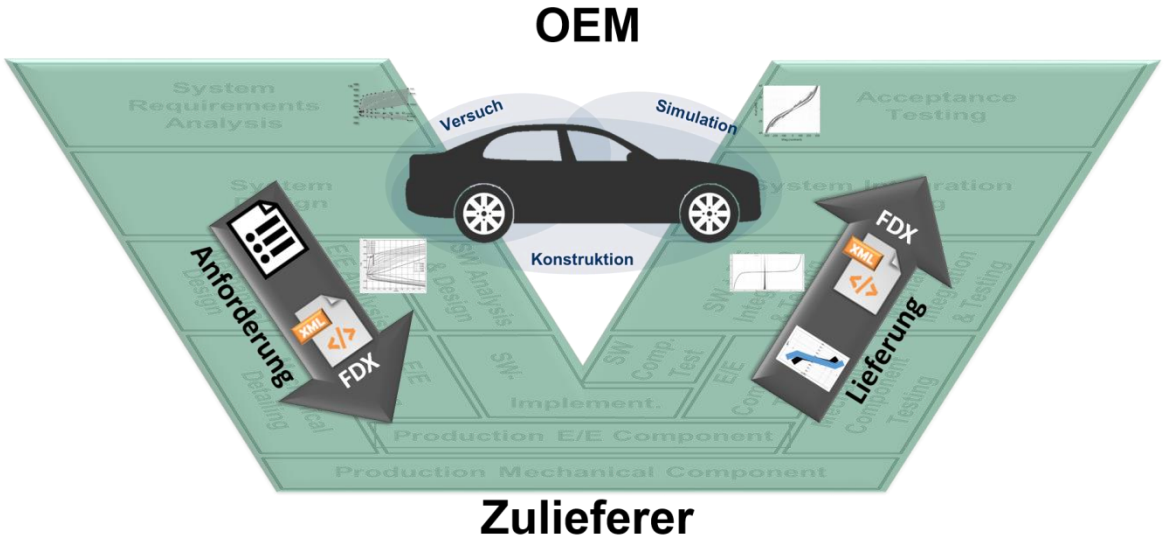
VDA Verband der Automobilindustrie

Automatisierter Funktionsdatenaustausch in der Automobilindustrie

VDA 5550

Version 1.0, März 2018

Arbeitskreis FDX



Verfahrensbeschreibung

Diese VDA Empfehlung wurde vom Arbeitskreis FDX entwickelt, um ein einheitliches Format und eine maschinenlesbare Spezifikation des Datenmodells/-formates für die Übertragung von Funktionsdaten (z.B. Kennfelder, Kennlinien, Kennwerte) zwischen Kunde und Lieferanten zu definieren.

Sie ermöglicht die Anwendung der strukturierten Datenübertragung. Mit dieser Empfehlung soll die einheitliche und effiziente Implementierung dieser Prozesse in der deutschen Automobilindustrie ermöglicht werden.

Ziele:

- Harmonisierung des Datenaustausches von Funktionsdaten zwischen OEMs und Zulieferern
- Verbesserung der Qualität und Verfügbarkeit von Funktionsdaten für CAE/Simulation
- Verringerung der Diskrepanz zwischen Bestellung und Lieferung von Funktionsdaten
- Steigerung der automatisierten Datenerzeugung, des Datenaustausches und der Datennutzung

Hinweis zur Version 1.0

Die VDA Empfehlung definiert das FDX Format. Die vorliegende Version beschreibt einheitliche Regeln für die Erstellung kompatibler Software Tools. Gegenwärtig laufen parallele Entwicklungen zur Erstellung spezialisierter Software Tools, die dieses Format unterstützen. Nach Abschluss einer Referenzimplementierung wird die VDA Empfehlung durch die notwendigen technischen Artefakte sowie Teil 2 und Teile 3-1 bis 3-n dieser VDA Empfehlung ergänzt, die eine Anwendung in der Praxis ermöglichen. Mit Implementierung der Software Tools ergeben sich ggf. Präzisierungen dieser VDA Empfehlung, die zeitnah veröffentlicht werden.

Haftungsausschluss

Die VDA-Empfehlungen sind Empfehlungen, die jedermann frei zur Anwendung stehen. Wer sie anwendet, hat für die richtige Anwendung im konkreten Fall Sorge zu tragen.

Sie berücksichtigen den zum Zeitpunkt der jeweiligen Ausgabe herrschenden Stand der Technik. Durch das Anwenden der VDA-Empfehlungen entzieht sich niemand der Verantwortung für sein eigenes Handeln. Jeder handelt insoweit auf eigene Gefahr. Eine Haftung des VDA und derjenigen, die an den VDA-Empfehlungen beteiligt sind, ist ausgeschlossen.

Sollten Sie bei der Anwendung der VDA-Empfehlung auf Unrichtigkeiten oder die Möglichkeit einer unrichtigen Auslegung stoßen, bitten wir Sie darum, dies dem VDA umgehend mitzuteilen, damit etwaige Mängel beseitigt werden können.

Herausgeber	Verband der Automobilindustrie e.V. (VDA) Behrenstraße 35, 10117 Berlin www.vda.de Diese Empfehlung wurde vom Arbeitskreis FDX erarbeitet.
Copyright	Verband der Automobilindustrie e.V. (VDA) Nachdruck und jede sonsige Form der Vervielfältigung ist nur mit Angabe der Quelle gestattet.
Stand	März 2018
Version	Version 1.0

Inhaltsverzeichnis

1 Einleitung

1.1 Hintergrund

Reduzierte Anzahl von Hardware-Erprobungsträgern, Verkürzung von Entwicklungszeiten und Beherrschung der zunehmend komplexer werdenden Produkte erfordern eine konsequente Forcierung der digitalen Fahrzeugentwicklung und Orientierung am durchgängigen Vorgehen für Systems Engineering. Die Entwicklung eines Gesamtfahrzeuges wird heute über den gesamten Entwicklungszeitraum beginnend von der Konzept- bis zur Hardwarephase durch verschiedene Konstruktions- und Simulationsgewerke unterstützt und begleitet. Diese Gewerke benötigen u.a. eine gesamthafte und konsistente, funktionale Bauteilbeschreibung durch Funktionsdaten.

Funktionsdaten sind gemessene, berechnete oder geschätzte skalare Werte, Kennlinien und Kennfelder, welche die Eigenschaften von Bauteilen beschreiben und für den Einsatz in der Produktentwicklung benötigt werden.

Für die digitale Transformation der Produktentwicklung ist die Verfügbarkeit der relevanten Daten in standardisierter, elektronischer Form eine Grundvoraussetzung. Eine automatisierte Datenbearbeitung und beispielsweise die Modellparametrierung für den effizienten Aufbau von Simulationsmodellen erfordern eine Standardisierung bzgl. Inhalt und Datenformat.

Der Datenaustausch erfolgt oft noch unstrukturiert in den verschiedensten Formaten. Erst ein einheitliches Datenaustauschformat ermöglicht eine weitere Automatisierung der Prozesse und erhöht deren Durchgängigkeit, Prozesssicherheit sowie Qualität.

1.2 Anwendung

Die vorliegende VDA Empfehlung beschreibt ein Datenmodell und ein Datenformat für den standardisierten und nachvollziehbaren Datenaustausch von Funktionsdaten und zugehörigen, relevanten Metadaten. Außerdem beschreibt sie die damit verbundenen Anwendungsfälle beim Funktionsdatenaustausch zwischen den Automobilherstellern und der Zulieferindustrie, sowie zwischen der Zulieferindustrie und ihren Vorlieferanten.

Der Fokus liegt dabei auf der effizienten Nutzbarkeit der relevanten Funktionsdaten für den virtuellen Entwicklungsprozess.

1.3 Zielsetzung

Ziel ist es, den gesamten Prozess von Anforderung, Austausch und Verarbeitung von Funktions-daten zu unterstützen und die Daten durchgängig in maschinenlesbarer Form bereitzustellen zu können.

Der Datenanforderer soll seine Anforderungen an die Funktionsdaten in diesem Format für den Datenlieferanten möglichst genau beschreiben können, ggf. unterteilt in Muss- und Kann-Daten.

Der Datenlieferant soll die Anforderungen möglichst interpretationsfrei verstehen können und auf dieser Basis seine Mess-/Simulationsdaten erzeugen. Das Format gibt dem Datenlieferanten vor, welche Daten als Muss- und Kann-Anforderungen zurück zu liefern sind.

Zur Absicherung der Qualität soll das Format und die Datenstruktur einfache programmtechnische Kontrollen zur Erkennung von fehlenden und fehlerhaften Einträgen ermöglichen und diese in einem Prüfsiegel festhalten können.

Am Ende soll ein kompletter Datensatz mit Anforderungen, zugehörigen Ergebnissen und Prüfsiegel stehen.

Durch die beschreibenden Attribute soll eine einfache Ablage der Daten in Datenmanagementsystemen wie PLM, PDM oder TDM ermöglicht werden. Ebenso soll eine Ausleitung und Befüllung der Daten automatisiert möglich sein.

Das Format soll internationalisierbar sein. Insbesondere soll es unterschiedliche Einheitensysteme unterstützen.

Mit dem gewählten Format soll eine Erweiterbarkeit für zukünftige Umfänge möglich sein. Somit können durch OEM und Lieferanten spezifische Informationsblöcke hinzugefügt werden.

Änderungen und Erweiterungen am Datenmodell sollen automatisch erkannt und verarbeitet werden, ohne dass die Softwaretools angepasst werden müssen.

Das Format soll die Möglichkeit eines file-basierten Datenaustausches zwischen IT-Werkzeugen bieten und somit eine hohe Automatisierbarkeit unabhängig von IT-Technologie und IT-Plattform ermöglichen.

Diese VDA Empfehlung soll dem Leser ermöglichen, das Datenmodell und das Datenformat zu verstehen und ggf. eigene Anwendungen auf dieser Basis zu entwickeln.

1.4 Einschränkungen

Diese VDA Empfehlung hat nicht zum Ziel, die OEM-seitigen inhaltlichen Anforderungen an die Funktionsdaten von Lieferanten zu vereinheitlichen, da hierbei Aspekte der OEM-individuellen Produktausprägung und Entwicklungsphilosophien, sowie die einhergehenden Vertraulichkeiten zu stark berührt würden.

Eine Speicherung der Funktionsdaten in binärer Form ist aufgrund der betrachteten Anwendungsfälle und des erwarteten Datenvolumens nicht vorgesehen, auch wenn das ASAM ATFX-Format grundsätzlich den Transport von binären Nutzdaten ermöglicht.

1.5 Das FDX-Datenformat

Ausgangspunkt für diese Empfehlung ist der vom ASAM e.V. (Association for Standardisation of Automation and Measuring Systems) entwickelte ASAM ODS (Open Data Services) Standard, das darauf basierende ASAM ATFX-Datenaustauschformat (XML) und das ebenfalls darauf basierende spezifische openMDM Applikationsmodell.

Das hier beschriebene Datenformat erweitert, ergänzt und konkretisiert diese Standards in drei Punkten:

1. Es erweitert das aktuelle openMDM4 Applikationsmodell
2. Es konkretisiert das openMDM4 Applikationsmodell durch Applikationsmodelle für konkrete Anwendungsfälle
3. Zusammenführung in einer Container-Datei, in der zusätzlich zur XML-Datei (ATFX-Datei), referenzierte Dateien und ggf. ein Prüfsiegel abgelegt werden.

Zusätzlich wird auch ein zwischen den beteiligten Mitgliedern empfohlener Basisumfang an auszutauschenden Funktionsdaten auf Bauteilebene beschrieben. Eine solche bauteilspezifische Vorlage für Funktionsdaten beinhaltet alle Informationen, die zur einheitlichen Formulierung der Anforderungen an einen Messauftrag und an die Datenlieferung benötigt werden

2 Aufbau dieser Empfehlung

Dieses Dokument ist so strukturiert, dass verschiedene Lesergruppen mit unterschiedlichem technischem Tiefgang eine geeignete Granularität an relevanten Informationen finden.

Grundsätzlich adressiert dieses Dokument die folgenden 3 Lesergruppen:

- fachliche Anwender und Entscheider (Bauteileverantwortlicher (BTV), Berechner, Prüfingenieur, …)
- technische Verantwortliche (Fachadministrator für Vorlagen, Prozessoptimierer, IT-Systemarchitekt, …)
- SW-Implementierer (Toolanbieter, Schnittstellenprogrammierer, Systemintegratoren, …)

Für eine leichtere Übersicht ist die Relevanz der verschiedenen Kapitel für die verschiedenen Lesergruppen tabellarisch dargestellt (Tabelle 2.1):

Tabelle 2-1 Relevanz der Kapitel für verschiedene Lesergruppen

Kapitel	Fachliche Anwender	Technische Verantwortliche	SW-Implementierer
1 Einleitung (Inhalte und Zielsetzung)	X	X	X
2 Aufbau dieser VDA Empfehlung	X	X	X
3 Anwendungsbereich und Beispiel	X	X	X
4 Technische Grundlagen (ASAM ODS, openMDM)		X	X
5 Konzeptioneller Aufbau FDX-Format	X	X	X
6 Mindestanforderungen an den Datenaustausch	X	X	X
7 Normative Referenzen		X	X
8 Glossar	X	X	X
8 Anhänge			X
Teil 2	X	X	X
Teil 3-n	X		

Die VDA Empfehlung strukturiert sich gemäß Darstellung in Abbildung 2-1 in das Hauptdokument (Teil 1 der Empfehlung) mit den allgemeinen Inhalten, Teil 2 mit übergreifenden, detailierteren Informationen zur Attributeliste und dem Datenmodell und den Teilen 3-n mit bauteilspezifischen Inhalten in Form von pdf-Dokumenten und Vorlagen, den ATFX-Dateien, die die bauteilspezifische Fachadministration enthalten. Detailinformationen sind in diverse Anhänge ausgegliedert. Die Gesamtübersicht für ein Bauteil bekommt man somit durch Teil 1, Teil 2 und dem jeweiligen Bauteil-spezifischen Teil 3-n.

Abbildung 2-1Struktur der VDA Empfehlung

Die bauteilspezifischen Vorlagen werden in einem VDA Repository verfügbar sein.

3 Anwendungsbereich

Die folgenden Anwendungsfälle beschreiben die Einsatzszenarien für das beschriebene Datenaustauschformat. Aus den Anwendungsfällen wurden die konkreten Anforderungen an das Datenaustauschformat identifiziert und in der beschriebenen Lösung umgesetzt.

Basierend auf den vier Hauptrollen Fachadministrator, Datenanforderer, Datenlieferant und Datennutzer (***Abbildung 3-1***), ergeben sich unterschiedliche Anwendungsfälle:

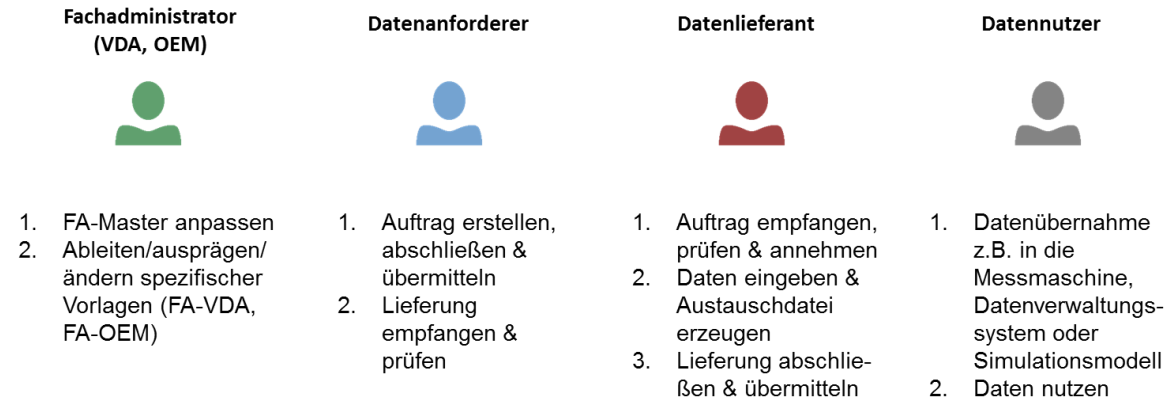


Abbildung 3-1 Hauptrollen für die Anwendung des FDX-Datenaustauschformates

Die Fachadministratoren haben im Wesentlichen die Aufgabe, Attribute und deren Eigenschaften im Datenmodell als Grundlage für die vereinheitlichte Beschreibung von Bauteilen und Bauteiltypen in das Datenmodell der Funktionsdatenaustauschdatei einzubringen. Damit wird für Datenanforderer und Datenlieferant der Rahmen der möglichen Informationen, welche gefordert bzw. geliefert werden, vorgegeben und standardisiert.

Der Datenanforderer nutzt die vorgegebenen Attribute und Eigenschaften, um ihnen nach den vorgegebenen Standards, konkrete Anweisungen, Regeln, Soll-Werte, Kennlinien etc. hinzuzufügen und diese nachvollziehbar beim Datenlieferant über das FDX-Austauschformat anzufordern. Basis dafür sind die von dem VDA AK FDX zur Verfügung gestellten bauteilspezifischen Vorlagen.

Der Datenlieferant ordnet den angeforderten Attributen und Eigenschaften entsprechend der Vorgaben des Datenanforderers konkrete Ist-Werte zu, sichert die Qualität der Informationen ab und übermittelt die Daten über das FDX-Austauschformat zurück an den Anforderer. Dieser kann nun ebenfalls eine Prüfung vornehmen und die Daten dann weiterverarbeiten.

Der Datennutzer übernimmt die Funktionsdaten aus dem FDX-Austauschformat in seinen Prozess bzw. in sein Datenverwaltungssystem. Beispiele wären die automatisierte Übernahme der Funktionsdaten in Simulationsmodelle/-Systeme oder die automatisierte Übernahme von Messgerätebetriebsgrößen in die Messsysteme/-maschinen. Das FDX-Austauschformat unterstützt dies durch eine umfassende und einheitliche Beschreibung aller relevanten Informationen.

Im Folgenden wird der Begriff Fachadministration (FA) in verschiedenen Konstellationen verwendet. Die Fachadministration enthält dabei das fachlich orientierte Datenmodell und dient als Vorlage für die weitere Instantiierung. Entsprechend des Einsatzzweckes werden verschiedene Fachadministrationen unterschieden. Folgende Darstellung veranschaulicht die Differenzierung der unterschiedlichen Ausprägungen.

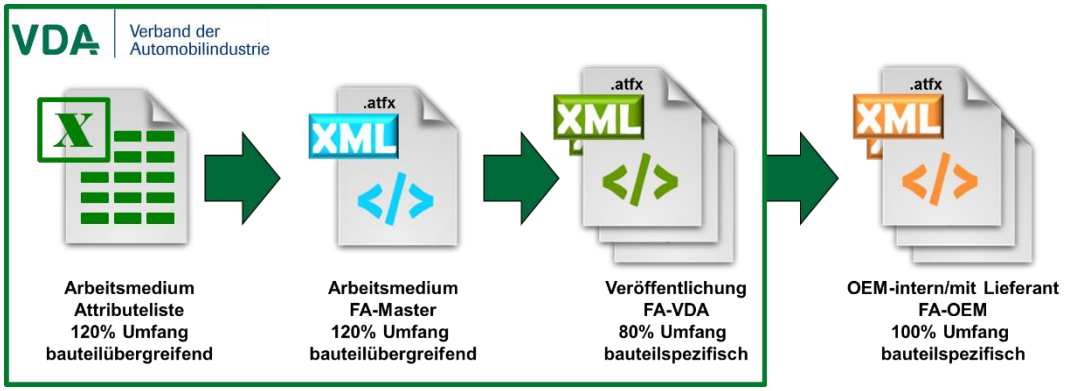


Abbildung 3-2 Anwendungsfall "FA-Master anpassen"

Hinweis:

Bei der Erarbeitung der Vorlagen wurde im Besonderen auf die Zuordnung des Status „Muss-“ oder „Kann-Attribut“ Wert gelegt. Es wird empfohlen, bei der Ableitung von konkreten Messaufträgen aus den Vorlagen durch den Datenanforderer, „Kann-Attribute“ nicht ohne zwingenden Grund in „Muss-Attribute“ zu ändern.

