform. DataSetIdentification RequirementsCatalog					MeasurementType Preload PreCondition TP_PC_Harm.Var.Freq. Measurement TP_Harm.Var.Freq. Environm.Param. DataSetMetaData Admin.Inform. DataSetIdentification RequirementsCatalog					MeasurementType Preload TP_Harm.Var.Freq. Measurement TP_Harm.Var.Freq. Environm.Param. DataSetMetaData Admin.Inform. DataSetIdentification RequirementsCatalog	
<>: TestEquipment	Curve.Displ.Freq.Trans. DataOrigin					Curve.Displ.Freq.Trans. DataOrigin						
	TestRig InertiaData FunctionalData Curve.Displ.Freq.Trans.					Simulation Estimation TestRig TestRigAdaptation InertiaData DynamicMeasurement Curve.Displ.Freq.Trans.					DataOrigin Simulation Estimation TestRig TestRigAdaptation InertiaData FunctionalData MeasurementSensors	
	=> Freq => C => LossAngle => um => ua => Fm => Fa => TE										=> Freq => C => LossAngle => um => ua => Fm => Fa => TE	Sensors Sensors Sensors Sensors Sensors Sensors Sensors
	Dynam. 0,1				Dynam. 2							
<>: UnitUnderTest	Rubber mount 001											
<>: TestSequence	Dynam. 0,1					Dynam.Harm.Var.Freq.						
	MeasurementType Measurement TP_Harm.Var.Freq. 0,05 Environm.Param. DataSetMetaData Admin.Inform. DataSetIdentification RequirementsCatalog					MeasurementType Preload PreCondition TP_PreCon_Harm.Var.Freq. Measurement TP_Harm.Var.Freq. Environm.Param. DataSetMetaData Admin.Inform. DataSetIdentification RequirementsCatalog					MeasurementType Preload PreCondition TP_Harm.Var.Freq. Measurement TP_Harm.Var.Freq. Environm.Param. DataSetMetaData Admin.Inform. DataSetIdentification RequirementsCatalog	
<>: TestEquipment	Curve.Displ.Freq.Trans.											
	Quasistat.				Quasistat.							
<>: UnitUnderTest	Rubber mount 001											
<>: TestSequence	Quasistatisch					OscillatingExcitation						
	MeasurementType Measurement TP_Oscill.Excit. Environm.Param. DataSetMetaData Admin.Inform. DataSetIdentification RequirementsCatalog					MeasurementType Preload PreCondition TP_PreCon_Oscill.Excit. Measurement TP_Oscill.Excit. Environm.Param. DataSetMetaData Admin.Inform. DataSetIdentification RequirementsCatalog					MeasurementType Preload PreCondition TP_Oscill.Excit. Measurement TP_Oscill.Excit. Environm.Param. DataSetMetaData Admin.Inform. DataSetIdentification RequirementsCatalog	
<>: TestEquipment	ForceDisplacem. DataOrigin					ForceDisplacem. DataOrigin						
	TestRig TestRigAdaptation InertiaData FunctionalData ForceDisplacem.					Simulation Estimation TestRig TestRigAdaptation InertiaData StaticMeasurement ForceDisplacem.					DataOrigin Simulation Estimation TestRig TestRigAdaptation InertiaData FunctionalData MeasurementSensors	
	=> t => v => u => F => Hy.Loo.Num.										=> t => v => u => F => Hy.Loo.Num.	Sensors Sensors Sensors Sensors Sensors

Table 8-2: Testing scenario with templates and catalogues for a measurement order - example of rubber mount

For each test step, the table is divided into three horizontal sections, containing the components of application elements *UnitUnderTest*, *TestSequence* and *TestEquipment*. Placeholder "<>" in the column headers stands for the associated classes of the application elements. Example: for placeholder "UnitUnderTest", the respective columns contain application elements *UnitUnderTest*, *TplUnitUnderTestRoot*, *TplUnitUnderTestComp* and *CatUnitUnderTestComp*. The table rows contain the names of the instances of the respective application elements. These names appear in the text below with single quotation marks (').

Selection of test steps for a test by means of application elements *TplTest*, *TplTestUsage*, *TplTestStep*

Classes *TplTest*, *TplTestUsage*, and *TplTestStep* represent templates for the planned test. Classes *Test* and *TestStep* described in the previous chapter represent the tests resulting from these templates.

In our example, this is a measurement of the rubber mount in three test steps. The first two test steps concern dynamic measurements with different amplitudes, while the third test step is a quasi-static measurement.

The multiplicity of classes *TplTest*, *TplTestUsage* and *TplTestStep* is an attributed N:M relationship. Class *TplTest* represents the template for the entire test (all 3 steps). Class *TplTestStepUsage* represents the selection of a template for one test step. Class *TplTestStep* represents the template of one test steps.

Application element *TplTest* contains 0..* application elements *TplTestUsage*. Class *TplTestUsage* represents the selection of a template for one test step. Application element *TplTestUsage* thereby refers to a specific application element *TplTestStep*.

By creating one instance of *TplTest*, the chain of associations between the classes *TplTest*, *TplTestStepUsage* and *TplTestStep* make sure that the correct number of instances of *TplTestStep* are found. For the instance of *TplTest*, an instance of class *Test* is generated, while for each instance of *TplTestStep* determined in this manner, an instance of class *TestStep* is generated and assigned to the instance of *TplTest*.

In the scenario shown in **Table 8-2**, we need an application element *TplTest* (named 'RubberMount'). As there are two dynamic measurements and one quasi-static measurement, three application elements *TplTestUsage* are required, each selecting one application element *TplTestStep* (named 'Dynam. 1', 'Dynam. 2' and 'Quasistat.').

Selection of application elements for a test step by means of application elements *TplUnitUnderTestRoot*, *TplTestSequenceRoot*, *TplTestEquipmentRoot*

These templates can be assigned according to their hierarchy to main categories *Measurement order*, *Order meta data*, *Component-specific additional information*, *Unit under test*, *Dataset meta data*, *Test equipment parameters*, *Derived parameters*, *Test equipment setup* and *Functional data*, as well as to their subcategories.