3.1 Anwendungsfall Fachadministrator: FA-Master anpassen (für bauteilspezifische Ergänzungen)

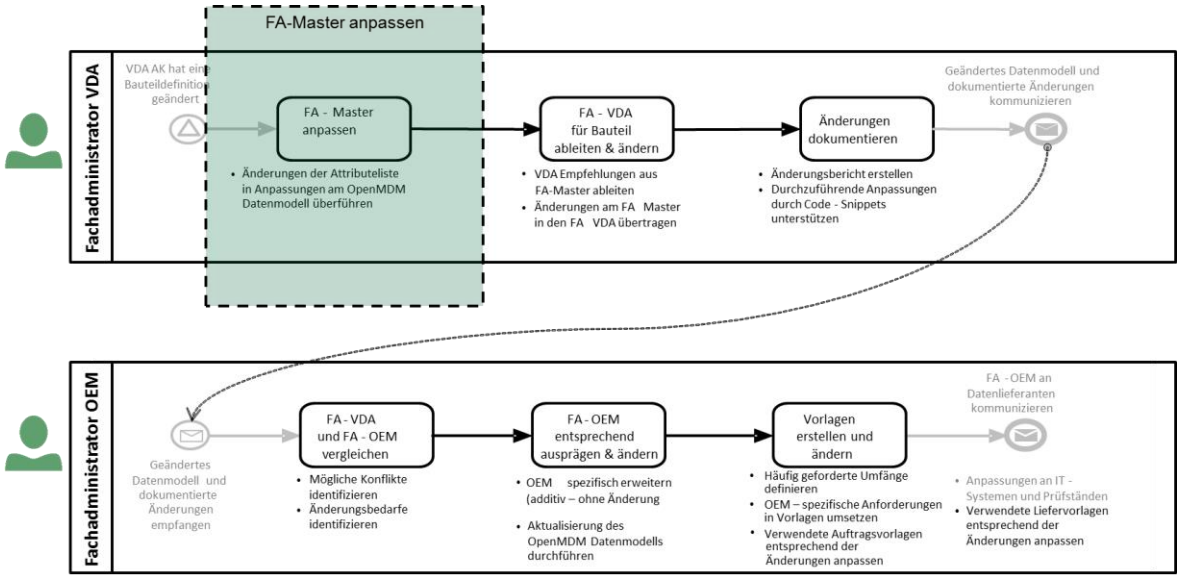


Abbildung 3-3 Anwendungsfall „FA-Master anpassen“

Tabelle 3-1 Anwendungsfall Fachadministrator: FA-Master erstellen und ändern

Beschreibung u. Zweck	Erstellen eines Fachadministration Masters (FA-Master), d.h. Erstellen von Beschreibungsattributen in den Kategorien Prüfling (Bauteil), Betriebsgrößen (Versuchsdurchführung) und Setup (Versuchsaufbau) und von Katalogkomponenten (Aggregate von Beschreibungsattributen). Dient zur Definition der Attribute und deren Eigenschaften als Grundlage für die vereinheitlichte Beschreibung verschiedener Bauteiltypen, so dass Lieferanten genaue Vorgaben erhalten, welche Informationen geliefert werden müssen. D.h. hiermit soll eine Vereinheitlichung der Beschreibung verschiedener Bauteiltypen erfolgen, da sich deren Beschreibungen auf dieselben FA-Master beziehen sollen.
Szenario	1. Erstellen 1.1. Der Anwender legt Katalogkomponenten (Attributgruppen) entsprechend den verschiedenen Kategorien und Subkategorien in der Attributeliste fest. 1.2. Der Anwender legt die Attribute innerhalb der Katalogkomponente an. Dabei werden die Eigenschaften entsprechend der Attributeliste festgelegt. 2. Änderungen der Attribute und deren Eigenschaften für bestehende Katalogkomponenten. 2.1. Katalogkomponente löschen

	2.2. Attribut löschen 2.3. Attribut anlegen 2.4. Attributeigenschaften ändern 3. Speichern: Der FA-Master wird auf einem geeigneten Medium, z.B. einer Festplatte als ATFX-Datei abgespeichert.
Ergebnis	Ein FA-Master als ATFX-Datei entsprechend der ASAM ODS ATFX und der openMDM Definition, welche die Katalogkomponenten und deren Attribute enthält.

3.2 Anwendungsfall Fachadministrator: FA-VDA für Bauteil ableiten und ändern

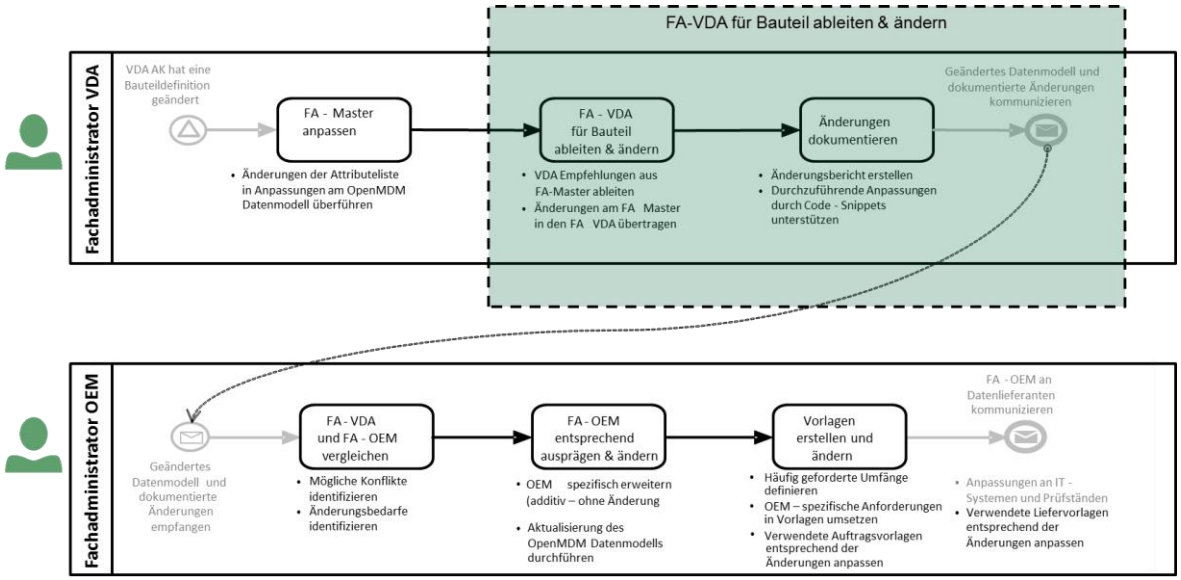


Abbildung 3-4 Anwendungsfall „FA-VDA für Bauteil ableiten und ändern“

Tabelle 3-2 Anwendungsfall FA-VDA für Bauteil ableiten und ändern

Beschreibung u. Zweck	Erstellen/Definieren und Ändern einer Fachadministration VDA (FA-VDA) d.h. einer Vorlage für einen bestimmten Bauteiltyp. Durch Selektion der Katalogkomponenten und Attribute aus dem FA-Master und durch das Vorbefüllen bestimmter Attribute sowie durch die Wahl der Kombination von Attributen wird eine FA-VDA für zukünftige Aufträge erstellt (d.h. sie dient als Vorlage in der die jeweils zu befüllenden Attribute für eine bestimmte Art von Aufträgen spezifiziert sind).
Szenario	1. Anlegen, Ändern und Löschen neuer hierarchische Gruppen (ähnlich wie Verzeichnisse) zur besseren Übersicht. 2. Anlegen, Ändern und Löschen von neuen openMDM Komponenten-Vorlagen unter Verwendung der Katalogkomponenten aus dem FA-Master. 3. Es besteht die Möglichkeit der Strukturierung, indem die neuen Komponenten als Unter-Komponenten von bestehenden Komponenten angelegt werden können. 4. Festlegung, ob die Komponente Optional, Default Aktiv und/oder Messreihenvariabel ist. 5. Festlegung der zu verwendenden Attribute für die Komponente.

	6. Festlegung von Attributeigenschaften, wie z.B. Vorbefüllung mit Werten, Schreibschutz, oder ob das Attribut optional ist. 7. Anlegen, Ändern und Löschen von neuen Messschritt-Vorlagen, ggf. unter Auswahl der zugehörigen Gruppe. 8. Zuweisung und Änderung der Zuweisung von Prüflings-Vorlagen, Betriebsgrößen-Vorlagen und einem Setup-Vorlagen zu den Messschritt-Vorlagen. 9. Anlegen und Löschen der FA-VDA (Versuchs-Vorlage), ggf. unter Auswahl der zugehörigen Gruppe. 10. Zuweisung und Änderung der Zuweisung (inkl. Löschung) von beliebig vielen Messschritt-Vorlagen zur FA-VDA (Versuchs-Vorlage). 11. Speichern: Der FA-VDA wird auf einem geeigneten Medium, z.B. einer Festplatte als ATFX-Datei abgespeichert.
Ergebnis	Die Fachadministration-VDA ist für den Bauteiltyp in Form einer FA-VDA erstellt oder geändert und syntaktisch korrekt.

3.3 Anwendungsfall Fachadministrator: FA-OEM für Bauteil ausprägen und ändern

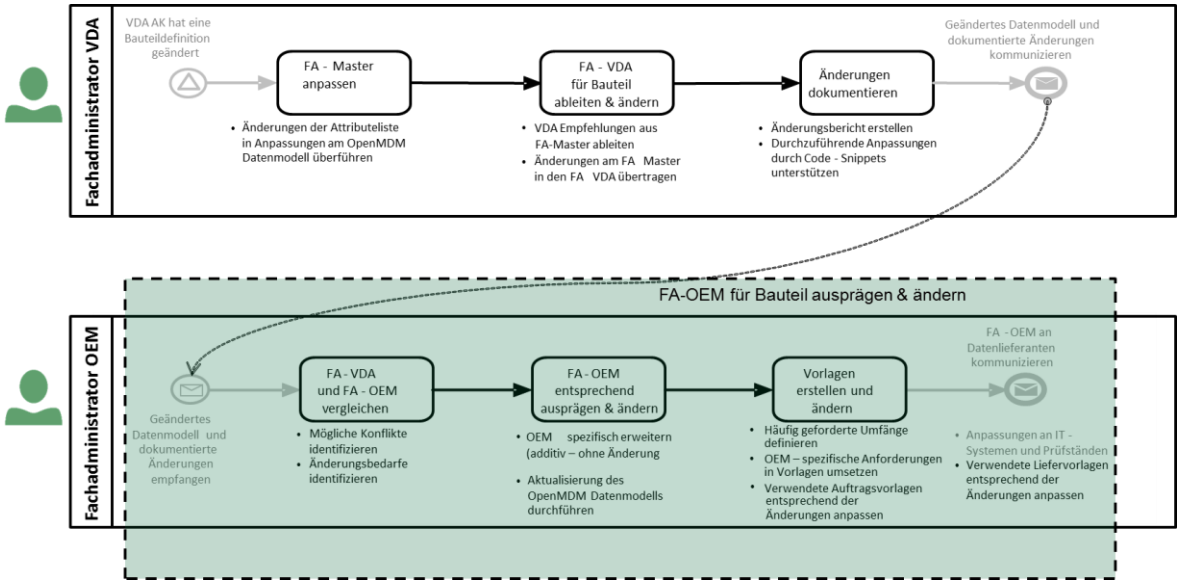


Abbildung 3-5 Anwendungsfall „FA-OEM für Bauteil ausprägen und ändern“

Tabelle 3-3 Anwendungsfall „FA-OEM für Bauteil ausprägen und ändern“

Beschreibung u. Zweck	Datenmodell des Austauschformats auf Basis der FA-VDA mit OEM-spezifischen Änderungen/Ergänzungen implementieren. Aktualisierungen an dem daraus entstehenden Format vornehmen.
Szenario	A: Analog: FA-VDA 1. Anlegen, Ändern und Löschen neuer hierarchische Gruppen (ähnlich wie Verzeichnisse) zur besseren Übersicht. 2. Anlegen, Ändern und Löschen von neuen openMDM Komponenten-Vorlagen unter Verwendung der Katalogkomponenten aus der FA-VDA.

	3. Es besteht die Möglichkeit der Strukturierung, indem die neuen Komponenten als Unter-Komponenten von bestehenden Komponenten angelegt werden können. 4. Festlegung ob die Komponente Optional, Default Aktiv und/oder Messreihenvariabel ist. 5. Festlegung der zu verwendenden Attribute für die Komponente. 6. Festlegung von Attributeigenschaften, wie z.B. Vorbefüllung mit Werten, Schreibschutz, oder ob das Attribut optional ist. 7. Anlegen, Ändern und Löschen von neuen Messschritt-Vorlagen. Ggf. unter Auswahl der zugehörigen Gruppe. 8. Zuweisung und Änderung der Zuweisung von Prüflings-Vorlagen, Betriebsgrößen-Vorlagen und einem Setup-Vorlage zu den Messschritt-Vorlagen. 9. Anlegen und Löschen der FA-OEM (Versuchs-Vorlage), ggf. unter Auswahl der zugehörigen Gruppe. 10. Zuweisung und Änderung der Zuweisung (inkl. Löschung) von beliebig vielen Messschritt-Vorlagen zur FA-OEM (Versuchs-Vorlage). 11. Speichern: Der FA-OEM wird auf einem geeigneten Medium gespeichert. B: Analog: FA-Master: 1. Erstellen 1.1. Der Anwender legt Katalogkomponenten (Attributgruppen) fest. 1.2. Der Anwender legt die Attribute innerhalb der Katalogkomponente an. 2. Änderungen der Attribute und deren Eigenschaften für bestehende Katalogkomponenten. 2.1. Katalogkomponente löschen 2.2. Attribut löschen 2.3. Attribut anlegen 2.4. Attributeigenschaften ändern
Ergebnis	Eine FA-OEM als ATFX-Datei entsprechend der ASAM ODS ATFX und der openMDM Definition, welche die Katalogkomponenten und deren Attribute enthält. OEM spezifische Fachadministration und Vorlagen für Bauteiltyp basierend auf der FA-VDA.

3.4 Anwendungsfall Datenanforderer: Auftrag erstellen, abschließen und übermitteln

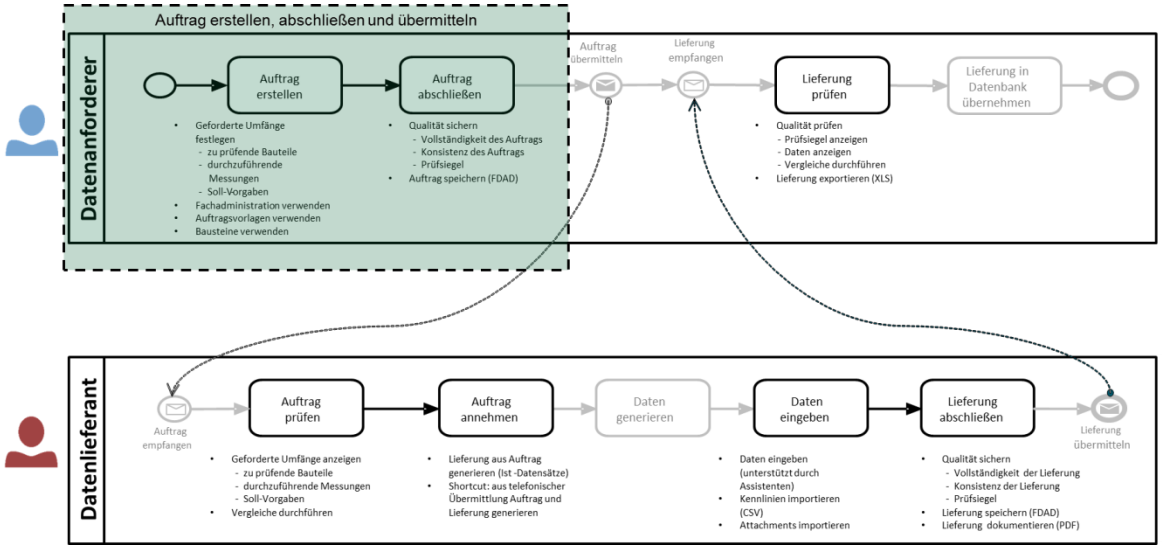


Abbildung 3-6 Anwendungsfall „Auftrag erstellen, abschließen und übermitteln“

Tabelle 3-4 Anwendungsfall Datenanforderer: Auftrag erstellen, abschließen und übermitteln

Beschreibung u. Zweck	Der Datenanforderer legt die geforderten Informationen (ordnet den Attributen Wert oder wählt aus einer Werteliste aus) in der ATFX-Datei fest. Danach schließt er den Auftrag ab, indem er die inhaltliche Qualität der ATFX-Datei mittels Prüfsiegel(n) dokumentiert, die ATFX-Datei zusammen mit der Prüfsiegeldatei und falls erwünscht, mit weiteren referenzierten Dokumenten (Bilder, pdf, etc.) im Container der Funktionsdatenaustauschdatei (FDAD) zusammenführt. Zum Schluss wird der Auftrag per Datenübertragung zum Datenlieferanten übermittelt.
Szenario	1. Festlegung der folgenden Umfänge unter Verwendung von Fachadministration, Auftragsvorlagen, Bausteinen und direkter Eingabe von Daten: 1.1. Messauftragsdaten (z.B. Auftragsstatus, Auftragsnummer) 1.2. Auftragsmetadaten (z.B. Lieferant, Konstruktionsstand, Teilebeschreibung, Gewicht) 1.3. Datensatzmetadaten (z.B. Verwaltungsinformationen, Datensatzidentifikation, Lastenheftinformationen) 1.4. Bauteilspezifische Zusatzinfos (z.B. Bauteilmaterial, Bauteilabmessungen, Bauteilausprägung) 1.5. Prüflingsmetadaten erfassen (Bauteil-/Qualitätsstatus, Teileidentifikation, -beschreibung, Vorerprobung, Bauteil Modifikation) 1.6. Messgerätesetup: (z.B. Datenerzeugung: Berechnung/Schätzung/ Prüfstand, Prüfstandsadaption, Bauteilspezifische Adaption) 1.7. Messgerätebetriebsgrößen: (z.B. Administratives, Versuchstyp, Vorlast, Vorkonditionierung, Messprogramm, Umgebungsparameter) 1.8. Soll Funktionsdaten (z.B. Massegeometrische Eigenschaften, Statische Messung, Dynamische Messung, Skalare Kennwerte, Kennlinien, Abgeleitete Kenngrößen (z.B. Steigung)) 2. Abschluss des Auftrags:

	2.1. Erstellung Prüfsiegel für einzelne Datensätze 2.2. Erstellung Prüfsiegel für die Gesamtdatei 2.3. Speicherung des gesamten Umfanges in der Funktionsdatenaustauschdatei 3. Übermittlung der FDAD an den Lieferanten
Ergebnis	Geprüfter, vollständiger Messauftrag wurde dem Datenlieferanten übermittelt

3.5 Anwendungsfall Datenlieferant: Auftrag empfangen, prüfen und annehmen

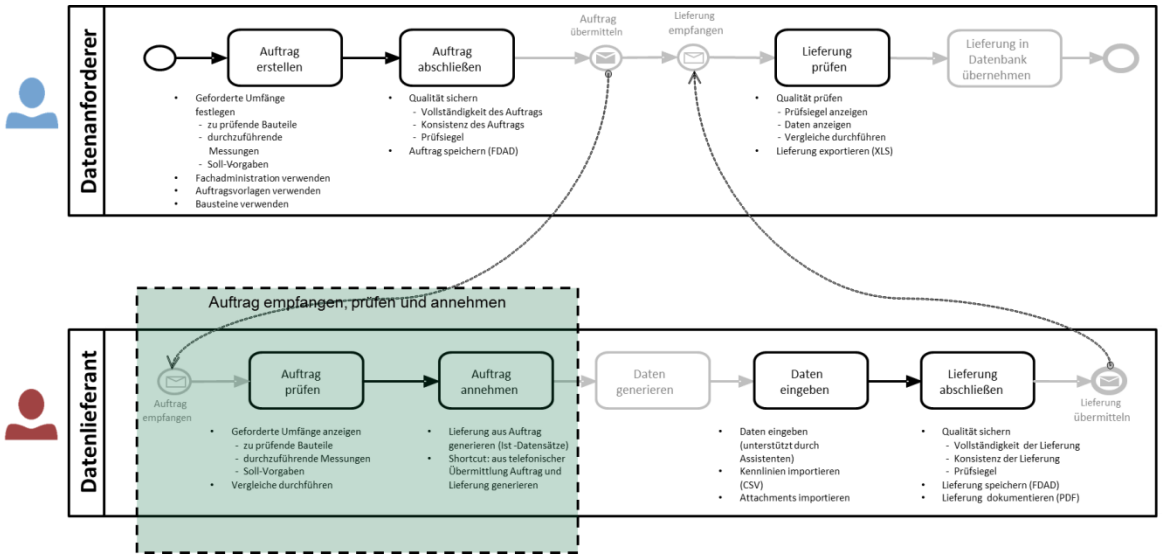


Abbildung 3-7 Anwendungsfall „Auftrag empfangen, prüfen und annehmen“

Tabelle 3-5 Anwendungsfall Datenlieferant: Auftrag empfangen, prüfen und annehmen

Beschreibung u. Zweck	Der Datenlieferant empfängt eine Funktionsdatenaustauschdatei mit dem Messauftrag. Der Datenlieferant öffnet die Datei mit einem SW-Tool, prüft den empfangenen Auftrag und nimmt ihn an. Anschließend generiert der Datenlieferant eine neue Datei, in der er die Bestelldaten um die Lieferdaten ergänzt.
Szenario	1. Der Datenlieferant prüft den Auftrag: 1.1. Er lädt die Funktionsdatenaustauschdatei 1.2. Er validiert das Prüfsiegel um sicherzustellen, dass mit der letzten Qualitätsprüfung keine Änderungen an der FDAD vorgenommen wurden. 1.3. Er visualisiert die Daten, welche im Szenario des vorhergehenden Anwendungsfalls (Datenanforderer: Auftrag erstellen, abschließen und übermitteln) beschrieben sind, um sich einen Überblick und Verständnis über den gegenwärtigen Stand der Befüllung eines/der Datensatzes/Datensätze zu verschaffen und den Auftrag zu bewerten. Hierzu kann er unter anderem: • Skalare Werte tabellarisch darstellen

	• Messkurven grafisch darstellen • Messkurven grafisch vergleichen • Skalare Werte tabellarisch vergleichen 2. Der Datenlieferant nimmt den Auftrag an und generiert aus den Soll-Datensätzen zugehörige Ist-Datensätze, welche mit den Werten der Funktionsmetadaten aus dem Soll-Datensatz vorbefüllt sind. 2.1. Er stellt sicher, dass bei der Neuanlage das Datenmodell resp. der Versionsstand des Datenmodells für seine Datensätze identisch mit dem des Anforderers ist. 2.2. Er übernimmt die Auftragsinformationen aus den Datensätzen des Anforderers 2.3. Er setzt den Auftragsstatus auf Lieferung und aktuell
Ergebnis	Der Messauftrag des Datenanforderers ist gesichtet, bewertet und angenommen. Die Liefer-Datensätze sind zur Befüllung mit den Ist-Werten vorbereitet.

3.6 Anwendungsfall Datenlieferant: Daten eingeben, Lieferung abschließen und übermitteln

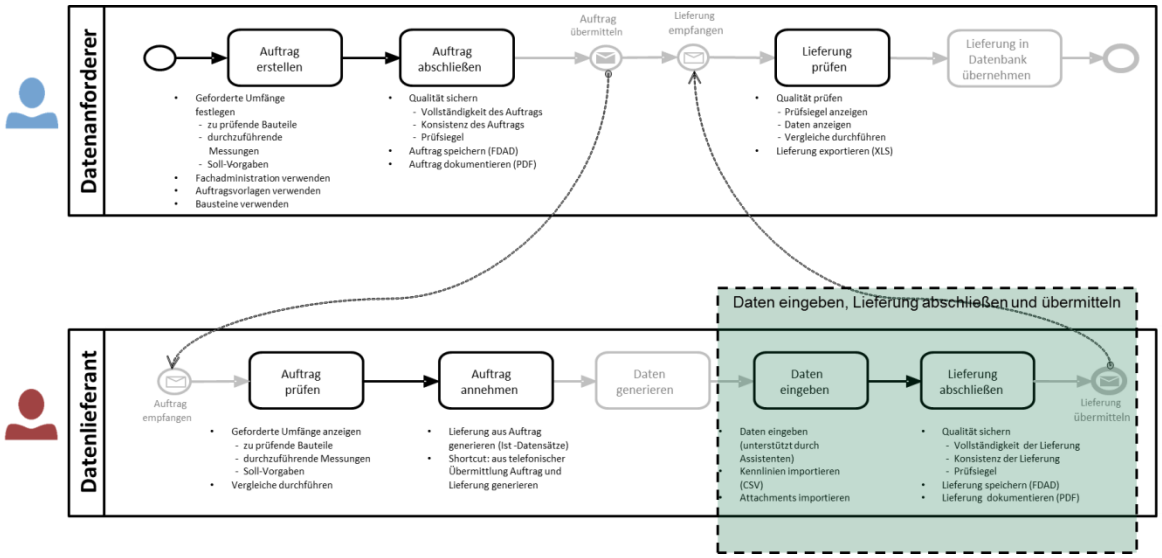


Abbildung 3-8 Anwendungsfall „Daten eingeben, Lieferung abschließen und übermitteln“

Tabelle 3-6 Anwendungsfall Datenlieferant: Daten eingeben, Lieferung abschließen und übermitteln

Beschreibung u. Zweck	Der Datenlieferant gibt alle relevanten Informationen gemäß Auftrag des Datenanforderers ein, d.h. er trägt zu den Soll-Abfragen, die entsprechenden Ist-Werte ein. Danach sichert er die Qualität ab, in dem er die Vollständigkeit und Konsistenz der Lieferung überprüft und mit Prüfsiegel dokumentiert (alle Pflichtattribute vorhanden, alle angeforderten Datensätze vorhanden, alle vorgegebene Wertebereiche eingehalten?).

	Die Lieferung wird dann in der Funktionsdatenaustauschdatei gespeichert und an den Datenanforderer übermittelt.
Szenario	1. Ergänzen/Befüllen der folgenden Umfänge mit **Ist-Werten** unter Verwendung von Vorlagen, Bausteinen und direkter Eingabe von Daten: 1.1. Messauftragsdaten (z.B. Auftragsstatus) 1.2. Auftragsmetadaten (z.B. Lieferant, Konstruktionsstand, Teilebeschreibung) 1.3. Datensatzmetadaten (z.B. Verwaltungsinformationen, Datensatzidentifikation, Lastenheftinformationen) 1.4. Bauteilspezifische Zusatzinfos (z.B. Bauteilmaterial, Bauteilabmessungen, Bauteilausprägung) 1.5. Prüflingsmetadaten erfassen (Bauteil-/Qualitätsstatus, Teileidentifikation, -beschreibung, Vorerprobung, Bauteil Modifikation) 1.6. Messgerätesetup: (z.B. Datenerzeugung: Berechnung/Schätzung/ Prüfstand, Prüfstandsadaption, Bauteilspezifische Adaption) 1.7. Messgerätebetriebsgrößen (Administratives, Versuchstyp, Vorlast, Vorkonditionierung, Messprogramm, Umgebungsparameter) 1.8. Ist Funktionsdaten (z.B. Massegeometrische Eigenschaften, Statische Messung, Dynamische Messung, Skalare Kennwerte, Kennlinien, Abgeleitete Kenngrößen (z.B. Steigung)) 1.9. Attachements: Der Anwender kann zu Dokumentationszwecken an den Auftrag ein Attachment anhängen. Dies kann z.B. ein Bild oder eine Zeichnung sein. 2. Abschluss der Lieferung: 2.1. Erstellung Prüfsiegel für einzelne Datensätze 2.2. Erstellung Prüfsiegel für die Gesamtdatei 2.3. Speicherung des gesamten Umfanges in der Funktionsdatenaustauschdatei 3. Übermittlung der FDAD an den Datenanforderer
Ergebnis	Befüllte Funktionsdatenaustauschdatei mit Prüfsiegel wurde an den Datenanforderer übermittelt.

3.7 Anwendungsfall Datennutzer: OEM-interne oder Lieferanten-interne Speicherung der Messdaten mithilfe des ATFX-Formates

Tabelle 3-7 Anwendungsfall Datennutzer: OEM-interne oder Lieferanten-interne Speicherung der Messdaten mithilfe des ATFX-Formates

Beschreibung u. Zweck	Automatisierte Übernahme der Daten und Informationen aus der FDAD oder der ATFX-Datei in Folgeprozesse. Effiziente, fehlerfreie Datenübernahme in firmeninterne Prozesse.
Szenario	1. Der Datenanforderer erhält die FDAD und überprüft deren Qualität. 2. Der Datenanforderer speichert die FDAD in das Datenmanagementsystem

	und gibt diese frei. 3. Der Datennutzer lädt sich die relevanten Informationen aus der FDAD im Datenmanagementsystem in seine Maschine, sein System oder sein Modell
Ergebnis	Die für den Anwendungsfall des Datennutzers relevanten Daten aus der FDAD/ATFX-Datei sind fehlerfrei in die Maschine, das System oder das Modell des Datennutzers übertragen

3.8 Fachliches Konzept und konkretes Beispiel

In diesem Kapitel werden an einem konkreten Beispiel ausgewählte Inhalte und deren Struktur im Datenmodell vorgestellt. Wegen der besseren Lesbarkeit sind die Inhalte und Strukturen in Tabellen dargestellt, die von der Abbildung in der FDX-XML-Austauschdatei deutlich abweichen können.

Die im Folgenden dargestellten und verwendeten Feldnamen und Anzeigenamen der Attribute sowie die dargestellten Werte sind beispielhaft. Feldnamen und Anzeigenamen können einer fortlaufenden Änderung unterliegen und werden in diesem Dokument nicht zwingend zeitnah nachgeführt. Insbesondere bei den Blockregeln werden tlw. englische oder die Begriffe aus dem openMDM und ASAM Standards verwendet.

3.8.1 Fachliches Konzept und Attributekategorien

Im Folgenden wird die fachliche Struktur des FDX-Formates kurz erläutert. Zur besseren Übersichtlichkeit werden die Attribute in drei Strukturebenen unterteilt:

- Hauptkategorien
  - Subkategorien
    - Attribute

D.h. den Hauptkategorien sind meist mehrere Subkategorien zugeordnet und diesen wiederum mehrere Attribute.

Hauptkategorie	Subkategorie	Attribut	Wertebereich	Feldtyp	Einheitentyp	Blockregel
Messgeräte-betriebsgrößen	Administrative Information	Prüfprogramm-Name	vom Benutzer anzugeben	Text	dimensionslos	
	Messungstyp	Messung Bewegungsrichtung	[translatorisch; rotatorisch]	Text	dimensionslos	
		Messung Raumrichtung	[x; y; z]	Text	dimensionslos	
		Messung mit Vorlast	[ja; nein]	Text	dimensionslos	
		Vorkonditionierung	[ja; nein]	Text	dimensionslos	
	Vorlast	Vorlast Bewegungsrichtung	[translatorisch; rotatorisch]	Text	dimensionslos	Messgeräte
		Vorlast Raumrichtung	[x; y; z]	Text	dimensionslos	Messungsty
		Vorlast Steuerungsart	[-;weggesteuert; kraftgesteuert]	Text	dimensionslos	Vorlast= 'ja'
		Vorlast Weg	vom Benutzer anzugeben	Dezimal	Länge	

Abbildung 3-9 Ausschnitt der Hauptkategorie Messgerätebetriebsgrößen mit verschiedenen Sub-kategorien und Attributen

Die oberste Ebene bilden die folgenden neun Hauptkategorien:

- Messauftrag
- Auftragsmetadaten
- Bauteilspezifische Zusatzinformationen
- Datensatzmetadaten
- Prüfling
- Messgerätesetup
- Messgerätebetriebsgrößen
- Funktionsdatum
- abgeleitete Kenngrößen

Abbildung 3-10 stellt das fachliche Datenmodell auf Ebene dieser Hauptkategorien dar:

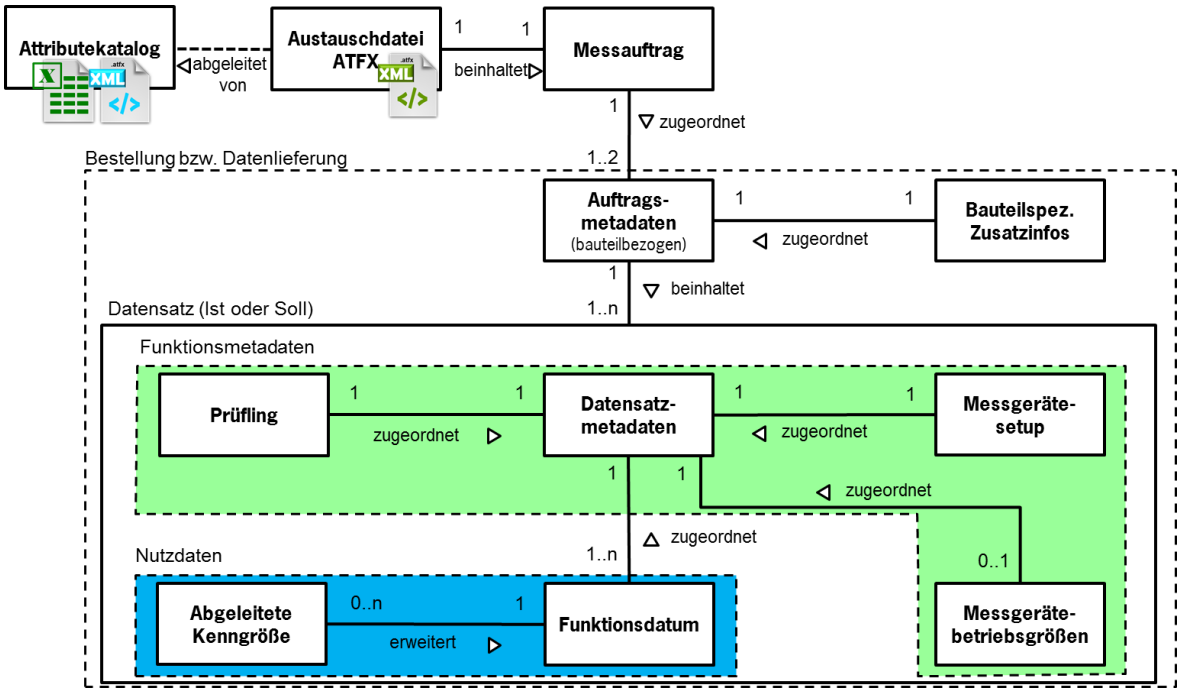


Abbildung 3-10 Struktur Fachkonzept Applikationsmodell

Ein *Messauftrag* wird von einem generischen Attributekatalog, der alle möglichen Attribute-kategorien und Inhalte enthält, abgeleitet. D.h. der Ersteller kennzeichnet bzw. befüllt in der Vorlage nur die Attribute, welche für seinen konkreten Anwendungsfall relevant sind. Eine vollständige Beschreibung aller möglichen Attribute befinden sich in Teil 2 und den Teilen 3-x der VDA Empfehlung.

Dem *Messauftrag* sind, je nach Status, nur die Daten zur Bestellung oder die Daten zur Bestellung und Datenlieferung zugeordnet. Dies wird im **Abbildung 3-10**, durch den Vermerk 1..2 an der Verbindungslinie zwischen *Messauftrag* und *Auftragsmetadaten* dargestellt. 1 bedeutet es ist nur ein Datensatz für die Bestellung enthalten, 2 heißt, es sind jeweils ein Datensatz für die Bestellung und für die Datenlieferung vorhanden.

Die Bestellung und Datenlieferung beinhalten an oberster Stelle die *Auftragsmetadaten*, denen auch die *bauteilspezifischen Zusatzinformationen* zugeordnet sind.

Je nach Anforderung kann die Bestellung beliebig viele Soll-Datensätze bzw. Soll- und Ist-Datensätze enthalten (Abbildung 3-10, Vermerk 1..n an der Verbindungslinie von *Auftragsmetadaten* zu Datensatz (Ist oder Soll)).

Ein Soll- oder Ist-Datensatz setzt sich wiederum zusammen aus den Funktionsmetadaten und den *Nutzdaten*. Die Funktionsmetadaten enthalten *Datensatzmetadaten*, denen die Daten zu *Prüfling*, *Messgerätesetup* und *Messgerätebetriebsgrößen* zugeordnet werden. Die *Messgerätebetriebsgrößen* sind hierbei optional d.h. können auch entfallen. *Prüfling* und *Messgerätesetup* müssen aber in jedem Fall vorhanden sein.

Je nach Anforderung sind den *Datensatzmetadaten* beliebig viele Nutzdaten zugeordnet. Die Nutzdaten setzen sich wiederum zusammen aus dem *Funktionsdatum*, welche je nach Anforderung durch mehrere *abgeleitete Kenngrößen* erweitert werden können.

Fehler! Verweisquelle konnte nicht gefunden werden. beschreibt die wesentlichen Informationen, die mit em FDX-Format ausgetauscht werden.