Class *TplUnitUnderTestRoot* represents the template for the components of class *UnitUnderTest* to be captured during the test step. Class *TplTestSequenceRoot* represents the template for the components of class *TestSequence* to be captured during the test step. Class *TplTestEquipmentRoot* represents the template for the components of class *TestEquipment* to be captured during the test step.

For this purpose, one instance of *TplTestStep* refers to 0..1 instances of class *TplUnitUnderTestRoot*, *TplTestSequenceRoot* and *TplTestEquipmentRoot* respectively.

One instance of class *TplUnitUnderTestRoot* refers to 0..* instances of class *TplUnitUnderTestComp*. Class *TplUnitUnderTestComp* represents the selection of the class of a component of class *UnitUnderTest*. For this purpose, one instance of class *TplUnitUnderTestComp* refers to one instance of class *CatUnitUnderTestComp*. Class *CatUnitUnderTestComp* represents the catalogue of the classes of application elements that are available as components in class *UnitUnderTest*.

One instance of class *TplUnitUnderTestComp* might also refer to 0..* other instances of the same class *TplUnitUnderTestComp*. In this manner, a tree of instances of class *TplUnitUnderTestComp* can be generated. This means that templates *TplUnitUnderTestComp* can be arranged so as to reflect the hierarchical structure of main categories *Measurement order*, *Order meta data*, *Component-specific additional information* and *Unit under test* and their respective subcategories.

In the scenario in Table 8-2, application element *TplTestStep* 'Dynam. 1' refers to application element *TplUnitUnderTestRoot* 'Rubber mount', forming a tree of application elements *TplUnitUnderTestComp*. Application element *TplUnitUnderTestRoot* 'Dynam. 1' refers to three application *TplUnitUnderTestComp*, namely 'PartIdentification', 'OrderMetaData' and 'AddInformationPartSpecific'. Application element *TplUnitUnderTestComp* 'PartIdentification' refers to three application elements *TplUnitUnderTestComp*, namely 'PartStatus', 'PreTest' and 'PartModification'. Application element *TplUnitUnderTestComp* 'OrderMetaData' refers to five application elements *TplUnitUnderTestComp*, namely 'MeasurementOrder', 'Supplier', 'PartRevision', 'PartDescription' and 'MassGeometricalProperties'. Application element *TplUnitUnderTestComp* 'AddInformationPartSpecific' refers to one application element *AddInformationPartSpecific* 'AddInformationPartSpecific'.

Note: In scenario in **Table 8-2**, the templates reflect the data constellation of a specific openMDM application model for the rubber mount that was available at the time of drafting of this document. While the tree of application elements *TplUnitUnderTestComp* includes all main categories *Measurement order*, *Order meta data*, *Component-specific additional information* and *Unit under test* and the associated subcategories, the structure of the tree differs from that of the main categories and subcategories in the prostep ivip / VDA FDX data model (see also **Figure 8-4**). Example: application element *TplUnitUnderTestComp* 'MeasurementOrder' is positioned under application element *TplUnitUnderTestComp* 'OrderMetaData' while there is a separate main category *Measurement order* in the prostep ivip / VDA FDX data model.

For the instance of class *CatUnitUnderTestComp* determined in this manner, an application element of class *UnitUnderTest* is generated. For each instance of class *CatUnitUnderTestComp* determined in this manner, an application element of the respective class is generated and assigned as a component of this instance to *UnitUnderTest*.

In the example of the rubber mount, to obtain an application element *MeasurementOrder* as a component of *UnitUnderTest* would require that, based on application element *TplUnitUnderTestRoot*, the tree of application elements *TplUnitUnderTestComp* contains one application element referring to a specific application element *CatUnitUnderTestComp*. This application element *CatUnitUnderTestComp* would correspond to class *MeasurementOrder* as a component of *UnitUnderTest*.

This approach applies accordingly to classes *TplTestSequenceRoot*, *TplTestSequenceComp* and *CatTestSequenceComp* for the components of class *TestSequence* to be included in the test step, and for classes *TplTestEquipmentRoot*, *TplTestEquipmentComp* and *CatTestEquipmentComp* for components of class *TestEquipment* to be included in the test step.

For the instance of class *CatTestSequenceComp* determined in this manner, an application element of class *TestSequence* is generated, and for the instance of class *CatTestEquipmentComp* an application element of class *TestEquipment* is generated.

For the rubber mount in our example, an application element *TplTestSequenceComp* would point to an application element *Measurement*, and an application element *TplTestEquipmentComp* would point to an application element *DataOrigin*.

In the same way, templates *TplTestSequenceComp* can be arranged to organize the main categories *Dataset meta data*, *Test equipment parameters* and *Derived parameters* as well as their subcategories in a hierarchical structure. This also applies to templates *TplTestEquipmentComp* with main categories *Test equipment setup* and *Functional data* and their respective subcategories.

For the two dynamic measurements of the scenario shown in **Table 8-2**, application element *TplTestStep* 'Dynam. 1' refers to application element *TplTestSequenceRoot* 'HarmonicVarFrequency'. For the quasi-static measurement, application element *TplTestStep* 'Quasistat.' refers to application element *TplTestSequenceRoot* 'OscillatingExcitation'. These application elements *TplTestSequenceRoot* each create a tree of application elements *TplTestSequenceComp*.

Note: Again, the tree of application elements *TplTestSequenceComp* includes all main categories *Dataset meta data*, *Test equipment parameters* and *derived parameters* as well as their subcategories. The tree structure differs however from the structure of the main categories and subcategories of the prostep ivip / VDA FDX data model (see also **Figure 8-4**). Example: application element *TplTestEquipmentComp* 'TP_HarmonicVarFrequency' is positioned under application element *TplUnitUnderTestComp* 'Measurement', while there is a separate main category *Test program oscillating excitation* in the prostep ivip / VDA FDX data model.

For the two dynamic measurements of the scenario shown in **Table 8-2**, application element *TplTestStep* 'Dynam. 1' refers to application element *TplTestEquipmentRoot* 'CurveDisplacementFrequencyTranslational'. For the quasi-static measurement, application element *TplTestStep* 'Quasistat.' refers to application element *TplTestSequenceRoot* 'ForceDisplacement'. These application elements *TplTestEquipmentRoot* each create a tree of application elements *TplTestEquipmentComp*.