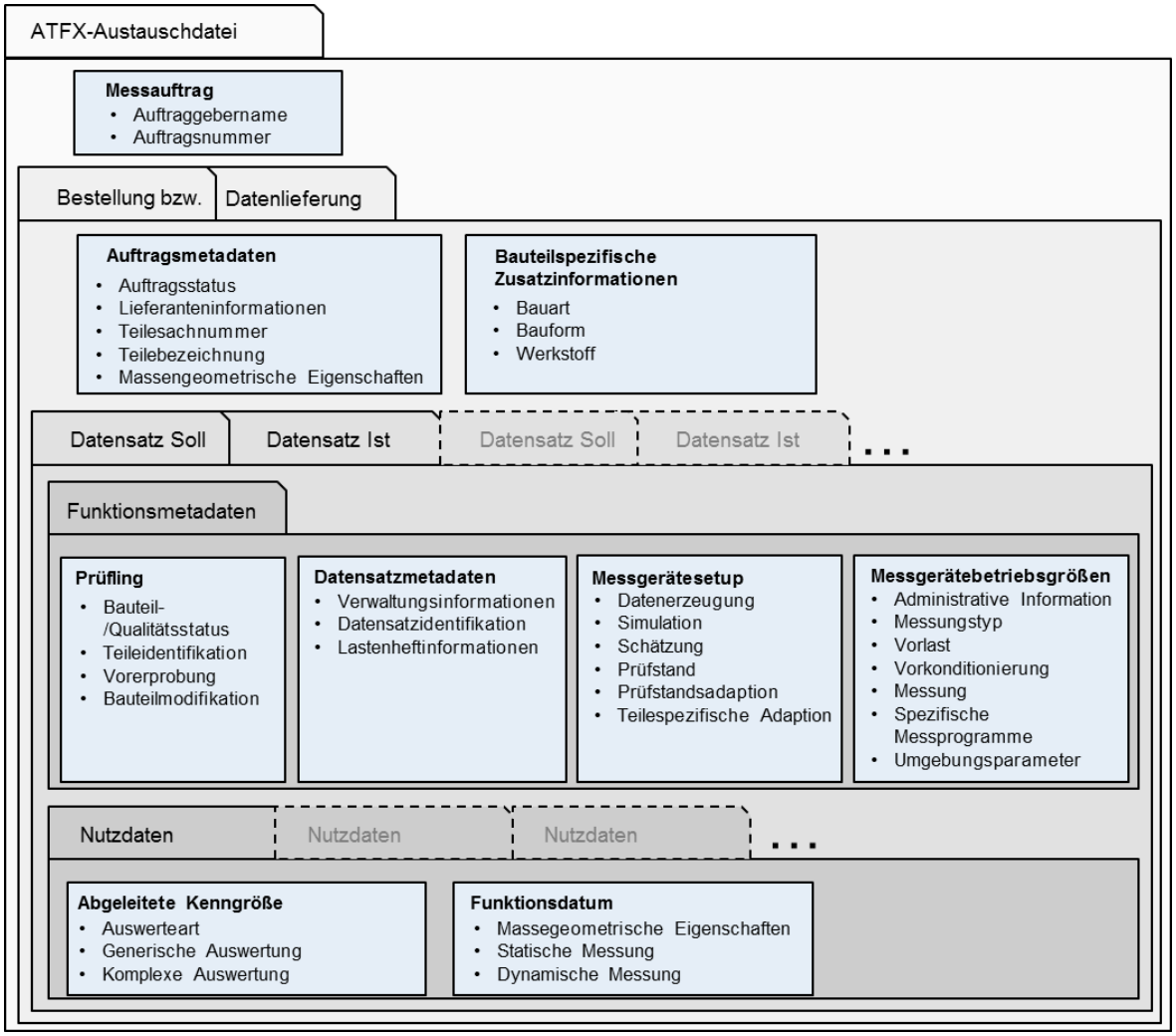
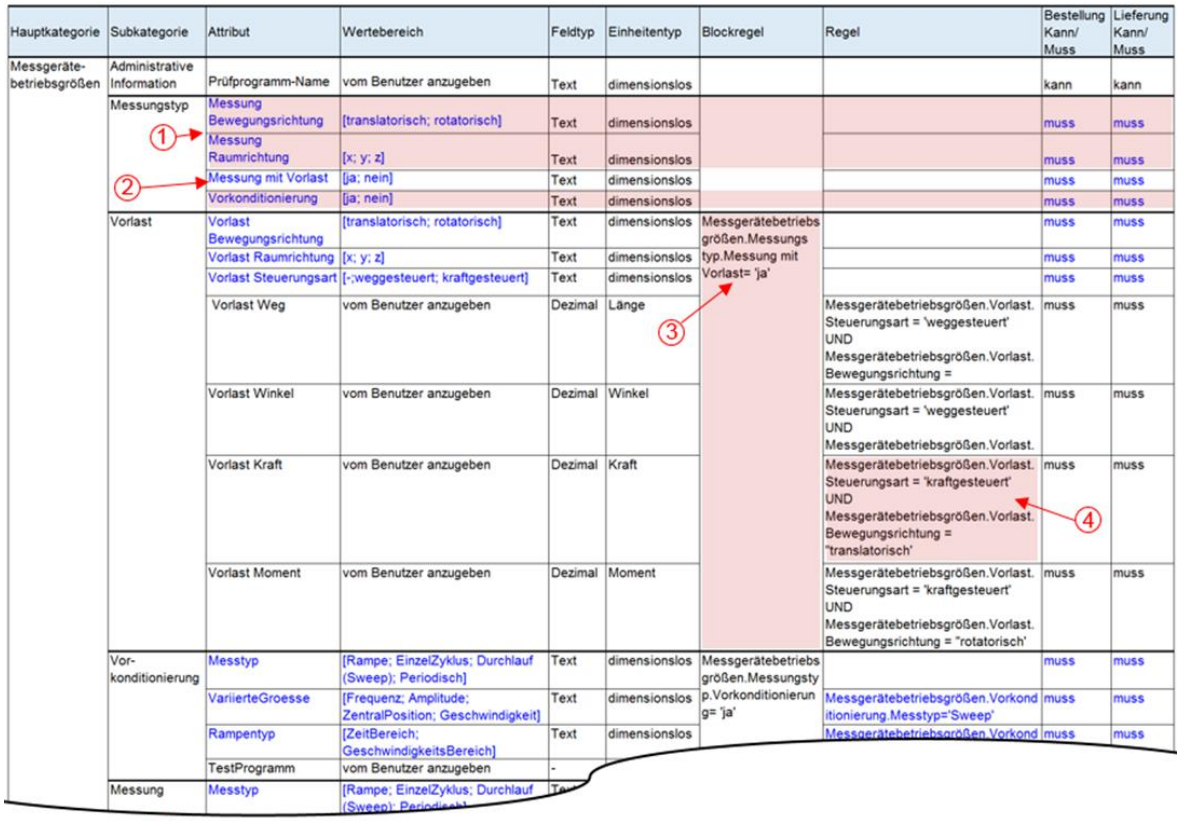


Abbildung 3-11 Funktionsdatenaustauschdatei: Hauptkategorien und Subkategorien bzw. Inhalte

Abbildung 3-12 zeigt die im FDX-Format vorhandene Logik. Der Ansatz ist, Vorlagen zur Verfügung zu stellen, die je nach Anforderung individuell konfiguriert und befüllt werden können.

Unterhalb der Subkategorien befinden sich die einzelnen Attribute. Grundsätzlich sind alle Attribute in Muss- oder Kann-Attribute aufgegliedert, je nachdem, ob sie vom Datenanforderer oder Datenlieferant zwingend befüllt werden müssen oder nicht. Außerdem gibt es steuernde und einfache Attribute.



Hauptkategorie	Subkategorie	Attribut	Wertebereich	Feldtyp	Einheitentyp	Blockregel	Regel	Bestellung Kann/ Muss	Lieferung Kann/ Muss
Messgeräte-betriebsgrößen	Administrative Information	Prüfprogramm-Name	vom Benutzer anzugeben	Text	dimensionslos			kann	kann
	Messungstyp	Messung Bewegungsrichtung	[translatorisch; rotatorisch]	Text	dimensionslos			muss	muss
		Messung Raumrichtung	[x; y; z]	Text	dimensionslos			muss	muss
		Messung mit Vorlast	[ja; nein]	Text	dimensionslos			muss	muss
		Vorkonditionierung	[ja; nein]	Text	dimensionslos			muss	muss
	Vorlast	Vorlast Bewegungsrichtung	[translatorisch; rotatorisch]	Text	dimensionslos	Messgerätebetriebsgrößen.Messungstyp.Messung mit Vorlast= 'ja'		muss	muss
		Vorlast Raumrichtung	[x; y; z]	Text	dimensionslos			muss	muss
		Vorlast Steuerungsart	[-;weggesteuert; kraftgesteuert]	Text	dimensionslos			muss	muss
		Vorlast Weg	vom Benutzer anzugeben	Dezimal	Länge		Messgerätebetriebsgrößen.Vorlast. Steuerungsart = 'weggesteuert' UND Messgerätebetriebsgrößen.Vorlast. Bewegungsrichtung =	muss	muss
		Vorlast Winkel	vom Benutzer anzugeben	Dezimal	Winkel		Messgerätebetriebsgrößen.Vorlast. Steuerungsart = 'weggesteuert' UND Messgerätebetriebsgrößen.Vorlast.	muss	muss
		Vorlast Kraft	vom Benutzer anzugeben	Dezimal	Kraft		Messgerätebetriebsgrößen.Vorlast. Steuerungsart = 'kraftgesteuert' UND Messgerätebetriebsgrößen.Vorlast. Bewegungsrichtung = 'translatorisch'	muss	muss
		Vorlast Moment	vom Benutzer anzugeben	Dezimal	Moment		Messgerätebetriebsgrößen.Vorlast. Steuerungsart = 'kraftgesteuert' UND Messgerätebetriebsgrößen.Vorlast. Bewegungsrichtung = 'rotatorisch'	muss	muss
	Vor-konditionierung	Messtyp	[Rampe; EinzelZyklus; Durchlauf (Sweep); Periodisch]	Text	dimensionslos	Messgerätebetriebsgrößen.Messungstyp.Vorkonditionierung= 'ja'		muss	muss
		VariierteGroesse	[Frequenz; Amplitude; ZentralPosition; Geschwindigkeit]	Text	dimensionslos		Messgerätebetriebsgrößen.Vorkonditionierung.Messtyp='Sweep'	muss	muss
		Rampentyp	[ZeitBereich; GeschwindigkeitsBereich]	Text	dimensionslos		Messgerätebetriebsgrößen.Vorkond	muss	muss
		TestProgramm	vom Benutzer anzugeben	-					
	Messung	Messtyp	[Rampe; EinzelZyklus; Durchlauf (Sweep); Periodisch]	Text					

① Steuernde Attribute „Messung Bewegungsrichtung“ und „Messung Raumrichtung“. Es handelt sich um Attribute, die zwingend befüllt werden müssen. Dabei können nur die vorgegebenen Werte „translatorisch“ oder „rotatorisch“ bzw. „x“ oder „y“ oder „z“ ausgewählt werden. Abhängig von der Werteauswahl werden über Regeln Attribute in den Subkategorien zu den spezifischen Messprogrammen aktiviert oder deaktiviert.

② Steuerndes Attribut „Messung mit Vorlast“: Durch setzen des Wertes „ja“ ist die Blockregel ③ der Subkategorie Vorlast erfüllt, der Block wird aktiviert und die Attribute unter Vorlast müssen befüllt werden.

④ Einfache Regel: Wenn bei den Attributen Vorlast „Steuerungsart“ und Vorlast „Bewegungsrichtung“ mit „kraftgesteuert“ und „translatorisch“ ausgewählt wurde, muss der Anwender einen Wert für die „Vorlast Kraft“ als Zahl mit der Einheit N oder kN angeben.

Abbildung 3-12 Beispiele zu Attributen, Wertebereichen, Blockregeln und einfachen Regeln

Die steuernden Attribute, welche in den Abbildungen blau dargestellt sind, sind wesentlicher Bestandteil des Vorlagen-Konzeptes. Sie besitzen einen vordefinierten Wertevorrat, der im VDA AK FDX abgestimmt wurde. Der Datenanforderer/-lieferant kann hier keine eigenen Werte eingeben, sondern nur aus diesen vordefinierten Werten auswählen.

Weiterhin sind den steuernden Attributen Regeln zugeordnet, welche auf Basis der zuvor ausgewählten Werte den Inhalt und die Struktur der zu liefernden Daten bestimmen. Die Regeln definieren, ob Attribute befüllt werden müssen oder nicht, bzw. unzutreffend sind und deaktiviert werden können. Dabei gibt es einfache Regeln, bei denen nur einzelne Attribute gesteuert werden, und Blockregeln, welche mehrere Attribute oder ganze Subkategorien aktivieren oder deaktivieren.

3.8.2 Messauftrag

Die Hauptkategorie *Messauftrag* enthält die folgenden Attribute, in denen die allgemeinen Informationen zum Auftraggeber, zum Anforderer und zur Identifikation des Auftrages abgelegt werden:

- *Auftraggebernummer (D-U-N-S Nummer)*
- *Auftraggebername*
- *Anfordererabteilung*
- *Anforderername*
- *Anforderertelefonnummer*
- *Auftragsteilung (mehrere Teilaufträge)*

- *Auftragsnummer*
- *Unterauftragsnummer*

Im Beispiel ist der Anforderer die ´MusterOEM AG´ mit der *Auftraggebernummer* ´123456789´, die im neunstelligen D-U-N-S Format anzugeben ist. Die drei folgenden Kann-Attribute werden in diesem Fall freigelassen. Der Auftrag soll nicht geteilt werden und die *Auftragsnummer* ist ´9876-2017´ (***Abbildung 3-13***).

Haupt-kategorie	Sub-kategorie	Attribut	Werte-bereich	Feldtyp	Einheitentyp	Block-regel	Regel	Bestellung Kann/ Muss	Lieferung Kann/ Muss	Ein-heit	Wert
Messauftrag		Auftraggebernummer		Dezimal	dimensionslos			kann	kann	-	123456789
		Auftraggebername		Text	dimensionslos			muss	muss	-	MusterOEM AG
		Anfordererabteilung		Text	dimensionslos			kann	kann	-	
		Anforderername		Text	dimensionslos			kann	kann	-	
		Anforderertelefonnummer		Text	dimensionslos			kann	kann	-	
		Auftragsteilung	[ja;nein]	Text	dimensionslos			kann	kann	-	nein
		Auftragsnummer		Text	dimensionslos			muss	muss	-	9876-2017
		Unterauftragsnummer		Text	dimensionslos		Messauftrag. Auftragsteilung= ja	muss	muss	-	

Legende:
Grün umrandet: Vom VDA-vorgegebene Attribute und Auswahlmöglichkeiten
Rot umrandet: Vom Anforderer auszufüllende Werte und ggf. Einheiten

Abbildung 3-13 Hauptkategorie Messauftrag mit zugehörigen Attributen und beispielhafter Befüllung der Werte

Eine ATFX-Austauschdatei enthält den *Messauftrag* genau einmal. Eine Besonderheit bei der Hauptkategorie *Messauftrag* ist, dass sie aufgrund der geringen Anzahl der zugehörigen Attribute keine Subkategorien besitzt. Die zugehörigen Attribute sind in diesem Fall direkt unter der Hauptkategorie angeordnet.

3.8.3 Auftragsmetadaten und Bauteilspezifische Zusatzinformationen

Dem *Messauftrag* zugeordnet sind die Bestellung und je nach Bestellstatus, d.h. wenn der Auftrag vom Lieferanten bearbeitet und befüllt wurde, die Datenlieferung. Die Informationen zu Lieferant und Bauteil, sowie bauteilspezifische Informationen für Bestellung und Datenlieferung werden unter den Hauptkategorien *Auftragsmetadaten* und *Bauteilspezifische Zusatzinformationen* abgelegt.

Hauptkategorie	Subkategorie	Attribut	Wertebereich	Feldtyp	Einheitentyp	Block regel	Regel	Bestellung Kann/ Muss	Lieferung Kann/ Muss	Ein-heit	Wert
Auftrags-metadaten		Auftragsart	[Bestellung; Lieferung]	Text	dimensionslos			muss	muss	-	Bestellung
	Lieferant	Lieferantennummer		Decimal	dimensionslos			kann	muss	-	987654321
		Lieferantenname		String	dimensionslos			muss	muss	-	MusterTier1 AG
	Konstruktions-stand	Teilebezeichnung (Auftraggeber)		String	dimensionslos			muss	kann	-	Gummilager einfach links
		Sachnummer (Auftraggeber)		Text	dimensionslos			muss	kann	-	A 999 999 99 99 99
		Sachnummernversion		Text	dimensionslos						
	Teile-beschreibung	Komponente		Text	dimensionslos			muss	muss	-	Gummilager
	Masse-geometrische Eigenschaften	Bauteilmasse		Dezimal	Masse			kann	kann	kg	0,75
		Massenermittlungs-methode		Text	dimensionslos			kann	kann	-	

Abbildung 3-14 Auszug aus der Hauptkategorie Auftragsmetadaten mit zugehörigen Attributen und beispielhafter Befüllung der Werte

Im Beispiel werden bei den Auftragsmetadaten die Attribute *Auftragsart*, *Lieferantennummer*, *Lieferantenname*, *Teilebezeichnung*, *Sachnummer*, *Teilebeschreibung* und *Bauteilmasse* auf die in ***Abbildung 3-14*** in der letzten Spalte dargestellten Werte gesetzt.

Bei den *Bauteilspezifischen Zusatzinformationen* soll als *Bauform* ein Gummilager ´ohne Axialanschlag´ definiert werden (***Abbildung 3-15*****Fehler! Verweisquelle konnte nicht gefunden werden.**). Die Hauptrichtung des Bauteils soll in x-Richtung translatorisch sein. Sobald für das Attribut *Hauptrichtung x*

translatorisch der Wert ´ja´ ausgewählt wurde, greift die zugehörige Regel für die Aktivierung des Attributes *Nennsteifigkeit in x-Richtung translatorisch* und es muss hier ein Dezimalwert mit dem Einheitentyp Lokale Steifigkeit eingetragen werden. Gleichzeitig werden die Attribute für die *Hauptrichtungen y und z translatorisch*, und *x, y und z rotatorisch* inaktiv und müssen nicht befüllt werden.

Hauptkategorie	Sub-kategorie	Attribut	Werte-bereich	Feldtyp	Einheitentyp	Block-regel	Regel	Bestellung Kann/ Muss	Lieferung Kann/ Muss	Einheit	Wert
Bauteilspezifische Zusatzinformationen		Bauform		Text	Dimensionslos			muss	kann	-	ohne Axial-anschlag
		Shore-Härte (Hauptkörper)		Dezimal	Härte			kann	muss	-	
		Hauptrichtung x translatorisch	[ja; nein, nicht relevant]	Text	Dimensionslos			kann	kann	-	ja
		Nennsteifigkeit in x-Richtung translatorisch		Dezimal	Lokale Steifigkeit		Bauteilspezifische Zusatzinformationen. Hauptrichtung X Translatorisch = ja	muss	muss	N/mm	8500
		Hauptrichtung y translatorisch	[ja; nein]	Text	Dimensionslos			kann	kann	-	nein
		Nennsteifigkeit in y-Richtung		Dezimal	Lokale		Bauteilspezifische	muss	muss	-	-
		rotatorisch			Dimensionslos			kann	kann	-	nein
		Nennsteifigkeit in z-Richtung rotatorisch		Dezimal	Lokale Steifigkeit		Bauteilspezifische Zusatzinformationen. Hauptrichtung Z Rotatorisch = ja	muss	muss	-	-
		Kennung		Text	Dimensionslos			kann	kann		GF-X1
		Hülsendurch-messer		Dezimal	Länge			kann	muss	-	-
		Kernlänge		Dezimal	Länge			kann	muss	mm	75

Abbildung 3-15 Auszug aus der Hauptkategorie Bauteilspezifische Zusatzinformationen mit zugehörigen Attributen und beispielhafter Befüllung der Werte

3.8.4 Datensatzmetadaten

Da es sich um einen Auftrag handelt, wird unter der Subkategorie *Verwaltungsinformationen* bei dem Attribut *Soll-Ist-Kennung* der Wert ´Soll´ ausgewählt. Zur Subkategorie *Datensatzidentifikation* werden der *Datensatzname*, die *Versionsnummer*, der *Datensatz Zeitstempel* und der *Ersteller* angegeben (***Abbildung 3-16***).

Haupt-kategorie	Subkategorie	Attribut	Werte-bereich	Feldtyp	Einheitentyp	Block-regel	Regel	Bestellung Kann/ Muss	Lieferung Kann/ Muss	Ein-heit	Wert
Datensatz-metadaten	Verwaltungs-informationen	Prüfer Name		Text	Dimensonslos			irrelevant	kann		
		Soll-Ist-Kennung	[Soll; Ist]	Text	Dimensionslos			muss	muss	-	Soll
		Test		Datum	Dimensionslos			irrelevant	muss		
		Messziel	[Standard; Betrieb unter Last; Spezielles Ereignis; Zerstörend]	Text	Dimensionslos			kann	kann		
	Datensatz-identifikation	Datensatz-name		Text	Dimensionslos		'=Auftragsmetadaten .Teilebeschreibung. Komponente. &<Zähler>	muss	muss	-	Gummilager.001
		Versions-nummer		Dezimal	Dimensionslos			muss	muss	-	002
		Datensatz Zeitstempel		Datum	Dimensionslos			muss	muss	-	20170120-085831
		Ersteller		Text	Dimensionslos			muss	muss	-	Max.Muster
		Entwicklungs-auftrags-nummer		Text	Dimensionslos			kann	kann		
		Bemerkung		Text	Dimensionslos			kann	kann		
		Gültigkeits-kennzeichen		Datum	Dimensionslos		Messauftrag. Auftragsteilung=ja	muss	muss		
	Lastenheft	Lastenheft Bezeichnung		Text	Dimensionslos			kann	kann		
		Lastenheft Version		Text	Dimensionslos			kann	kann		
		Lastenheft Konformität	[ja, nein]	Text	Dimensionslos			irrelevant	muss		
		Bemerkung		Text	Dimensionslos			irrelevant	muss		

Abbildung 3-16 Hauptkategorie Datensatzmetadaten mit zugehörigen Attributen und beispielhafter Befüllung der Werte

3.8.5 Prüfling

In der Subkategorie *Prüfling* werden die spezifischen Informationen zum Prüfling angegeben. In diesem Fall handelt es sich um einen Prüfling mit der *Vorerprobung* ´Welldauerlauf mit Ozonkammer´ und mit einer Modifikation, einer Bohrung zur Temperaturmessung im Lagerkern (***Abbildung 3-17***).

Durch die jeweilige Auswahl der Werte ´ja´ bei den steuernden Attributen *Vorerprobung durchgeführt* und *Prüflingsmodifikation*, werden die Subkategorien *Vorerprobung* und *Bauteilmodifikation*, sowie die zugehörigen Attribute aktiviert und die entsprechenden Werte können eingegeben werden.

Haupt-kategorie	Sub-kategorie	Attribut	Werte-bereich	Feldtyp	Einheitentyp	Block-regel	Regel	Bestellung Kann/ Muss	Lieferung Kann/ Muss	Ein-heit	Wert
Prüfling	TeileStatus	Bauteilstatus		Text	Dimensonslos			kann	kann		
		Qualitätsstatus		Text	Dimensionslos			kann	kann		
		Vulkanisations Status		Datum	Dimensionslos			kann	kann		
		Vorerprobung durchgeführt	[ja, nein]	Text	Dimensionslos			muss	muss	-	ja
		Prüflingsmodifikation	[ja, nein]	Text	Dimensionslos			muss	muss	-	ja
	Teile-identifikation	Eindeutige Prüflingsidentifikationsnummer (Lieferant)		Text	Dimensionslos			kann	kann		
		Eindeutige Prüflingsidentifikationsnummer (Auftraggeber)		Text	Dimensionslos			kann	kann		
		Chargennummer (Lieferant)		Text	Dimensionslos			kann	kann		
		Herstellungsdatum		Datum	Dimensionslos			kann	kann		
	Vor-erprobung	Bemerkung zu Vorerprobungen		Text	Dimensionslos	Prüfling.TeileStatus. Vorerprobung durchgeführt='ja'		kann	muss	-	Welldauerlauf mit Ozonkammer
		Dateianhang zu Vorerprobungen		Datei	Dimensionslos			kann	kann	-	20170120-0923--Vorerprobung-Welldauerlauf.pdf
	Bauteil-modifikation	Bemerkung zu Prüflingsmodifikationen		Text	Dimensionslos	Prüfling.TeileStatus. Prüflingsmodifikation='ja'		kann	muss	-	Bohrung für Temperaturmessung in Lagerkern eingebracht
		Dateianhang zu Prüflingsmodifikationen		Datei	Dimensionslos			kann	kann	-	20170120-0941--Prüflingsmodifikation-Bohrung-Lagerkern.pdf

Abbildung 3-17 Hauptkategorie Prüfling mit zugehörigen Attributen und beispielhafter Befüllung der Werte

3.8.6 Messgerätesetup

Im *Messgerätesetup* werden die Angaben zum *Datenursprung* hinterlegt. In der Regel sind dies Messungen, es ist aber auch vorgesehen, durch Simulation bestimmte oder geschätzte Werte auszutauschen (***Abbildung 3-18***).

In unserem Fall handelt es sich um eine Messung und somit wird für das Attribut *Datenerzeugung* der Wert ´Messung´ ausgewählt. Über Blockregeln werden damit die Subkategorien *Prüfstand* und *Prüfstandsadaption* aktiviert und bei den jeweiligen Attributen für die Bezeichnung des *Prüfstands*- und der *Prüfstandsadaption*, sowie die Referenzen für zugehörigen *Dateianhänge* werden die entsprechenden Informationen eingetragen.

Haupt-kategorie	Sub-kategorie	Attribut	Werte-bereich	Feld-typ	Einheitentyp	Block-regel	Regel	Bestellung Kann/ Muss	Lieferung Kann/ Muss	Ein-heit	Wert
Mess-geräte-setup	Daten-ursprung	Daten-erzeugung	[Messung; Simulation; Schätzung]	Text	Dimensionslos			muss	muss	-	Messung
	Simulation	Software-produkt		Text	Dimensionslos	Messgerätesetup. Datenursprung. Datenerzeugung = Simulation		kann	muss		
		Software Version		Text	Dimensionslos			kann	kann		
		Bemerkung		Text	Dimensionslos			kann	kann		
	Schätzung	Benennung		Text	Dimensionslos	Messgerätesetup. Datenursprung. Datenerzeugung = Schätzung		kann	muss		
		Beschreibung		Text	Dimensionslos			kann	kann		
	Prüfstand	Prüfstands-bezeichnung		Text	Dimensionslos	Messgerätesetup. Datenursprung. Datenerzeugung = Messung		kann	muss	-	Servohydraulischer Prüfstand.003
		Bemerkung		Text	Dimensionslos			kann	kann	-	-
		Dateianhang		Datei	Dimensionslos			kann	kann	-	20170120-100421--Servohydraulischer Prüfstand-003.pdf
	Prüf-stands-adaption	Adaptions-bezeichnung		Text	Dimensionslos	Messgerätesetup. Datenursprung. Datenerzeugung = Messung		kann	kann	-	Großer Prüfring mit Kraftmessdose ringseitig
		Sachnummer		Text	Dimensionslos			kann	kann		
		Sachnummer-Version		Text	Dimensionslos			kann	kann		
		Zeichnungs-nummer		Text	Dimensionslos			kann	kann		
		Zeichnungs-stand		Text	Dimensionslos			kann	kann		
		Bemerkung		Text	Dimensionslos			kann	kann		
		Dateianhang		Datei	Dimensionslos			kann	kann	-	20170120-100637--Gr-Pruefring-Kraftmessdose.pdf

Abbildung 3-18 Hauptkategorie Prüfling mit zugehörigen Attributen und beispielhafter Befüllung der Werte

3.8.7 Messgerätebetriebsgrößen

Im Folgenden wird die Definition der *Messgerätebetriebsgrößen* anhand folgender Messung eines Gummilagers beschrieben:

- Es handelt sich um eine statische Messung
- Die Bewegung ist translatorisch in x-Richtung
- Es soll keine Vorlast aufgebacht werden
- Es soll keine weitere Vorkonditionierung erfolgen
- Die maximale Prüfgeschwindigkeit soll 10mm/min sein
- Die Anregung soll sinusförmig sein
- Es soll eine kraftgeregelte Messung von +10 bis -10kN durchgeführt werden

In der Attributeliste wird in die Hauptkategorie *Messgerätebetriebsgrößen* und die Subkategorie *Messungstyp* gesprungen.

Die Subkategorie *Messungstyp* besitzt vier Attribute, welche für die Definition der o.g. Messung relevant sind. Diese vier Attribute sind steuernde Elemente, d.h. wie schon in Abschnitt 3.8.1, ***Abbildung 3-12*** beschrieben, kann der Anwender hier keine eigenen Werte eingeben, sondern kann nur aus vordefinierten Werten auswählen:

- Attribut: *Messung Bewegungsrichtung* – Werte: ´translatorisch´, ´rotatorisch´
- Attribut: *Messung Raumrichtung* – Werte: ´x´ oder ´y´ oder ´z´
- Attribut: *Messung mit Vorlast* – Werte: ´ja´ oder ´nein´
- Attribut: *Vorkonditionierung* – Werte: ´ja´ oder ´nein´

Hauptkategorie	Subkategorie	Attribut	Wertebereich	Feldtyp	Einheitentyp	Blockregel	Regel	Bestellung Kann/Muss	Lieferung Kann/Muss	Einheit	Wert
Messgeräte-betriebsgrößen	Verwaltungs-informationen	Prüfprogramm-Name	vom Benutzer anzugeben	Text	Dimensionslos			kann	kann	-	
	Messungstyp	Messung Bewegungsrichtung	[translatorisch; rotatorisch]	Text	Dimensionslos			muss	muss	-	translatorisch
		Messung Raumrichtung	[x; y; z]	Text	Dimensionslos			muss	muss	-	x
		Messung mit Vorlast	[ja; nein]	Text	Dimensionslos			muss	muss	-	nein
		Vorkonditionierung	[ja; nein]	Text	Dimensionslos			muss	muss	-	nein
	Vorlast	Vorlast Bewegungsrichtung	[translatorisch; rotatorisch]	Text	Dimensionslos	Messgeräte.Betriebs-größen.Messungstyp .Messung mit Vorlast = 'ja'		muss	muss	-	
		Vorlast Raumrichtung	[x; y; z]	Text	Dimensionslos			muss	muss	-	
		Vorlast	[-;weggesteuert;	Text	Dim...						

Abbildung 3-19 Subkategorie Messungstyp mit zugehörigen Attributen und beispielhafter Befüllung der Werte

Gemäß dem o.g. Beispiel sind folgende Werte auszuwählen: Für *Messung Bewegungsrichtung* ´translatorisch´, für *Messung Raumrichtung* ´x´, für *Messung mit Vorlast* und *Vorkonditionierung* jeweils ´nein´.

Da die Attribute *Messung mit Vorlast* und *Vorkonditionierung* mit dem Wert ´nein´ belegt wurden, werden die Subkategorien *Vorlast* und *Vorkonditionierung* über Blockregeln inaktiv und somit übersprungen (wenn das Attribut Messung mit Vorlast mit dem Wert ´nein´ belegt ist, dann ist die Subkategorie Vorlast unzulässig; wenn das Attribut Messung mit dem Wert ´ja´ belegt ist, dann ist die Subkategorie Vorlast obligatorisch).

Die nächste Subkategorie ist die *Messung* mit den folgenden vier Steuerattributen:

- *Messtyp*
- *Variierte Größe*
- *Rampentyp*
- *TestProgramm*

Als erstes ist der Wert für das Attribut *Messtyp* auszuwählen. Wegen der gewünschten sinusförmigen Anregung und weil mehr als ein Zyklus gemessen werden soll, wird der Wert ´periodisch´ ausgewählt. Bei der Auswahl von ´periodisch´ sind die Attribute *Variierte Größe* und *Rampentyp* nicht relevant und bleiben über Regeln automatisch inaktiv, d.h. werden nicht befüllt.

Hauptkategorie	Subkategorie	Attribut	Wertebereich	Feldtyp	Einheitentyp	Blockregel	Regel	Bestellung Kann/Muss	Lieferung Kann/Muss	Einheit	Wert
Messgeräte-betriebsgrößen	Verwaltungs-informationen	Prüfprogramm-Name	vom Benutzer anzugeben	Text	Dimensionslos			kann	kann	-	
	Messungstyp	Messung	[translatorisch;	Text	Dimensionslos			muss	muss	-	translatorisch
	Messung	Messtyp	[Rampe; EinzelZyklus; Durchlauf (Sweep); Periodisch]	Text	Dimensionslos			muss	muss	-	Periodisch
		VariierteGroesse	[Frequenz; Amplitude; ZentralPosition; Geschwindigkeit]	Text	Dimensionslos		Messgeräte-betriebsgrößen.Messung.Messtyp='Sweep'	muss	muss	-	
		Rampentyp	[ZeitBereich; Geschwindigkeits-Bereich]	Text	Dimensionslos		Messgeräte-betriebsgrößen.Messung.Messtyp='Rampe'	muss	muss	-	
		TestProgramm	vom Benutzer anzugeben	-	-			irrelevant	irrelevant	-	
	Messprogram	Symmetrie	[ja; nein]	Text	Dimensionslos						

Abbildung 3-20 Subkategorie Messung mit zugehörigen Attributen und beispielhaften Werten

Da es sich im Beispiel um eine Sinusanregung handelt, werden die weiteren Informationen in den Attributen der Subkategorie *Messprogramm Oszillierende Anregung* erfasst. Da eine max. Prüfgeschwindigkeit angegeben ist, wird dem Attribut *Steuerelement zur Auswahl der Anregungsgeschwindigkeit* der Wert ´Geschwindigkeit´ zugewiesen. Dem Attribut *Prüfgeschwindigkeit translatorisch* wird der Wert ´10´ und die Einheit ´mm/min´ zugewiesen.

Dem Attribut *Signalform* wird der Wert ´Sinus´ zugewiesen. Zum Schluss wird dem Steuerelement *Messprogramm Art der Regelung* der Wert ´Kraft´ sowie den Attributen *Startwert für Kraft* und *Endwert für Kraft* der Wert ´10´ und die Einheit ´kN´ zugewiesen, da es sich um eine kraftgeregelte Messung im Bereich von +/-10 kN handeln soll.

Haupt-kategorie	Subkategorie	Attribut	Wertebereich	Feldtyp	Einheitentyp	Blockregel	Regel	Bestellung Kann/ Muss	Lieferung Kann/ Muss	Ein-heit	Wert
Messgeräte-betriebs-größen	Verwaltungs-informationen	Prüfprogramm-Name	vom Benutzer anzugeben	Text	Dimensionslos			kann	kann	-	
	Messungstyp	Messung	[translatorisch;	Text	Dimensionslos			muss	muss	-	translatorisch
	Messprogramm Oszillierende Anregung	Symmetrie	[yes; no]	Text	Dimensionslos	Messgerätebetriebs-größen.Messung. Messtyp='Periodisch'	Messgerätebetriebs-größen.Messung. Messtyp='EinzelZyklus'	muss	muss		
		Hysterese	[load; load-unload]	Text	Dimensionslos	ODER 'Messgerätebetriebs-größen.Messung.	Messgerätebetriebs-größen.Messung. Messtyp='EinzelZyklus'	muss	muss		
		Steuerelement zur Auswahl der Anregung	[Geschwindigkeit, Frequenz, Rotations-geschwindigkeit]	Text	Dimensionslos	Messtyp='EinzelZyklus'	Messgerätebetriebs-größen.Messung. Messtyp='Periodisch'	muss	muss	-	Geschwindig-keit
		Prüfgeschwindigkeit translatorisch		Dezimal	Geschwindig-keit		(Messgerätebetriebs-größen.Messung.Messtyp=' EinzelZyklus' ODER Messgerätebetriebs-größen.Messprogramm OszillierendeAnregung. Steuerelement Auswahl Anregung='Geschwindigkeit') UND 'Messgerätebetriebs-größen.Messung.Messung Bewegungsrichtung= 'translatorisch'	muss	muss	mm/ min	10
		Prüfgeschwindigkeit rotatorisch		Dezimal	Rotations-geschwindig-keit		(Messgerätebetriebs-größen.Messung. Messtyp	muss	muss		
		Signalform	[Sinus; Dreieck; Rechteck]	Text	Dimensionslos			kann	muss	-	Sinus
		Zyklenanzahl		Integer	Dimensionslos			kann	muss		
		Messprogramm Art der Regelung	[Kraft; Weg]	Text	Dimensionslos			muss	muss	-	Kraft
		Startwert für Weg		Dezimal	Länge		Messgerätebetriebs-größen.Messprogramm	muss	muss		
							größen.Messprogramm				
		Startwert für Kraft		Dezimal	Kraft		Messgerätebetriebs-größen.Messprogramm OszillierendeAnregung.Mess programm Art der Regelung='Krafft' UND Messgerätebetriebs-größen.Messung.Messung Bewegungsrichtung= 'translatorisch'	muss	muss	kN	10
		Endwert für Kraft		Dezimal	Kraft		Messgerätebetriebs-größen.Messprogramm OszillierendeAnregung.Mess programm Art der Regelung='Krafft' UND Messgerätebetriebs-größen.Messung.Messung	muss	muss	kN	10

Abbildung 3-21 Subkategorie Messungstyp mit zugehörigen Attributen und beispielhafter Befüllung der WerteAbbildung 3-15

Damit sind alle vorhandenen Informationen für die o.g. Messung in den entsprechenden Attributen vordefiniert.

3.8.8 Funktionsdaten

Die Attribute in der Hauptkategorie *Funktionsdaten* enthalten letztendlich die konkreten Werte, die zu messen sind bzw. gemessen wurden. Untergliedert ist dieser Bereich in die Subkategorien *Massengeometrische Größen*, *Statische Messung* und *Dynamische Messung*. In unserem Fall handelt es sich um eine statische Messung mit Messwerten zu Kraft und Weg. Diese werden in den Attributen *=>u* für den Weg und *=>F* für die Kraft abgelegt. Da es sich um eine größere Reihe von Messwerten handelt sind hier als Feldtypen Vektoren vorgesehen.

Haupt-kategorie	Sub-kategorie	Attribut	Feldtyp	Einheitentyp	Bestellung Kann/ Muss	Lieferung Kann/ Muss	Ein-heit	Wert
Funktions-daten	Massen-geome-trische Größen	Masse	Dezimal	Masse	irrelevant	kann	-	
		CG.x	Dezimal	Länge	irrelevant	kann	-	
		CG.y	Dezimal	Länge	irrelevant	kann		
		CADBodyID	Text	Dimensionslos	irrelevant	kann		
	Statische Messung	Statische Kraft-Weg Kennlinie	Komplex	-	kann	muss		
		=>t	Vektor mit Dezimalwerten	Zeit	kann	kann		
		=>v	Vektor mit Dezimalwerten	Geschwindig-keit	kann	kann		
		=>u	Vektor mit Dezimalwerten	Länge	kann	muss	mm	0;0,119;0,343;0,644;1,001;1,393;1,806;2,212;2,611;2,947;3,213;3,409;3,535;3,619;3,661;3,696;3,717;3,731;3,731;3,71;3,675;3,619;3,549;3,479;3,388;3,29;3,164;2,989;2,814;2,59;2,345;2,058;1,729;1,351;0,931;0,469;-0,014;-0,518;-0,973;-1,393;-1,897;-2,296;-2,618;-2,856;-3,017;-3,094;-3,143;-3,171;-3,199;-3,22;-3,234;-3,227;-3,213;-3,185;-3,143;-3,087;-3,031;-2,954;-2,863;-2,751;-2,625;-2,436;-2,24;-2,009;-1,743;-1,428;-1,071;-0,602;0
		=>F	Vektor mit Dezimalwerten	Kraft	kann	muss	N	0;1,44;3,68;6,08;8,96;12;15,28;19,28;23,92;28,64;32,72;35,92;38,16;39,52;40,32;40,8;41,12;41,2;40,96;40,16;38,88;37,12;35,12;32,96;30,72;28,24;25,6;22,48;19,68;16,72;14;11,28;8,56;5,76;2,8;-0,48;-4,16;-8;-11,76;-15,12;-20,64;-25,52;-30,4;-34,64;-37,6;-39,2;-40;-40,64;-41,2;-41,6;-41,68;-41,2;-40,32;-39,12;-37,52;-35,68;-33,52;-31,2;-28,72;-26,08;-23,36;-20;-17,28;-14,56;-11,84;-9,12;-6,4;-3,12;0
		=> Hystereseschleifen Anzahl	Vector mit Integerwerten	Dimensionslos	kann	kann		
		=>Crot	Vektor mit Dezimalwerten	Lokale Rotatorische	kann	muss		

Abbildung 3-22 Hauptkategorie Funktionsdaten mit Subkategorien, zugehörigen Attributen und beispielhaften Werten

3.8.9 Abgeleitete Kenngrößen

Die *abgeleiteten Kenngrößen* sind Attribute bzw. Werte, die nicht direkt gemessen werden, sondern aus den Messdaten über Berechnungen ermittelt werden. Im Beispiel wurde die translatorische Steifigkeit des Gummilagers in x-Richtung in einem Bereich von -/+0,5mm aus den Messwerten ermittelt und hat den Wert ´8563 N/mm´.