Note: While the tree of application elements *TplTestEquipmentComp* includes all main categories *Test equipment setup* and *Functional data* and the associated subcategories, the structure of the tree differs from that of the main categories and subcategories in the prostep ivip / VDA FDX data model (see also **Figure 8-10**). Example: application element *TplTestEquipmentComp* 'DataOrigin' is positioned right under application element *TplTestEquipmentRoot* 'CurveDisplacementFrequencyTranslational', while there is a subcategory *Data origin* in main category *Test equipment setup* in the prostep ivip / VDA FDX data model.

Application elements that are not required, such as component *PreTest* of application element *UnitUnderTest*, components *Preload* and *PreCondition* of application element *TestSequence*, and components *Simulation* and *Estimation* of application elements *TestEquipment* will be represented in the scenario in **Table 8-2** by blank fields.

If application element *TplTestEquipmentComp* generates the template of an application element in subcategories *Static measurement* and *Dynamic measurement* from main categories *Functional data*, the measurements of the curve must be described by additional application elements. In our example of a rubber mount, the application element *MeasurementSensors* determined by *CatTestEquipmentComp* belongs to one of the subcategories *Static measurement* or *Dynamic measurement*.

To determine the measurements of the measurement curve, one instance of class *TplTestEquipmentComp* refers to 0..* instances of class *TplSensor*. Class *TplSensor* represents the selection of the class of the application element to be describe a measurement. For this purpose, one instance of class *TplSensor* refers to one instance of class *CatSensor*. Class *CatSensor* represents the catalogue of the classes of application elements that are used to describe measurements.

The application elements derived from *CatSensor* are components of the application element derived from application element *CatTestEquipmentComp*. Therefore, the application elements derived from *TplSensor* are the subcomponents of application element *TestEquipment*.

For each instance of class *TplSensor* determined in this manner, an instance of class according to *CatSensor* is generated and assigned the component to the instance corresponding to *CatTestEquipmentComp*. In the example of the rubber mount, the number of application elements *TplSensor* corresponds to the number of application elements *Sensors* appearing as components of application element *MeasurementSensors*.

Classes *CatUnitUnderTestAttr*, *CatTestSequenceAttr* and *CatTestEquipmentAttr* as well as *CatSensorAttr* (**Figure 8-1**) represent catalogues of all attributes available in the classes of the components of classes *UnitUnderTest*, *TestSequence* and *TestEquipment* as well as in the subcomponents of class *TestEquipment*.

Classes *TplUnitUnderTestAttr*, *TplTestSequenceAttr* and *TplTestEquipmentAttr* as well as *TplSensorAttr* (**Figure 8-1**) represent the templates with which the components of classes *UnitUnderTest*, *TestSequence* and *TestEquipment* as well as the attributes to be included in the subcomponents of class *TestEquipment*.

For this selection, application element *TplUnitUnderTestComp* contains 0..* application elements *TplUnitUnderTestAttr*; application element *TplTestEquipmentComp* contains 0..* application elements *TplTestEquipmentAttr* and application element *TplTestEquipmentComp* contains 0..* application elements *TplTestEquipmentAttr*.

To determine an attribute, application element *TplUnitUnderTestAttr* refer to 0..1 specific application element *CatTestEquipmentComp*; application element *TplUnitTestEquipmentAttr* refers to 0..1 specific application element *CatTestEquipmentAttr*, and application element *TplTestSequenceAttr* refers to 0..1 specific application element *CatTestSequenceAttr*. Application element *TplSensorAttr* refers to 0..1 specific application element *CatSensorAttr*.

With classes *ValueList* and *ValueListValue* of ASAM ODS base type *AoParameterSet* and *AoParameter* (**Figure 8-1**), it is possible to limit the value range of an attribute to a specific value list types and values. Class *ValueList* thereby represents a value list and class *ValueListValue* represents the type of the value list. For this purpose, application element *ValueList* contains 0..* application elements *ValueListValue*.

To assign the value list, application elements *CatUnitUnderTestAttr*, *CatTestSequenceAttr*, *CatTestEquipmentAttr* and *CatSensorAttr* refer to 0..1 specific application element *ValueList*.

Representation of prostep ivip / VDA FDX data model by prostep ivip / VDA FDX application model

The object diagram in **Figure 8-13** compares the prostep ivip / VDA FDX data model with the prostep ivip / VDA FDX application model, based on the scenario of a typical data request and delivery (see **Figure 3-10** and **Figure 3-11**):