Hauptkategorie	Subkategorie	Attribut	Wertebereich	Feldtyp	Einheitentyp	Bestellung Kann/Muss	Lieferung Kann/Muss	Einheit	Wert
Abgeleitete Kenngrößen	Analyse Typ	Auswerterichtung	[-; x; y; xy]	Text	Diimensionslos	kann	kann	-	x
		AuswertebereichXMin		Dezimal	<mult.>	kann	kann	mm	-0,5
		AuswertebereichXMax		Dezimal	<mult.>	kann	kann	mm	0,5
		AuswertebereichYMin		Dezimal	<mult.>	kann	kann	-	
		AuswertebereichYMax		Dezimal	<mult.>	kann	kann	-	
	Generisch	Min		Dezimal	<mult.>	kann	kann		
		Max		Dezimal	<mult.>	kann	kann		
		Betrag		Dezimal	<mult.>	kann	kann		
		Mittelwert		Dezimal	<mult.>	kann	kann		
		Median		Dezimal	<mult.>	kann	kann		
		Startwert		Dezimal	<mult.>	kann	kann		
		Endwert		Dezimal	<mult.>	kann	kann		
		SteigungMittel		Dezimal	<mult.>	kann	kann		
		SteigungRegression		Dezimal	<mult.>	kann	kann		
		StdAbweichung		Dezimal	<mult.>	kann	kann		
	Komplex	Gaskraft		Dezimal	Kraft	kann	muss		
		Reibkraft		Dezimal	Kraft	kann	muss		
		Daempfungsmass		Dezimal	Dimensionslos	kann	kann		
		Steifigkeit		Dezimal	LokaleSteifigkeit	kann	kann	N/mm	8563
		Rotatorische Steifigkeit		Dezimal	LokaleRotatorischeSteifigk	kann	kann		
		Verlustwinkel		Dezimal	Ebener Winkel	kann	kann		

Abbildung 3-23 Hauptkategorie Abgeleitete Kenngrößen mit Subkategorien, zugehörigen Attributen und beispielhaften Werten

4 Technische Grundlagen

Die vorliegende Spezifikation bietet die Möglichkeit, beschreibende Attribute, skalare Kennwerte und Kennlinien zu definieren. Dadurch ist die Spezifikation für verschiedene Bauteile universell einsetzbar (Gummilager, Stoßdämpfer, Reifendaten, …) und prinzipiell datenbanktauglich.

4.1 Grundverständnis

Die Zusammenhänge, um von fachlichen Anforderungen zu einer konkreten Beschreibung im ATFX-Austauschformat zu kommen sind in folgender ***Abbildung 4-1*** dargestellt..

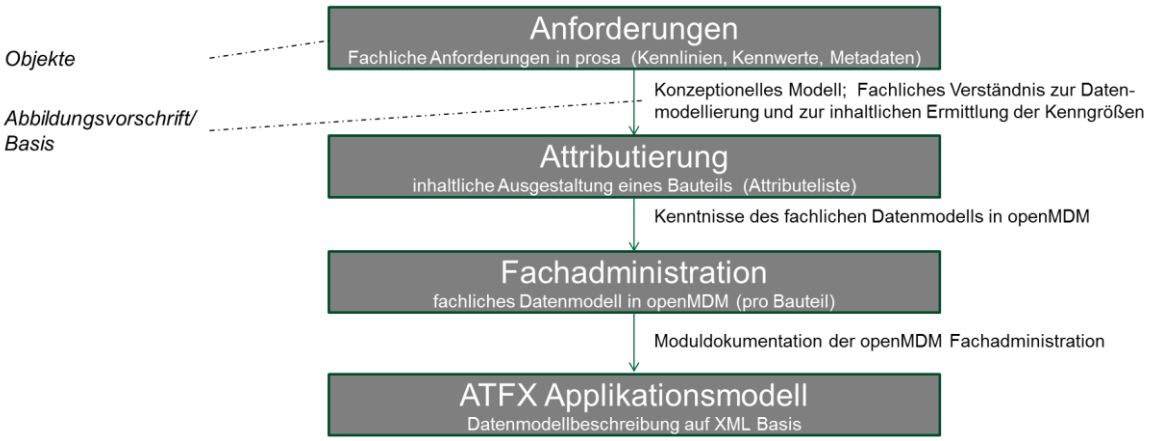


Abbildung 4-1 Zusammenhänge Objekte und Abbildungsvorschriften

Auf Basis des fachlichen Konzeptes und Datenmodells, wurde die technische Basis für das FDX-Datenaustauschformat ausgewählt. Das FDX-Datenformat basiert auf einer Kombination von ASAM ODS Standard, openMDM und openMDM-Erweiterungen durch den VDA Arbeitskreis FDX (*Abbildung 4-2*).

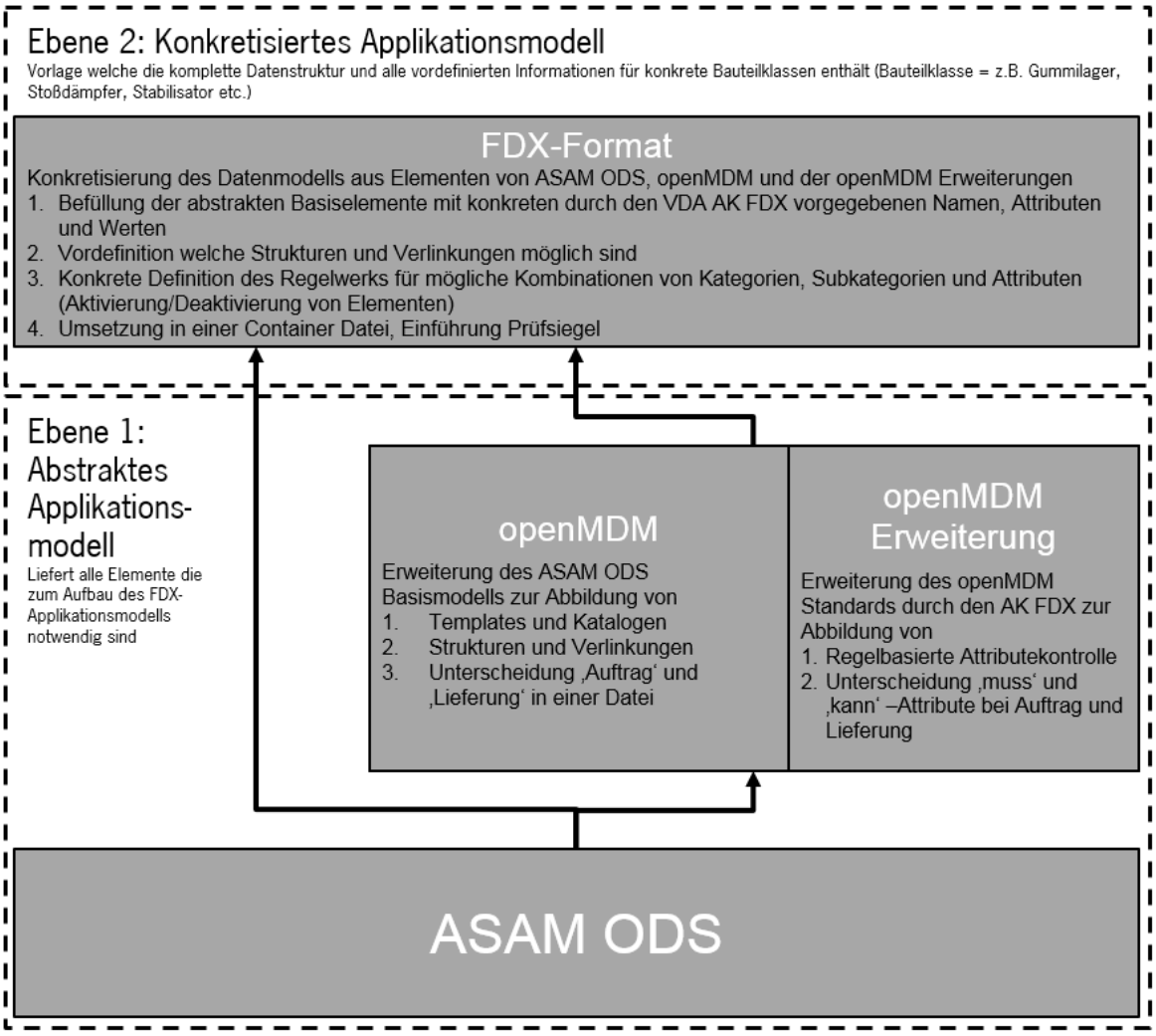


Abbildung 4-2 Zusammenhänge von ASAM ODS, openMDM und FDX. (Ebene 1 und 2)

Der konzeptionelle Aufbau der Spezifikation ist in drei Ebenen unterteilt. In Abbildung 4-2 sind hiervon Ebene 1 und 2 dargestellt.

1. Abstraktes Applikationsmodell: Eine Menge abstrakter Modellelemente (ASAM ODS Basiselemente und openMDM Applikationselemente), die zur Spezifikation von konkreten Applikationsmodellen (Application Models) genutzt werden können. Dieses abstrakte Applikationsmodell ist in ASAM ODS, openMDM und speziellen openMDM-Erweiterungen, welche durch den VDA Arbeitskreis FDX eingebracht wurden, spezifiziert.
2. Konkretisiertes Applikationsmodell im FDX-Format: Zusammenstellung von Modellelementen (Application Elements), die für eine bestimmte Bauteilklasse relevant und anwendbar sind. Dabei werden die abstrakten Modellelemente zugrunde gelegt und mit konkreten vom VDA-AK FDX definierten Informationen und Beziehungen versehen. Die Applikationsmodelle werden von spezialisierten Projektgruppen des Arbeitskreises für unterschiedliche Bauteilklassen (Gummilager, Stoßdämpfer, Stabilisator …) entwickelt und als Vorlage (FA-VDA) veröffentlicht.
3. Die konkreten Instanzen, die das Applikationsmodell auf ein konkretes Bauteil anwenden. Sie enthalten je nach Auftragsstatus die Auftrags- und Prüfdaten in der festgelegten Datenstruktur. Mit anderen Worten enthalten sie die Daten, welche als Datenanforderung und -lieferung in der ATFX-Datei enthalten sind.

4.2 ASAM ODS

ASAM ODS (Association for Standardisation of Automation and Measuring Systems Open Data Services) ist ein Standard zur Datenablage von Messdaten im Umfeld der Automobilindustrie und umfasst sowohl Messdaten, als auch Metadaten (Prüfstandsaufbau, Parameter) und Dateianhänge. ASAM ODS deckt folgende Aspekte ab:

- Basismodell und seine Basiselemente
- Den Rahmen zur Definition von Applikationsmodellen
- Dateiformate (insbesondere ATFX) für den Austausch zwischen Systemen
- Programmierschnittstellen für den Zugriff auf die Daten
- Physische Speicherung mit Langzeitarchivierung

Die ersten drei Aspekte kommen in dieser Empfehlung zur Anwendung. Die Festlegung auf den ASAM ODS Standard bringt folgende Vorteile mit sich:

- Etabliertes Format für Messdaten und verwandte Datentypen
- Zahlreiche existierende Programmsysteme können das Format lesen und schreiben

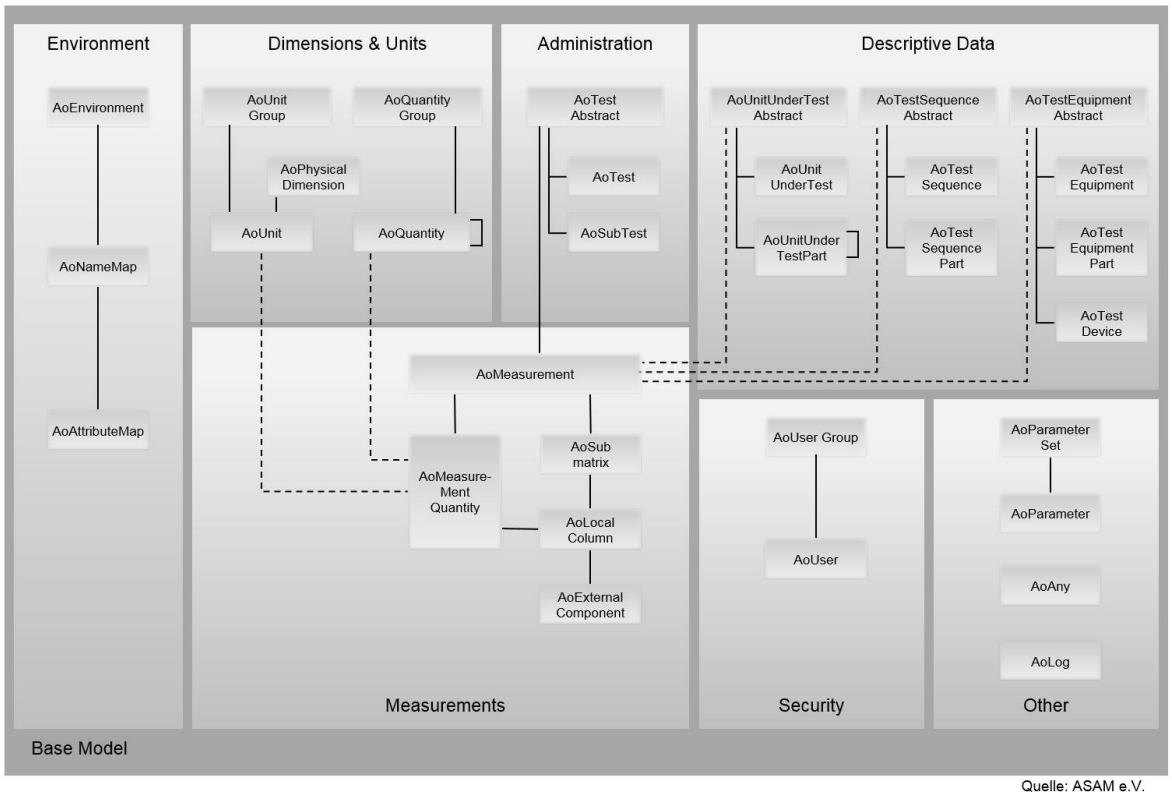
4.2.1 ASAM ODS Basismodell

Das Datenmodell von ASAM ODS unterscheidet zwischen einem Basismodell und Applikationsmodellen. Beides beschreibt anhand von vordefinierten Basiselementen und deren Beziehungen ausschließlich die Struktur der zu speichernden Daten. Um konkrete Werte zu speichern werden letztlich Instanzen dieser Applikationselemente erzeugt und die Werte darin abgelegt.

Das Basismodell ist ein allgemeines Datenmodell zur Beschreibung von Messungen und Funktionsdaten (*Abbildung 4-3*). Im Basismodell sind Basiselemente wie Messungen und Messdaten, Prüfstandsaufbau und deren Beziehungen (Basisbeziehungen) definiert und so beschrieben, dass sie elektronisch gespeichert werden können.

Jedes Basiselement repräsentiert einen Typ von Information. Beispielsweise ist AoUnit das Basiselement, das eine physikalische Einheit wie Newton oder Millimeter repräsentiert, AoMeasurementQuantity ist das Basiselement, das eine gemessene physikalische Größe wie Kraft oder Länge repräsentiert.

Jede Basisbeziehung repräsentiert eine Verbindung mit einer bestimmten Bedeutung zwischen zwei Basiselementen. Beispielsweise verweist AoMeasurementQuantity auf AoUnit, was angibt, welche aller möglichen Einheiten die aktuelle Einheit dieser Größe ist. Das Basismodell ist die Grundlage für alle Applikationsmodelle, die ASAM ODS verwenden.



Quelle: ASAM e.V.

Abbildung 4-3 ASAM ODS Basismodell

4.1.1 Applikationsmodelle

Applikationsmodelle decken spezifische Anwendungsfälle ab, z.B. eine Stoßdämpfer- oder Gummilager-Messung. Die allgemeinen Elemente des Basismodells werden im Applikationsmodell konkretisiert. Hierin wird definiert, welche Basiselemente verwendet werden und wie sie benannt werden. Jedes Element eines Applikationsmodells bezieht sich auf ein Element des Basismodells, und übernimmt Inhalte und Bedeutung des Basiselements (***Abbildung 4-4***).

Das Applikationsmodell muss die Basiselemente verwenden, um die Informationen und deren Struktur zu modellieren, welche dann in der reellen Anwendung über ASAM ODS gehandhabt oder gespeichert werden sollen. Ein Applikationsmodell aufzubauen bedeutet, eine Zusammenstellung von Applikationselementen zu spezifizieren, so dass jede Informationseinheit, welche gespeichert werden soll, in ein einzelnes Applikationselement platziert werden kann. Die Applikationselemente erben alle Pflichtattribute und alle Basisbeziehungen aus ihren korrespondierenden Basiselementen.

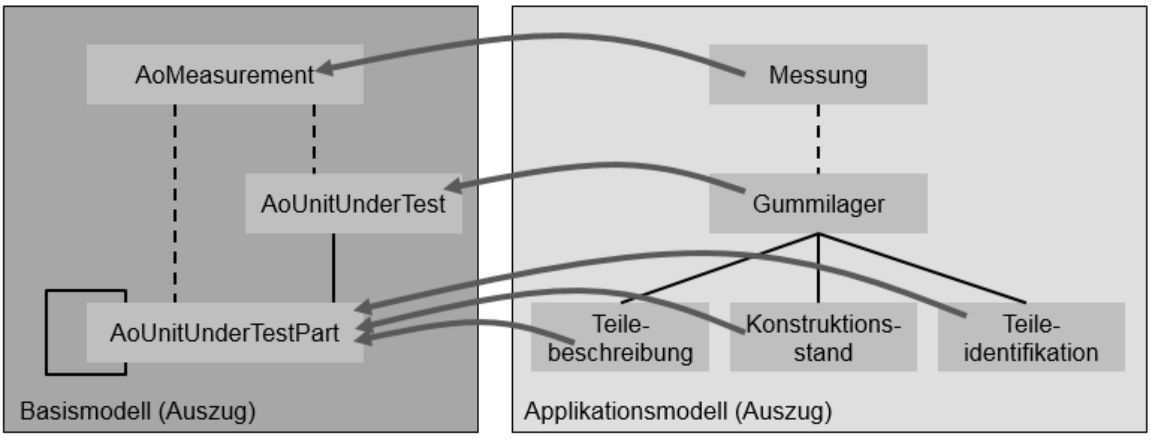


Abbildung 4-4 Auszug aus einem Applikationsmodell und dem zugehörigen Teil des Basismodells

Der Auszug aus dem Basismodell zeigt die Basiselemente `AoMeasurement`, `AoUnitUnderTest`, und `AoUnitUnderTestPart`. Die Linien zwischen den Basiselementen sind die Basisbeziehungen. Die durchgezogenen Linien sollen hier eine direkte Beziehung von über und untergeordneten Elementen symbolisieren und die gestrichelten Linien einen Verweis zwischen Basiselementen unterschiedlicher Kategorien (z.B. Measurements und Descriptive Data siehe ***Abbildung 4-3***)

Die Multiplizität der Assoziationen (welche im Bild nicht dargestellt ist) ist in diesem Fall:

- Immer 0..* für alle Beziehungen zu/von `AoMeasurement`.
- Eine `AoUnitUnderTest` genau wie `AoUnitUnderTestPart` können 0..* `AoUnitUnderTestPart` besitzen.
- Eine `AoUnitUnderTestPart` gehört zu einer `AoUnitUnderTest` oder zu einer `AoUnitUnderTestPart` im Sinne eines Komponentenbaums.

Der dargestellte Auszug aus dem Applikationsmodell ist ein möglicher Aufbau, bestimmte Informationen eines Gummilagers für die zugehörigen Messungen abzulegen. Der Gestalter dieses Applikationsmodells hat folgende Entscheidungen getroffen:

- Er nennt jede Messung ‚Messung' und hat deshalb ein Applikationselement ‚Messung' in sein Applikationsmodell aufgenommen. Dieses hat den Basistyp `AoMeasurement`.
- Er hat entschieden, dass der Fokus die Messung eines Gummilagers ist. Deshalb ist das Applikationselement ‚Gummilager' das Wurzelelement seiner Test-Objekt-Beschreibung. Dieses hat den Basistyp `AoUnitUnderTest`.
- Er hat entschieden, dass für die eindeutige Beschreibung des Gummilagers weitere Informationen wie Teilebeschreibung, Konstruktionsstand und Teileidentifikation notwendig sind. Deshalb hat er drei Applikationselemente des Typs `AoUnitUnderTestPart` in sein Applikationsmodell aufgenommen. Alle stehen in Beziehung zu ‚Gummilager'.
- Er hat entschieden, dass eine Messungs-Aktion genau dem Gummilager zugeordnet werden soll, von dem die Messdaten aufgenommen wurden. Die Messung sollte wissen, welches Gummilager auf dem Prüfstand war und das Gummilager sollte wissen welche Messungen von ihm gemacht wurden. Dies ist in ***Abbildung 4-4*** abgebildet über die Beziehung von `AoMeasurement` und `AoUnitUnderTestAbstract`, welche sich vererbt auf `AoUnitUnderTest` und `AoUnitUnderTestPart` und auf die daraus abgeleiteten Applikationselemente ‚Gummilager', ‚Teilebeschreibung', ‚Konstruktionsstand' und ‚Teileidentifikation'. (In ***Abbildung 4-4*** ist dies dargestellt durch die gestrichelten Linien zwischen `AoMeasurement`, `AoUnitUnderTest` und `AoUnitUnderTestPart`.)

Es gibt mehrere Möglichkeiten, Applikationsmodelle für den gleichen Zweck aufzustellen. Im Rahmen der Standardisierung liefert diese VDA-Richtlinie vordefinierte Applikationsmodelle für bestimmte Bauteile wie zum Beispiel Gummilager und Stoßdämpfer. Hierzu wird jedoch nicht der reine ASAM ODS Standard verwendet, sondern das openMDM Framework (siehe Abschnitt 4.2). Dieses bietet als Ergänzung zum ASAM ODS erweiterte Funktionen und stellt somit ein Metamodell dar, mit dem ebenfalls Applikationsmodelle spezifiziert werden können.

4.1.2 ASAM ODS Datenaustauschformat ATFX

ATFX (kurz für ASAM Transport Format / XML) ist das zum ASAM ODS gehörende, auf XML basierende Datenaustauschformat. Neben den eigentlichen Nutzdaten (Instance Data) enthalten ATFX-Dateien das Applikationsmodell, d.h. eine Beschreibung der Daten, Typen und Beziehungen, so dass die Daten formal und z.T. inhaltlich geprüft werden können. Zur Auswertung einer ATFX-Datei sind nur die Datei

selbst und das ASAM ODS Basismodell nötig. Das macht es offen für zukünftige Erweiterungen, weil Änderungen im Applikationsmodell in der aktuellen Datei gespeichert sind.

4.1.3 Weiterführende Informationen zu ASAM ODS und ATFX

Für weiterführende Informationen, insbesondere auch um eigene Applikationen zu entwickeln, wird hier auf die ASAM Internetseite und dem dortigen Link zur Beschreibung des Basismodells verwiesen (siehe auch Kap. 7 Normative Referenzen).

4.2 openMDM

Die openMDM® Eclipse Working Group entwickelt offene Konzepte und Softwarekomponenten für das Messdatenmanagement, basierend auf dem ASAM ODS Standard.

4.2.1 Basis Standard openMDM4

Für das FDX-Format setzt diese Empfehlung auf Version 4 des openMDM Frameworks auf. Dazu wurde das ASAM ODS Basismodel erweitert, um z.B. Vorlagen und Kataloge zu ermöglichen. Die Vorlagen und Kataloge können gruppiert und verschachtelt werden. Somit können komplexe Zusammenhänge über eine Art Baukastensystem, welches mit Beziehungen (Hierarchien und Referenzen) versehen wird, abgebildet werden. Eine Übersicht über das openMDM Applikationsmodell befindet sich im Anhang B.

4.2.2 Weiterführende Informationen zum Basisstandard openMDM

Für weiterführende Informationen, insbesondere auch um eigene Applikationen zu entwickeln, wird hier auf den Anhang B und die derzeitige openMDM Internetseite verwiesen (siehe auch Kap. 7 Normative Referenzen).

4.3 VDA FDX Erweiterungen zur Basis openMDM4 und ASAM ODS

Das FDX-Format verwendet eigene spezifische Applikationsmodelle auf Basis der Möglichkeiten des ASAM ODS Basismodells und des openMDM Applikationsmodells. Es konkretisiert die Standardformate durch eine strukturierte Vordefinition von relevanten Attributen und Strukturen für die zweckbezogenen Umfänge.

4.3.1 Erweiterungen zum openMDM Applikationsmodell

Um zwei wesentlichen Anforderungen aus der Zielsetzung für das FDX-Format gerecht werden zu können, müssen am aktuellen openMDM4 Applikationsmodell Erweiterungen vorgenommen werden.

1. Zum einen ist dies die Anforderung, die Auftragsvorlagen mit den in Abschnitt 3.8.1 genannten steuernden Attributen und Regeln für die Aktivierung und Deaktivierung von Komponenten („Blockregeln“) und einzelnen Attributen („einfachen Regeln“) bis hin zu kompletten Subkategorien zu berücksichtigen.
2. Zum anderen ist dies die Anforderung, bei Auftrag und Lieferung Muss-Attribute und Kann-Attribute unterscheiden zu können.

Mit „Komponenten“ sind Applikationselemente des openMDM Applikationsmodells gemeint (siehe Anhang D). Die „Auftragsvorlagen“ dürfen nicht mit den „Templates“ des openMDM Applikationsmodells verwechselt werden, die als Templates für die genannten „Komponenten“ dienen.

Diese Erweiterungen zum openMDM Applikationsmodell werden im Anhang C beschrieben.

Die Konventionen zur Interpretation der UML-Diagramme und der zugehörigen textuellen Beschreibung werden in Anhang I beschrieben.

4.3.2 Zusammenhang ASAM ODS Basismodell, openMDM Applikationsmodell und VDA FDX Applikationsmodell am Beispiel Gummilager

Die fachlichen Mechanismen, mit denen openMDM Applikationsmodelle auf Basis des ASAM ODS Basismodells modellgetriebenen ausgeprägt werden, werden in Anhang D, Anhang E und Anhang F beschrieben.

Wie die Daten technisch im ASAM ATFX-Datenaustauschformat festgehalten werden wird in Anhang G und Anhang H beschrieben.

Die Konventionen zur Interpretation der UML-Diagramme und der zugehörigen textuellen Beschreibung werden in Anhang I beschrieben.

5 Konzeptioneller Aufbau der FDX-Funktionsdatenaustauschdatei

Für den Austausch der Daten dient die Funktionsdatenaustauschdatei (FDAD), die die ATFX-Datei zusammen mit optionalen Referenzdateien und einem ebenfalls optionalen Prüfsiegel in einem Container zusammenfasst (*Abbildung 5-1*).

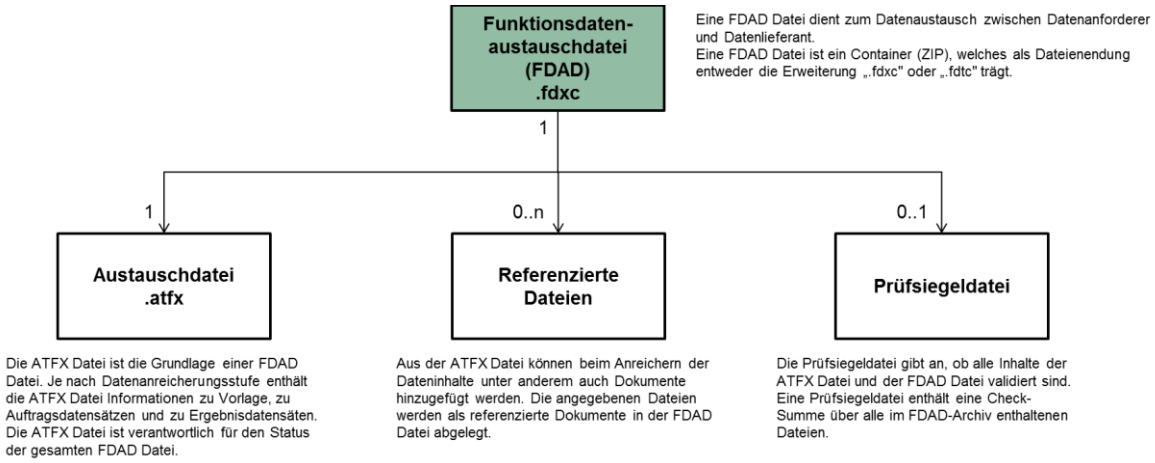


Abbildung 5-1 Konzept: Komponenten im FDX-Austauschformat

Eine FDAD Datei ist ein Container (ZIP), welches als Dateienendung entweder die Erweiterung ".fdxc" oder ".fdtc" trägt. Dabei ist ".fdxc" die Erweiterung für "normale" Dateien und ".fdtc" die Erweiterung für Vorlagen/Templates.

Je nach Anreicherungsstufe und FDAD-Status kann die FDAD-Datei verschiedene weitere Dateien enthalten.

- ATFX-Datei mit Auftragsvorlage, Auftrags- und Ergebnisdatensätzen
- Referenzierte Dateien zu Auftragsvorlage, Auftrags- und Ergebnisdatensätzen
- Prüfsiegel

Anmerkungen:

- Eine FDAD Datei enthält immer eine ATFX-Datei
- Eine FDAD Datei hat für jede Datenanreicherungsstufe einen bestimmten Status
- Die Existenz von referenzierten Dateien und der Prüfsiegeldatei sind abhängig von der Datenanreicherungsstufe und der Verwendung des zugrundeliegenden Datenmodells

5.1 Funktionsaustauschdatei FDAD als Vorlage (.fdtc)

Eine FDAD Datei in der Datenanreicherungsstufe „Vorlage“ wird vom Fachadministrator zur Erstellung von Aufträgen bereitgestellt. In dieser Datenanreicherungsstufe können bereits Dokumente in der FDAD Datei vorhanden sein, falls der Fachadministrator Dokumente als Default-Werte für Attribute hinterlegt hat. Die FDAD Datei muss initial kein Prüfsiegel enthalten, solange der Fachadministrator noch Veränderungen unternimmt und diese nur zwischenspeichert.

5.2 Funktionsdatenaustauschdatei FDAD als Datei (.fdxc)

Für den Funktionsdatenaustausch zwischen Anforderer und Lieferant dient die Funktionsdatenaustauschdatei (FDAD). Sie ist Ergebnis der im der im Kap. 3 beschriebenen Applikationsfälle „Datenanforderer: Auftrag erstellen, abschließen und übermitteln“, „Datenlieferant: Auftrag empfangen, prüfen und annehmen“ und „Datenlieferant: Daten eingeben, Lieferung abschließen und übermitteln“.

5.3 Austauschdatei ATFX-Datei (.atfx)

Die ATFX-Datei des FDX-Formats basiert auf ASAM-ATFX, einem XML-basierenden Industriestandard mit einem Datenmodell für die Speicherung von Messergebnissen. Es handelt sich um ein selbstbeschreibendes Format, das Anforderung, Messumstände und Ergebnisse enthält und ohne weitere Beschreibung ausgewertet werden kann. Alles, was zum Verständnis der Datei notwendig ist, ist in der ATFX-Datei enthalten.

Mit dem Applikationsmodell und seinen Applikationselementen, Attributen und Werten werden die Regeln für den gesamten Messauftrag für ein spezifisches Bauteil mitgeliefert. Somit liefert es eine vollständige Beschreibung für die Ermittlung der Funktionsdaten, deren Speicherung und Übermittlung auf XML-Basis.

Weiterhin handelt es sich um ein flexibles Format, das für unterschiedlichste Bauteile und Mess-/ Simulationsmethoden angepasst werden kann.

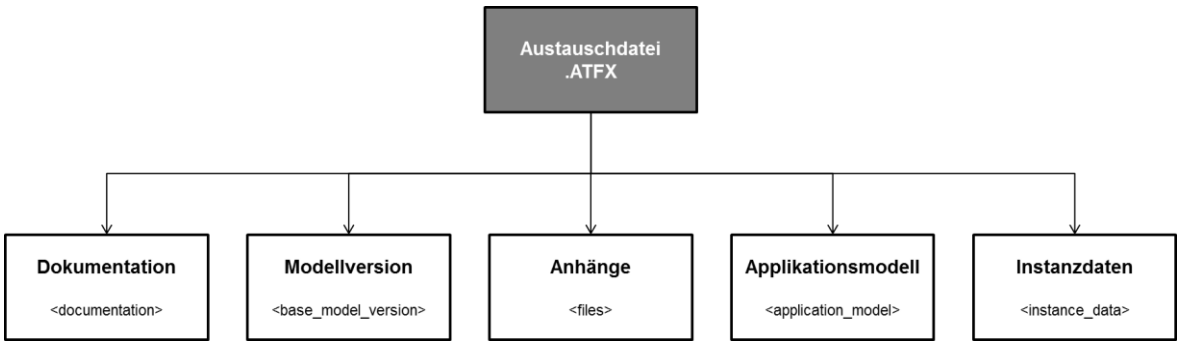


Abbildung 5-2 Konzept: Struktur ATFX-Datei

Die ATFX-Datei ist die zentrale Datei, in der die Kategorien, Attribute und deren Eigenschaften festgelegt und abgespeichert werden (***Abbildung 5-2***). ATFX-Dateien werden im Rahmen der im Kap. 3 dargestellten Anwendungsfälle FA-Master, FA-VDA und FA-OEM erstellt und mit entsprechenden Informationen angereichert (***Abbildung 5-3***).

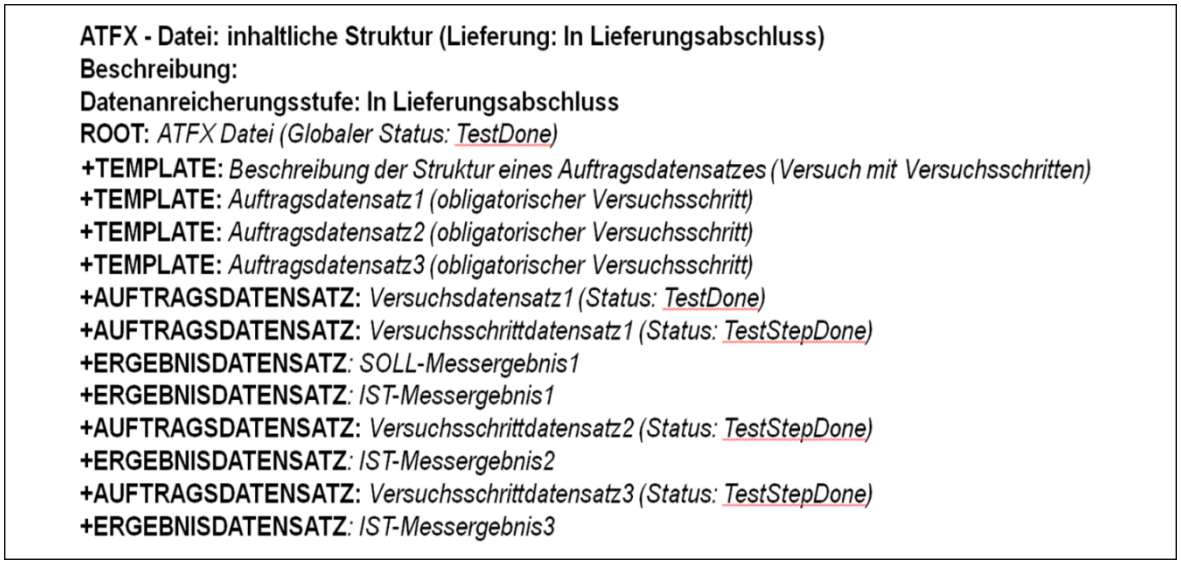


Abbildung 5-3 Datenanreicherung in der ATFX-Datei

5.4 Referenzierte Dateien

Aus der ATFX-Datei können beim Anreichern der Dateninhalte unter anderem auch Dokumente hinzugefügt werden. Bei Attributen vom Typ DT_EXTERNALREFERENCE oder DS_EXTERNALREFERENCE können Dateien angegeben werden. Die Dateipfade werden relativ an den Attributen in der ATFX-Datei gespeichert. Die angegebenen Dateien werden als referenzierte Dokumente in der FDAD Datei abgelegt (z.B. Abbildung der Einbausituation, Zeichnung). Eigentliche Messdaten sollen jedoch nicht als referenzierte Dateien abgebildet werden.

5.5 Prüfsiegel

Im FDX-Format ist die Verwendung eines Prüfsiegels vorgesehen. In der FDAD Datei ist ein Platzhalter für ein Prüfsiegel vorgesehen (siehe ***Abbildung 5-1***). Das Prüfsiegel dient dem Zweck, einem anderen Nutzer vorab bestätigen zu können, dass die Funktionsdatenaustauschdatei den vom Datenanforderer gestellten Anforderungen entspricht. Das Prüfsiegel soll folgende Zwecke erfüllen:

- Einhaltung der ATFX-Syntax
- Vollständigkeit der Daten
- Enthaltener Mindestumfang der Pflichtattribute
- Einhaltung der Datentypen
- Einhaltung der Wertebereiche

Eine Vorgabe über Aufbau, Erzeugung und Verifizierung des Prüfsiegels ist nicht Bestandteil der Version 1.0 der VDA Empfehlung 5550.

6 Mindestanforderungen an den Datenaustausch

Produktbezogene Daten sind in besonderem Maße schützenswerte Daten. Dem sicheren Umgang und Austausch von Produktdaten kommt daher eine sehr hohe Bedeutung zu.

Explizit sei hier darauf hingewiesen, dass die Daten im FDX-Format, d.h. der zip-Container, die beinhaltete ATFX-Datei und die beinhalteten referenzierten Dateien unverschlüsselt sind. Deshalb wird empfohlen die Hinweise unter Kapitel 6 zu beachten.

6.1 Datensicherheit

Produktbezogene Daten sind in der Regel vertraulich klassifiziert und sind durch geeignete Sicherungsmaßnahmen vor dem Zugriff Unbefugter zu schützen.

Die Ablage von Daten sowohl auf lokalen oder mobilen Datenträgern, als auch auf zentralen Speichermedien soll in strukturierter Form erfolgen. Schutzbedürftige Daten, die sich auf lokalen oder mobilen Datenträgern befinden, sollten verschlüsselt abgelegt werden, hierbei sind entsprechende Verschlüsselungsmechanismen einzusetzen.

6.2 Datenaustausch

Der Datenaustausch findet direkt zwischen den beteiligten Vertragspartnern statt. Das Normalverfahren ist ein manueller Prozess, bei dem die Dateien durch berechtigte und registrierte Nutzer auf dedizierte Kundenportale hochgeladen bzw. von diesen heruntergeladen werden. Die Datensicherheit wird dabei in der Regel durch https (TLS) gewährleistet.

Eine weitere Möglichkeit besteht in der Nutzung sicherer automatisierter Datenaustauschverfahren, die in der Automobilindustrie etabliert sind:

1. OFTP2 über das Internet: dabei wird die Datenübertragung mindestens durch TLS gesichert, zusätzlich können die Daten verschlüsselt werden, so dass eine Desk-to-Desk Sicherheit gewährleistet werden kann. Dieses Protokoll ist das verbreitetste Datenaustauschverfahren in der Automobilindustrie.
2. OFTP1 über ENX: dabei wird das Managed Virtual Private Network der Automobilindustrie für die Sicherung der Übertragung von Unternehmen zu Unternehmen verwendet, sofern vorhanden.
3. Sonstige VPN-Lösungen, die in der Regel eine individuelle Registrierung und Konfiguration benötigen.