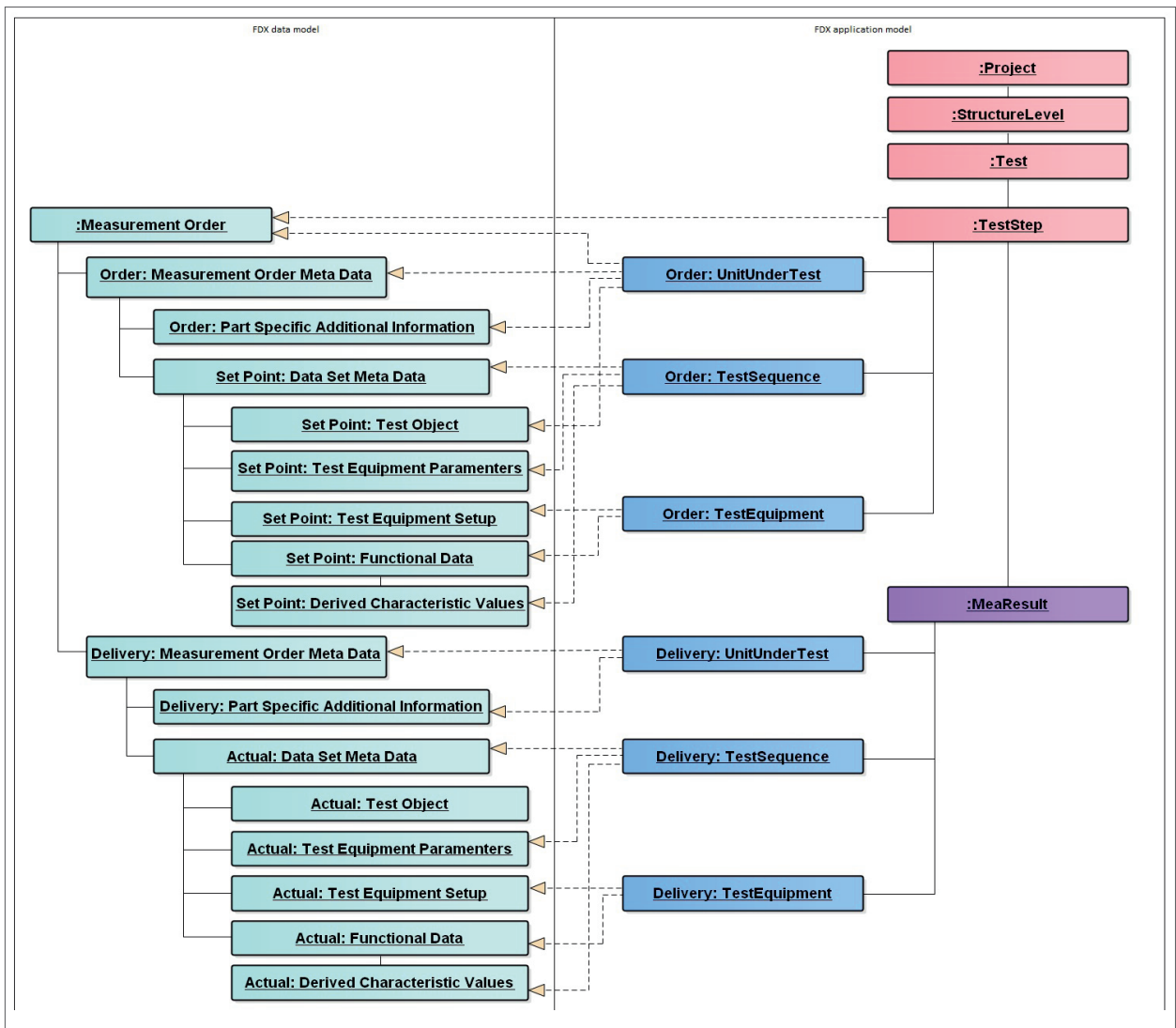


Figure 8-13: Scenario for representation of prostep ivip / VDA FDX data model by prostep ivip / VDA FDX application model

The prefix assigns the object to the data order ('Order') or the data delivery ('Delivery') data. The prefix assigns on object to setpoint and actual functional data.

The postfix indicates the class of the application element.

In the above scenario, there is a *measurement order* implemented by means of the application elements *TestStep* and *UnitUnderTest* labelled with 'Order'. The data of the order is distributed across the application elements *UnitUnderTest*, *TestSequence* and *TestEquipment* labelled with 'Order', and their components (not shown in diagram).

For this *measurement order*, there is a data delivery, implemented with application element *MeaResult*. The data of the delivery is distributed across the application elements *UnitUnderTest*, *TestSequence* and *TestEquipment* labelled with 'Delivery', and their components.

Scenario of a specific openMDM application model - example of rubber mount

The object diagram in *Figure 8-14* shows a typical scenario of a data order and a data delivery for a rubber mount:

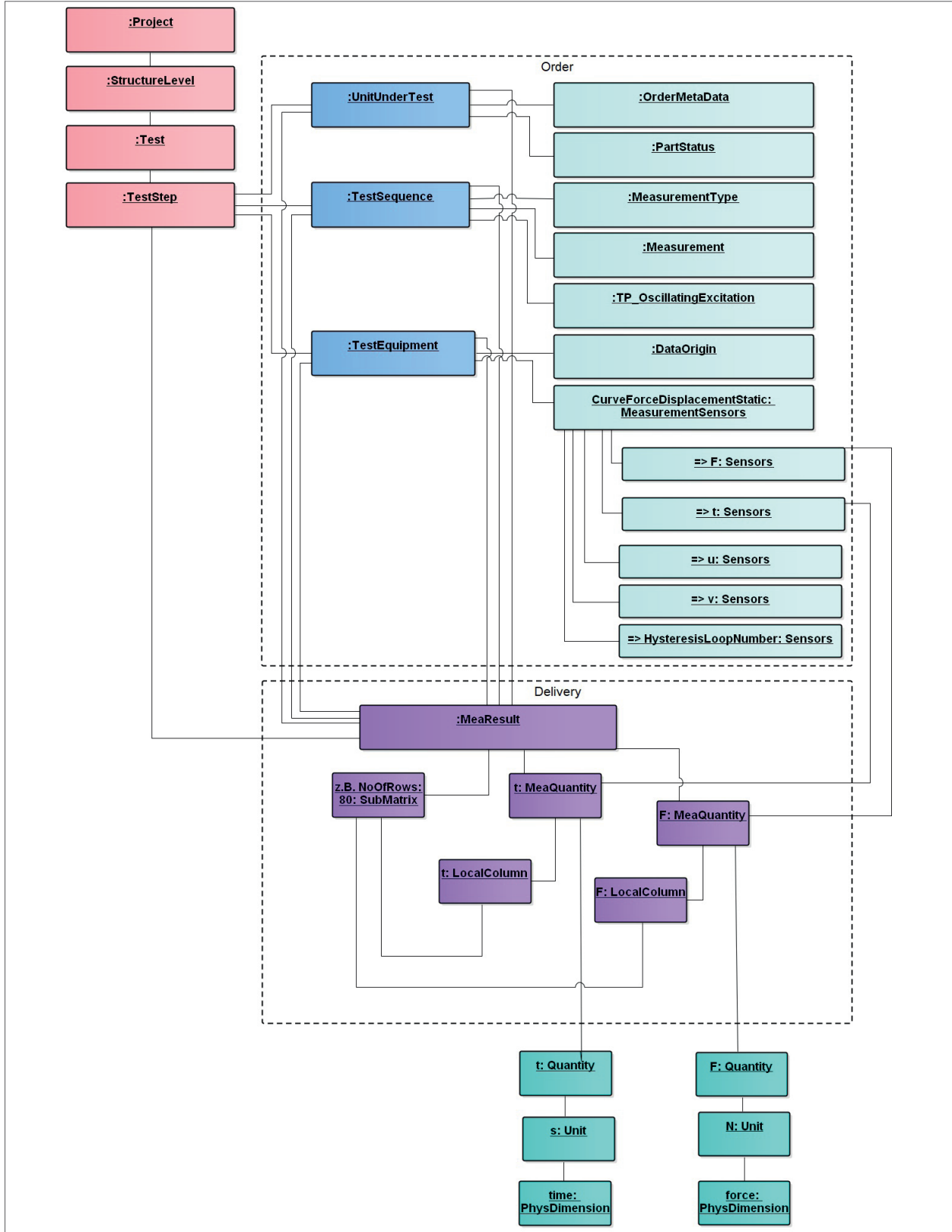


Figure 8-14: Scenario of a specific openMDM application model - example of rubber mount

The name of an object consists of a prefix and a postfix, separated by a colon (:).

The prefix identifies the purpose of the application element, if this information is not already clearly given by the class of the application element.

The postfix indicates the class of the application element.

The application element *TP_OscillatingExcitation* represents subcategory *Test program oscillating excitation*.

The application element *MeasurementSensors* with prefix 'CurveForceDisplacementStatic' represents *measuring program static force-displacement curve*. The application elements *Sensors* with prefixes '=> t', '=> F', '=> u', '=> v' and '> HysteresisLoopNumber' represent the individual measurements.

The result of the measurement is summarized in an application element *MeaResult*. Each measurement is captured through an application element *MeaQuantity*, *SubMatrix* and *LocalColumn*.

As the ordered measurement was performed by the data supplier according to the measurement order, so that the main categories *Order meta data*, *Component-specific additional information*, *Dataset meta data*, *Unit under test*, *Test equipment setup*, *Test equipment parameters*, *Functional data* and *Derived parameters* of the data delivery correspond exactly to those in the data request, application element *MeaResult* refers to application elements *UnitUnderTest*, *TestSequence* and *TestEquipment* in the order.

8.7 Appendix G: Representation of model-driven aspects of openMDM4 application model in ATFX

This appendix is an addendum to chapter "Relationship between ASAM ODS base model, openMDM application model and prostep ivip / VDA FDX application model - example of rubber mount" of prostep ivip / VDA Recommendation PSI 20.

Representation of model-driven aspects of openMDM4 application model in ATFX

The mixed class/object diagram in Figure 8-15 shows how the model-driven aspects of the openMDM4 application model are embedded in the ASAM ODS base model:

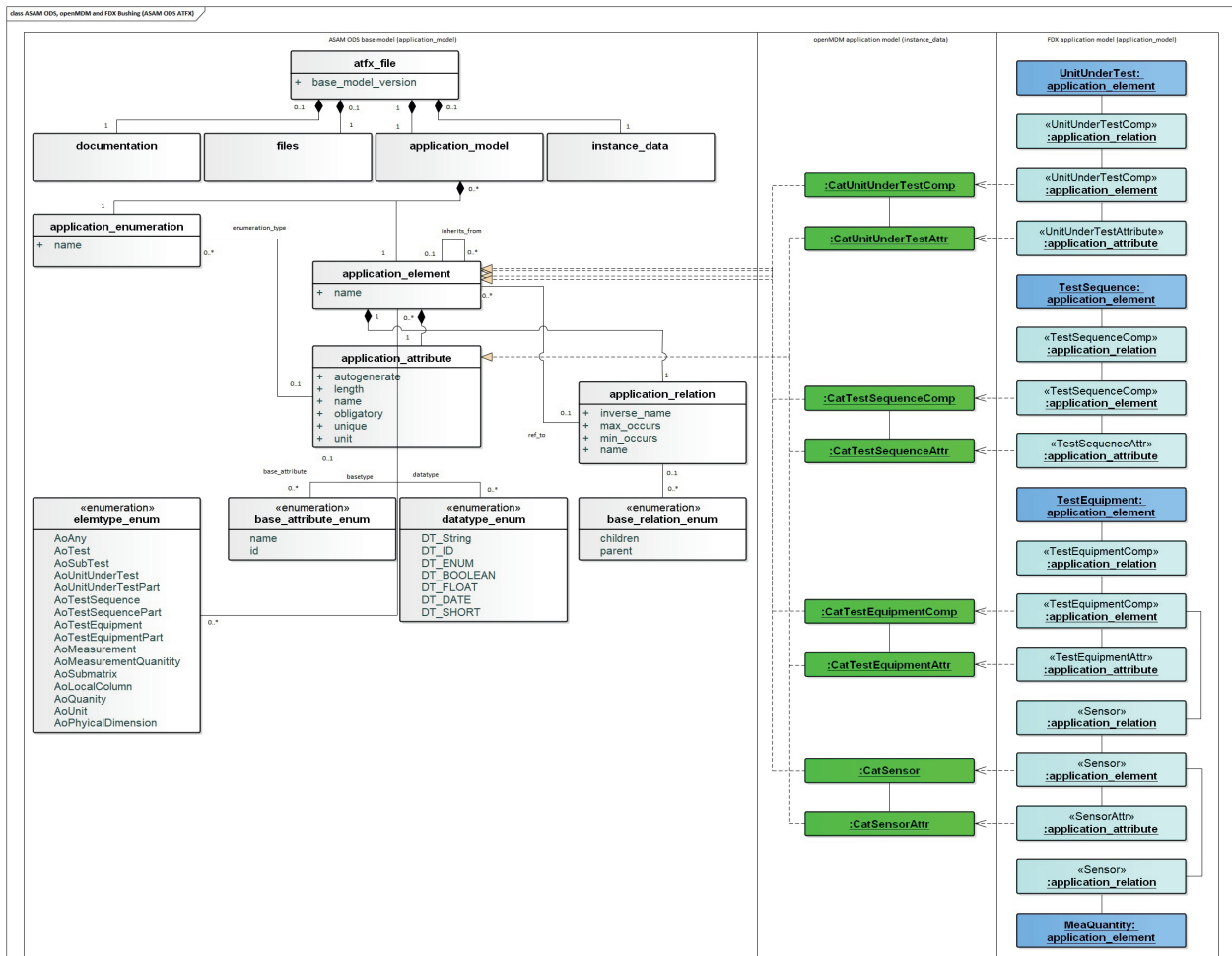


Figure 8-15: Model-driven aspects of openMDM application model

Section `application_model`, class `application_enumeration` and in particular class `application_element` describe all valid application elements for section `instance_data`.

Class `application_model` thereby represents the class of the application element, as it defines its attributes through class `application_attribute` and its inclusion in an association with another application element through class

application_relation. Association *application_relation.ref_to* is the application element at the other end of the association. A single-direction association is indicated by an instance of *application_relation*; a bi-directional association between two application elements is indicated by two instances of *application_relation*.

Classes *elemtype_enum*, *base_attribute_enum*, *datatype_enum* and *base_relation_enum* describe properties of the application elements, i.e. their ASAM ODS base type, the ASAM ODS base attribute through which the attribute of an application element is instantiated, the data type of the attribute of an application model and the basic relationship that implements the relationship of an application element.

As described above, classes *CatUnitUnderTestComp*, *CatTestSequenceComp*, *CatTestEquipmentComp* and *CatSensor* represent the catalogues of the classes of application elements that are available as components of classes *UnitUnderTest*, *TestSequence* and *TestEquipment*, and as subcomponents of class *TestEquipment*.

As described above classes *CatUnitUnderTestAttr*, *CatTestSequenceAttr*, *CatTestEquipmentAttr* and *CatSensorAttr* represent the catalogues of the attributes of these components.

Classes *CatUnitUnderTestComp*, *CatTestSequenceComp*, *CatTestEquipmentComp* and *CatSensor* as well as *CatUnitUnderTestAttr*, *CatTestSequenceAttr*, *CatTestEquipmentAttr* and *CatSensorAttr* are themselves modelled as *application_element* in section *application_model*.

The instances of the classes *CatUnitUnderTestComp*, *CatTestSequenceComp*, *CatTestEquipmentComp* and *CatSensor*, as well as *CatUnitUnderTestAttr*, *CatTestSequenceAttr*, *CatTestEquipmentAttr* and *CatSensorAttr* in section *instance_data* are now fully fledged templates in which the respective instances of classes *application_element*, *application_attribute* and *application_relation* are generated in section *application_model*, describing the classes of the components of classes *UnitUnderTest*, *TestSequence* and *TestEquipment* (see lane "openMDM application model (instance_data)" and dashed arrows in **Figure 8-15**).

For each instance of class *CatUnitUnderTestComp*, *CatTestSequenceComp*, *CatTestEquipmentComp* and *CatSensor* in section *application_model*, an instance of class *application_element* is generated (see lane in "prostep ivip / VDA FDX application model (application_model)" and dashed arrows in **Figure 8-15**).

Along the same lines, for each instance of classes *CatUnitUnderTestAttr*, *CatTestSequenceAttr*, *CatTestEquipmentAttr* and *CatSensorAttr* in section *application_model*, a corresponding instance of class *application_attribute* is generated, which is assigned to the respective instance of class *CatUnitUnderTestComp*, *CatTestSequenceComp*, *CatTestEquipmentComp* and *CatSensor* (see lane in "prostep ivip / VDA FDX application model (application_model)" and dotted arrows in **Figure 8-7**).

To indicate that a new application element originating from an instance of *CatUnitUnderTestAttr*, *CatTestSequenceAttr* or *CatTestEquipmentAttr* is a component of class *UnitUnderTest*, *TestSequence* or *TestEquipment*, a corresponding instance of class *application_relation* is generated.

To indicate that a new application element originating from an instance of *CatSensor* is a component of a class originating from *CatTestEquipmentComp*, two instances of class *application_relation* are generated for bidirectional associations.

In addition, two instances of class *application_relation* are generated, representing the bidirectional association between class *MeaQuantity* and the class originating from *CatSensor*.

8.8 Appendix H: XML schema of ASAM ATFX data exchange format

The XML schema of the ASAM ATFX data exchange format has been developed by ASAM ([ASAM ODS - Standards](#)) and is laid down in the 4 XML schema definition named Schema.xsd, ASAM_HDTypes.xsd, HelperSchema.xsd and ODSBaseModelSpecSchema.xsd.

These define in particular the XML schema for the ASAM ATFX data exchange format in general, and the XML schema of the application elements of section application_model. They do however not define the XML schema of the instances of the application elements of section instance_data, as these application elements are only created in the course of the instantiation of the ASAM ODS base model.

In this context, the ASAM ODS Checker has been developed ([Schema Validation of ATFX-Files According to ASAM ODS V5.3.0](#)), which can be used to translate the application_element described in section application_model (see previous chapter) into the respective XML schema definitions (xsd) of the instances of the application elements of section instance_data.

An application_attribute of base_attribute_enum 'id' and of application_attribute.name 'Id' is translated into xsd:element of type xsd:long. An application_relation representing the end of an association is translated into xsd:element of type t_reference. xsd:simpleType t_reference is an xsd:union of xsd:integer and xsd:string. Such an attribute thus compiles all id s of the application elements at the opposite end of the association in a list of integers separated by spaces.

The excerpt from an ATFX exchange file in **Figure 8-16** is from a FDA FDX application model for the rubber mount and shows the selected application_attribute and application_relation of application_element for application element MeasurementOrder in section application_element:

```
<application_model>
```

```

<application_element>
  <name>UnitUnderTest</name>
  <basetype>AoUnitUnderTest</basetype>
  <application_attribute>
    <name>Id</name>
    <base_attribute>id</base_attribute>
    <autogenerate>true</autogenerate>
    <obligatory>true</obligatory>
  </application_attribute>
  <application_attribute>
    <name>Name</name>
    <base_attribute>name</base_attribute>
    <obligatory>true</obligatory>
    <length>100</length>
  </application_attribute>
  <relation_attribute>
    <name>MeasurementOrder</name>
    <ref_to>MeasurementOrder</ref_to>
    <base_relation>children</base_relation>
    <min_occurs>0</min_occurs>
    <max_occurs>Many</max_occurs>
    <inverse_name>UnitUnderTest</inverse_name>
  </relation_attribute>
  <relation_attribute>
    <name>MeaResults</name>
    <ref_to>MeaResult</ref_to>
    <base_relation>measurement</base_relation>
    <min_occurs>0</min_occurs>
    <max_occurs>Many</max_occurs>
    <inverse_name>UnitUnderTest</inverse_name>
  </relation_attribute>
  <relation_attribute>
    <name>TestSteps</name>
    <ref_to>TestStep</ref_to>
    <min_occurs>0</min_occurs>
    <max_occurs>Many</max_occurs>
    <inverse_name>UnitUnderTest</inverse_name>
  </relation_attribute>
  <relation_attribute>
    <name>TplUnitUnderTestRoot</name>
    <ref_to>TplUnitUnderTestRoot</ref_to>
    <min_occurs>0</min_occurs>
    <max_occurs>1</max_occurs>
    <inverse_name>UnitUnderTest</inverse_name>
  </relation_attribute>
</application_element>
</application_model>

```

Figure 8-16: Excerpt from application_element section of an ATFX exchange file; example of prostep ivip / VDA FDX application model for the rubber mount, for application element UnitUnderTest

The excerpt from an ATFX exchange file in Figure 8-17 is from a FDA FDX application model for the rubber mount and shows the selected application_attribute and application_relation of application_element for application element MeasurementOrder in section application_element:

```

<application_model>
  <application_element>
    <name>MeasurementOrder</name>
    <basetype>AoUnitUnderTestPart</basetype>
    <application_attribute>
      <name>Id</name>
      <base_attribute>id</base_attribute>
      <autogenerate>>true</autogenerate>
      <obligatory>>true</obligatory>
    </application_attribute>
    <application_attribute>
      <name>Name</name>
      <base_attribute>name</base_attribute>
      <obligatory>>true</obligatory>
      <length>50</length>
    </application_attribute>
    <relation_attribute>
      <name>UnitUnderTest</name>
      <ref_to>UnitUnderTest</ref_to>
      <base_relation>parent_unit_under_test</base_relation>
      <min_occurs>1</min_occurs>
      <max_occurs>1</max_occurs>
      <inverse_name>MeasurementOrder</inverse_name>
    </relation_attribute>
    <relation_attribute>
      <name>TplUnitUnderTestComp</name>
      <ref_to>TplUnitUnderTestComp</ref_to>
      <min_occurs>1</min_occurs>
      <max_occurs>1</max_occurs>
      <inverse_name>MeasurementOrder</inverse_name>
    </relation_attribute>
  </application_element>
</application_model>

```

Figure 8-17: Excerpt from application_element section of an ATFX exchange file; example of prostep ivip / VDA FDX application model for rubber the mount, for application element MeasurementOrder

The excerpt from the XML schema of an ATFX exchange file in Figure 8 18 shows the schema element of application element UnitUnderTest in section instance_data:

```
<xsd:complexType name="t_UnitUnderTest">
  <xsd:all>
    <xsd:element minOccurs="1" name="Id" type="xsd:long"/>
    <xsd:element minOccurs="1" name="Name" type="xsd:string"/>
    <xsd:element minOccurs="0" name="MimeType" type="xsd:string"/>
    <xsd:element minOccurs="0" name="MeasurementOrder" type="t_reference"/>
    <xsd:element minOccurs="0" name="MeaResults" type="t_reference"/>
    <xsd:element minOccurs="0" name="TestSteps" type="t_reference"/>
    <xsd:element minOccurs="0" name="TplUnitUnderTestRoot" type="t_reference"/>
  </xsd:all>
</xsd:complexType>
```

Figure 8-18: Excerpt from instance_data section of an ATFX exchange file; example of prostep ivip / VDA FDX application model for the rubber mount, for application element UnitUnderTestr

The excerpt of the XML schema of an ATFX exchange file in Figure 8 19 shows the schema element of application element MeasurementOrder in section instance_data:

```
<xsd:complexType name="t_MeasurementOrder">
  <xsd:all>
    <xsd:element minOccurs="1" name="Id" type="xsd:long"/>
    <xsd:element minOccurs="1" name="Name" type="xsd:string"/>
    <xsd:element minOccurs="0" name="MimeType" type="xsd:string"/>
    <xsd:element minOccurs="1" name="CustomerIdentificationNumber" type="xsd:int"/>
    <xsd:element minOccurs="1" name="OrderNumber" type="xsd:string"/>
    <xsd:element minOccurs="1" name="UnitUnderTest" type="t_reference"/>
    <xsd:element minOccurs="1" name="TplUnitUnderTestComp" type="t_reference"/>
  </xsd:all>
</xsd:complexType>
```

Figure 8-19: Excerpt from instance_data section of an ATFX exchange file; example of prostep ivip / VDA FDX application model for the rubber mount, for application element MeasurementOrder

The excerpt from an ATFX exchange file in Figure 8 20 is from an anonymized measurement order of the prostep ivip / VDA FDX application model for the rubber mount and shows application element UnitUnderTest in section instance_data:

```
<instance_data>
  <UnitUnderTest>
    <Id>10004</Id>
    <Name>ABCDEFGH</Name>
    <MimeType>application/x-asam.aoany.aounitundertest.UnitUnderTest</MimeType>
    <MeaResults>10032 10094 10151</MeaResults>
    <MeasurementOrder>10008</MeasurementOrder>
    <TestSteps>10137</TestSteps>
    <TplUnitUnderTestRoot>1</TplUnitUnderTestRoot>
  </UnitUnderTest>
</instance_data>
```

Figure 8-20: Excerpt from instance_data section of an ATFX exchange file; example of prostep ivip / VDA FDX application model for the rubber mount, for application element UnitUnderTest

The excerpt from an ATFX exchange file in Figure 8 21 is from an anonymized measurement order of the prostep ivip / VDA FDX application model for the rubber mount and shows application element MeasurementOrder in section instance_data:

```
<instance_data>
  <MeasurementOrder>
    <Id>10008</Id>
    <MimeType>application/
      x-asam.aoany.aounitundertestpart.MeasurementOrder</MimeType>
    <CustomerIdentificationNumber>123456789</CustomerIdentificationNumber>
    <OrderNumber>123456789</OrderNumber>
    <UnitUnderTest>10004</UnitUnderTest>
    <TplUnitUnderTestComp>7</TplUnitUnderTestComp>
  </MeasurementOrder>
</instance_data>
```

Figure 8-21: Excerpt from instance_data section of an ATFX exchange file; example of prostep ivip / VDA FDX application model for the rubber mount, for application element MeasurementOrder










Editing of ASAM ATFX data exchange format without changing XML schema

For the editing of the ASAM ATFX data exchange format outside the XML schema, a number of Java classes have been made available ([openATFX download | SourceForge.net](#)). These Java classes can be used to edit any ASAM ATFX file, completely independently of the openMDM application model. This requires however an IDL interface definition file ([ASAM ODS - Standards](#)) that must be translated into a Java archive.

8.9 Appendix I: Conventions for UML diagrams and associated texts

This appendix is an addendum to chapters "Extensions of openMDM application model" and "Relationship between ASAM ODS base model, openMDM application model and prostep ivip / VDA FDX application model - example of rubber mount" of prostep ivip / VDA Recommendation PSI 20.

The following conventions and associated texts apply to UML:

- In the class and object diagrams, the classes and objects of the prostep ivip / VDA FDX data model, the ASAM ODS base model, the prostep ivip / VDA FDX application model and the openMDM application model are arranged in 2 or 3 lanes, each representing a separate model.
- For better readability, the classes and objects are also color-coded: beige  for prostep ivip / VDA FDX data model; grey  for ASAM ODS base model; light blue  for prostep ivip / VDA FDX application model; red , blue , purple , turquoise , yellow  and green  for openMDM application model.
- Inheritance from one class to another is denoted with a line featuring an outline triangle at the end. The arrow points to the more general class.
- An association between two classes is denoted with a line.
 - An association might be defined in more detail (role, multiplicity, additional symbols).
 - To simplify the diagram, the end points of the associations are not depicted, as they are normally named after the classes linked through association (in the example below, the end points would be called "Structure Level" and "Project" respectively).
 - An exception is made for situations where two classes are related by more than one association, and in the case of reflexive associations where one class refers back to itself. In these situations, it is necessary to clearly distinguish between the different association ends, which are therefore depicted.
 - The multiplicity of an association is denoted by integers or intervals (n_1 ; $n_1..n_2$, etc. where n is an integer; * = unlimited). The multiplicity at one association end indicates the number of instances that can be assigned to the object at the other association end.

Caution: This convention does not conform fully to the UML standard language. In contrast to the UML standard, the multiplicities in the diagrams are "mirrored".
- The multiplicity of an association characterizes the various constellations in which the associated objects relate to each other:
 - Multiplicities 0..* and 1 are generally UML constellations describing a composition and its parts. In other words, an object does exist independent of the composition.
To denote this clearly, the association end with multiplicity 0..* also features a filled diamond, and the association is described with term "includes".
Example: The association between classes *Project* and *StructureLevel* with multiplicities 0..* and 1 respectively denotes that 1 instance of class *Project* can be assigned to 0, 1, or * instances of class *StructureLevel*, and that 1 instance of class *StructureLevel* is assigned to exactly 1 instance of class *Project*. In other words, one instance of class *Project* relates to 0..* instances of class *StructureLevel*.
In this document, we would then speak about a relationship where a subordinate application element is a component of the application element at the next higher level.
 - A special type of this constellation are associations with multiplicities of 0..* and 0..1 respectively, where one class is associated with multiple other classes, and one object of the first class must be a subordinate to exactly one object from another class. Again, the association ends with multiplicity 0..* are marked with a filled diamond. An example for such a constellation can be found in chapter 4.3.1 for the control of the visibility of components and attributes. Application element *ComponentVisibility* must belong to exactly one application element *TplUnitUnderTestComp*, *TplTestSequenceComp* or *TplTestEquipmentComp*. Accordingly, one *AttributeVisibility* must belong to exactly one application element *TplUnitUnderTestAttr*, *TplTestSequenceAttr* or *TplTestEquipmentAttr*.

- o There are also constellations with multiplicities of 0..* and 0..1, denoting an association between two objects that exist independently. In UML class diagrams, such associations are not particularly highlighted, and their labels read "refers to" and "uses".
Example: The association between classes *TestStep* and *UnitUnderTest* with multiplicities 0..1 and 0..* respectively means that 1 instance of class *TestStep* can be assigned to 0 or 1 instance of class *UnitUnderTest*, and that 1 instance of class *UnitUnderTest* can be assigned to 0, 1 or * instances of class *TestStep*. In other words, one instance of class *TestStep* refers to 0..1 instance of class *UnitUnderTest*, and one instance of class *UnitUnderTest* can be used by 0..* instances of class *TestStep*.
- o Objects can also be linked by attributed n:m relationships. In this case, there are two compositions connecting three classes. The multiplicities of the first and second class are 0..* and 1 respectively, while the multiplicities of the second and third classes are 1 and 0..*. A subordinate object of the second class can therefore only exist in the context of two objects of the first and third class. Such a relationship exists for example in Appendix F: Model-driven aspects of openMDM application model for the selection of the test steps for a test project by means of application elements *TplTest*, *TplTestUsage* and *TplTestStep*. The subordinate object is application element *TplTestUsage*, and the two related objects at a higher level are application elements *TplTest* and *TplTestStep*.
- The color codes used for the classes in the diagram below correspond to those in the openMDM overview in appendix B.
- For packages *Administration*, *Descriptive Data and Measurement*, the organization of the classes is not shown in the way it is done for in the overviews of the ASAM ODS (see **Figure 4-3**) and the openMDM application model (appendix B).
- In this document, the term "application element" refers to an instance of the respective class.



prostep ivip association

Dolivostraße 11
64293 Darmstadt
Germany

Phone +49-6151-9287336
Fax +49-6151-9287326
psev@prostep.com
www.prostep.org

ISBN 978-3-9820795-7-8
Version 1.1, 2020-2