In allen Fällen sollte der Datenaustausch durch entsprechende Quittungen belegt werden. Bei Datei up- und -download kann das über E-Mails erfolgen, beim OFTP-Verfahren sollte die End-to-End-Response verwendet werden.

Den spezifischen Anforderungen des Vertragsgegenstandes ist vor gemeinsamem Projektstart durch Abstimmung des Datenaustausches zwischen den Vertragspartnern Rechnung zu tragen.

7 Normative Referenzen

ASAM ODS

ASAM ODS ist ein Standard zur Datenablage von Messdaten im Umfeld der Automobilindustrie und umfasst sowohl Messdaten, als auch Metadaten (Prüfstandsaufbau, Parameter) und Dateianhänge.

Die vorliegende VDA Empfehlung baut auf dem ASAM ODS Standard auf.

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

openMDM4

openMDM4 ist eine Erweiterung des ASAM ODS Standards um z.B. Vorlagen abbilden zu können.

Die vorliegende VDA Empfehlung baut auf dem ASAM ODS und dem openMDM4 Standard auf.

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

URI werden u.a. verwendet, um Namensräume in XML-Schemadateien (XSD) zu identifizieren.

UTF-8

ISO 10646 Character Transformation Format

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

ATFX benutzt den UTF-8 Zeichensatz, womit die meisten gängigen Sprachen unterstützt werden.

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

Die XML-Syntax ist das technische Format für die ATFX-Dateien.

XML-Namespaces (XML-Namensräume)

Namespaces in XML 1.0 , Third Edition W3C, Tim Bray et al., eds., 8 December 2009 http://www.w3.org/TR/REC-xml-names.

Mit Namensräumen werden die in XML verwendeten Informationsbausteine (Elemente und Attribute) einer bestimmten Inhaltsdomäne (Verantwortungsbereich) zugeordnet.

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-Schemata beschreiben das verwendbare Vokabular (Elemente und Attribute) und die Struktur von XML-Dokumenten (Instanzen). Sie werden auch zur Prüfung der syntaktischen Korrektheit von XML-Dokumenten genutzt.

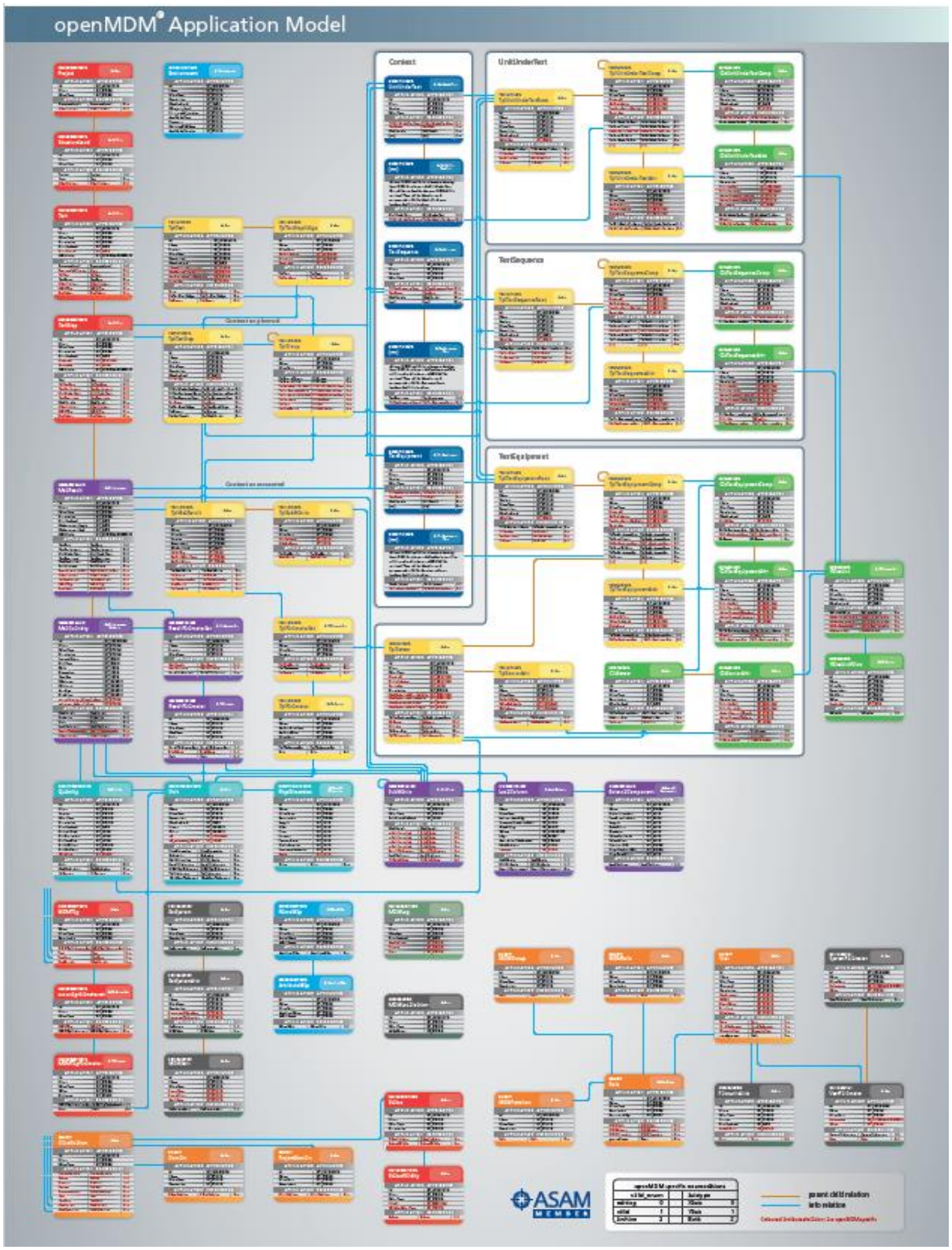
8 Anhang

8.1 Anhang A: Begriffsbestimmungen und Definitionen

Term	Definition / Beschreibung
Applikationsmodell	Das Applikationsmodell spezifiziert, welche Elemente aus dem Basismodell für einen definierten Anwendungsfall tatsächlich genutzt werden.
ASAM	**A**ssociation for **S**tandardisation of **A**utomation and **M**easuring Systems - Internationale Organisation für die Standardisierung von Messdaten und deren Austauschformaten
ASAM ODS	ASAM ODS ist ein Standard zur Datenablage von Messdaten im Umfeld der Automobilindustrie und umfasst sowohl Messdaten, als auch Metadaten (Prüfstandsaufbau, Parameter) und Dateianhänge.
ATFX-Datei	ASAM Transportformat in XML: eine ATFX-Datei ist das konkrete XML-Dokument, das zum Austausch der Daten (Messauftrag, Messdaten …) verwendet wird.
Attribut	Ein Attribut ist eine definierte Eigenschaft eines zu prüfenden Objekts. Die Messdaten beschreiben die Ausprägung dieser Eigenschaft am physischen oder virtuellen Objekt.
Basismodell	Ein Set definierter Basiselemente und der Beziehungen zwischen ihnen.
Betriebsgröße	hier: konkrete physikalische Größe, unter der ein bestimmtes Messverfahren an einem Objekt durchgeführt wird (z.B. Temperatur, Drehzahl ...) .
Datenanforderer	Partner, der die Messdaten anfordert / die Messung beauftragt.
Datenlieferant	Partner, der die Messdaten liefert / die Messung durchführt.
Datenmodell	Modellhafte Abbildung der Daten- und ggf. semantischen Strukturen der Informationen, die in den Austauschdateien (Instanzen) übertragen oder in Datenbanken gespeichert werden.
D-U-N-S Nummer	Eindeutige Identifizierungsnummer eines Unternehmens oder einer Betriebsstätte, zugewiesen von Dun & Bradstreet.
Fachadministrator	Verantwortliche Rolle für die Entwicklung, Verwaltung und Publikation von Template-Spezifikationen für standardisierte Messaufträge zu definierten Baugruppen.
FA-Master	Fachadministration Master - Template-Spezifikation für standardisierte Messaufträge zu allen (bislang) definierten Baugruppen
FA-OEM	Fachadministration OEM – Vorlagen Spezifikation für bauteilspezifische Messaufträge, die auf Basis der FA-VDA Vorlage OEM spezifisch angepasst wurden und als konkrete Vorlage zwischen OEM und Zulieferer genutzt werden soll
FA-VDA	Fachadministration VDA – Vorlagen Spezifikation für bauteilspezifische Messaufträge als Ergebnis der Abstimmung des VDA AKs FDX
FDAD	Funktionsdatenaustauschdatei: Containerformat mit ATFX-Datei, ggf. zusätzlichen Referenzdateien und Prüfsiegel; Dateiendung: *.FDX

.fdtc-Datei	FDAD- Vorlage welche vom Fachadministrator erstellt wird, um sie dem Datenanforderer und dem Lieferanten zur weiteren Befüllung zur Verfügung zu stellen. Durch die Anwendung, sprich die Befüllung durch den Datenanforderer und den Datenlieferanten wird aus der .fdtc-Datei eine .fdxc-Datei. (D.h. letztendlich dies wird über die Dateiendung .fdtc oder .fdxc ersichtlich.)
.fdxc-Datei	normale FDAD, wie sie zum Datenaustausch zwischen Anforderer und Lieferanten verwendet wird. (Unterscheidung zwischen .fdtc und .fdxc-Datei siehe .fdtc-Datei.)
FDAD-Template	Ein FDAD-Template wird vom Fachadministrator als Vorlage bereitgestellt, um konkrete Messaufträge für spezifizierte Baugruppen oder Teile auszulösen.
FDX	Funktionsdatenaustausch (Functional Data Exchange), Name des Arbeitskreise und des Datenformats.
Funktionsdaten	Die Gesamtheit der Daten, die entsprechend eines Messauftrags für einen bestimmten Prüfling (real oder virtuell) durch Messung, Schätzung oder Berechnung unter definierten Bedingungen ermittelt wurden.
Katalogkomponente	Aggregation von Beschreibungsattributen, die funktional eine zusammengehörige Einheit bilden und in verschiedenen Applikationsmodellen wiederverwendet werden können / sollen.
Kennfeld	Auch Kennlinienfeld: stellt mehrere Kennlinien in Abhängigkeit von weiteren Eingangsgrößen (Parameter) in Form einer Kennlinienschar oder in einem dreidimensionalen Koordinatensystem dar.
Kennlinie	Eine Kennlinie ist die graphische Darstellung von zwei voneinander abhängigen physikalischen Größen.
Messauftrag	Auftrag zur Durchführung von Messungen an einem Objekt, zusammen mit den zu beachtenden Parametern und der zu verwendenden Messvorrichtung.
OMG	Object Management Group - internationales Konsortium für die Entwicklung von Technologiestandards für die Modellierung und Integration von Softwaresystemen. Zu diesen Standards gehören u.a. die Unified Modelling Language ® (UML).
openMDM®4	Eine Sammlung von Komponenten und Konzepten, die genutzt werden können, um Anwendungen für Messdatenmanagement zu entwickeln. Die Entwicklung erfolgt durch die openMDM® Eclipse Arbeitsgruppe.
XHTML	XML konforme Version der Hypertext Markup Language HTML.
XML Dokument	Inhaltlich zusammengehörige Datenstruktur, die aus ein oder mehreren Dateien bestehen kann.

8.2 Anhang B: Open MDM Applikationsmodell



8.3 Anhang C: Erweiterungen zum openMDM Applikationsmodell um „Blockregeln“ und „einfachen Regeln“ und um Attribute zur Unterscheidung von Muss- und Kann-Attributen bei Auftrag und Lieferung

Dies ist eine Ergänzung zu Abschnitt „Erweiterungen zum openMDM Applikationsmodell “ der VDA_Empfehlung_5550.

Vorwärtsreferenz auf weitere Anhänge

Konkret werden wie im Klassendiagramm in ***Abbildung 8-1*** ersichtlich dem aktuellen openMDM4 Applikationsmodell die Klassen `ComponentVisibility` und `AttributeVisibility` vom ASAM ODS Basistyp `AoAny` neu hinzugefügt und die bestehenden Klassen `TplUnitUnderTestAttr`, `TplTestSequenceAttr` und `TplTestEquipmentAttr` werden jeweils um ein neues Attribut ergänzt:

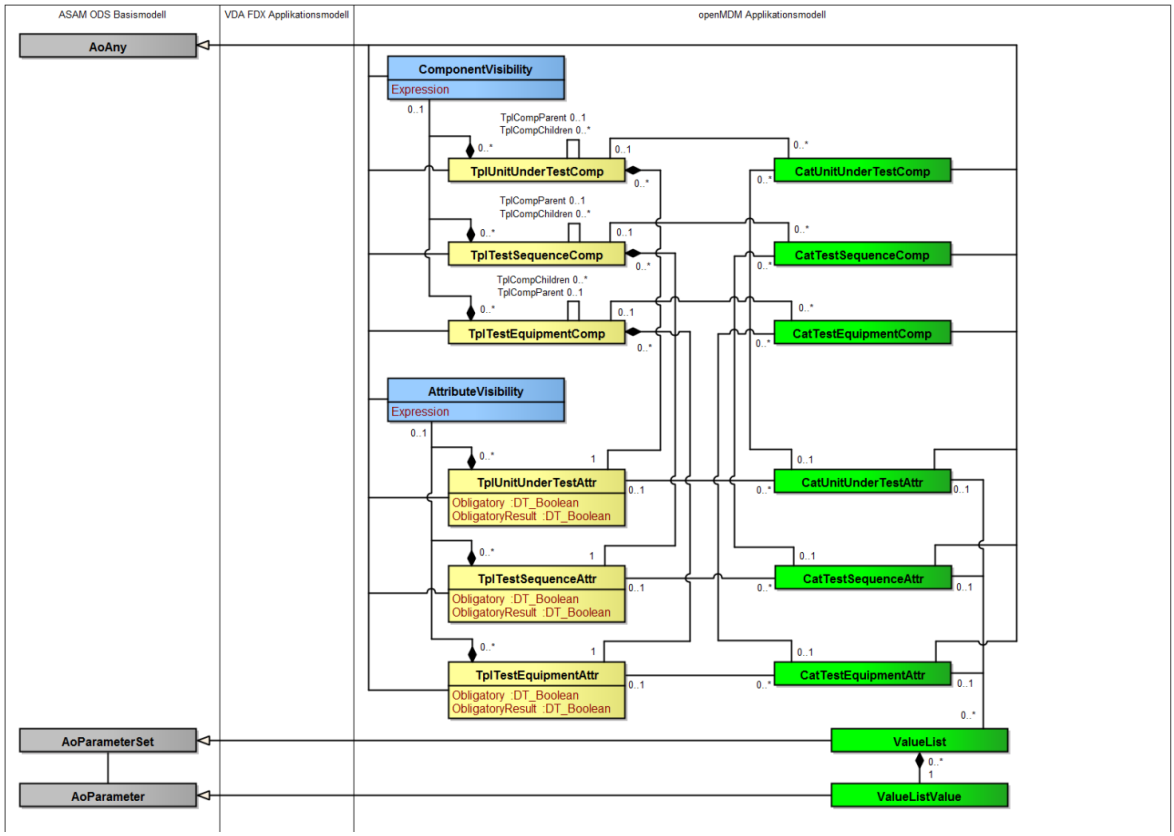


Abbildung 8-1 VDA FDX Neuerungen am openMDM4 Applikationsmodell

Steuerung der Sichtbarkeit von Komponenten und Attributen

Durch die Steuerung der Sichtbarkeit, das heißt die Aktivierung oder Deaktivierung von Komponenten und Attributen mittels der neuen Applikationselemente `ComponentVisibility` und `AttributeVisibility`, sollen "Blockregeln" für Komponenten und „einfache Regeln“ für Attribute ermöglicht werden, ***Abbildung 8-1*****Fehler! Verweisquelle konnte nicht gefunden werden.**.

Eine „Blockregel“ definiert, dass bei Auswahl eines bestimmten Attributwertes verschiedene davon abhängige Komponenten aktiviert oder deaktiviert werden. Eine „einfache Blockregel“ definiert, dass bei Auswahl eines bestimmten Attributwertes verschiedene davon abhängige Attribute derselben Komponente aktiviert oder deaktiviert werden.

- Das Applikationselement `ComponentVisibility` wird neu eingeführt. Dieses definiert die „Blockregeln“ für die Sichtbarschaltung (Aktivierung oder Deaktivierung) von Komponenten auf Basis von Attributwerten. Ein Applikationselement `ComponentVisibility` bestimmt eine Komponente, die im spezifischen openMDM Applikationsmodell unter der angegebenen Bedingung auftreten muss. Diese Komponente muss dann aus dem angegebenen Template hervorgehen.

  Unter Bezug auf die in ***Abbildung 3-12*** mit den Ziffern 2 und 3 gekennzeichneten Zeilen wäre es unter der Bedingung, dass das Attribut *Messungstyp.Messung mit Vorlast* mit dem Wert 'ja' belegt ist, erforderlich, die Attribute der Subkategorie *Vorlast* mit Werten zu belegen. Mit dem Wert 'nein' wäre die Subkategorie *Vorlast* unzulässig.

  Hierzu würde für die Subkategorie *Preload* ein Applikationselement `ComponentVisibility` angelegt, das im Attribut `ComponentVisibility.Expression` einen Bezug auf das Attribut `Measurement.Preload` herstellt und dessen Wert auf ‘ja‘ testet. Dazu verweist dieses Applikationselement `ComponentVisibility` auf ein Applikationselement `TplTestSequenceComp`, das wiederum auf ein Applikationselement `CatTestSequenceComp` verweist, welche der Klasse des Applikationselements `Preload` entspricht. Wenn die Bedingung erfüllt ist, muss das spezifische Applikationsmodell das Applikationselement `Preload` als Komponente des gegebenen Applikationselements `TestSequence` enthalten.

- Das Applikationselement `AttributeVisibility` wird neu eingeführt. Dieses definiert die „einfachen Regeln“ für die Sichtbarschaltung von Attributen auf Basis von Attributwerten und deren Kombinationen.

  Unter Bezug auf die in ***Abbildung 3-12*** mit der Ziffer 4 gekennzeichnete Zeile wäre es unter der Bedingung, dass sowohl das Attribut *Vorlast.Steuerungsart* mit dem Wert 'kraftgesteuert‘ und das Attribut *Messgerätebetriebsgrößen.Vorlast.Bewegungsrichtung* mit dem Wert 'translatorisch' belegt sind, erforderlich, für das Attribut *Vorlast.Vorlast Kraft* einen Wert anzugeben, nicht jedoch für das Attribut *Vorlast.Vorlast Moment*.

  Hierzu würde für das Attribut *Vorlast.Vorlast Kraft* ein Applikationselement `AttributeVisibility` angelegt, das im Attribut `ComponentVisibility.Expression` einen Bezug auf die Attribute `Preload.ControlType` und `Preload.DirectionOfMotion` herstellt sowie deren Werte auf ‘Force‘ beziehungsweise ‚translational‘ testet. Dazu verweist dieses Applikationselement `AttributeVisibility` auf ein Applikationselement `TplTestSequenceAttr`, das wiederum auf ein Applikationselement `CatTestSequenceAttr` verweist, welche dem Attribut `Preload.Force` entspricht. Wenn die Bedingung erfüllt ist, muss im spezifischen Applikationsmodell das Attribut `Preload.Force` mit einem Wert befüllt werden.

- In beiden Fällen werden die Regeln als Java-EL (ExpressionLanguage) Ausdrücke hinterlegt.

Unterscheidung der Pflicht- oder Kann-Attribute für Auftrag und Datenlieferung

Im aktuellen openMDM4 Applikationsmodell ist es nicht möglich, unterschiedliche Pflichtattribute für Auftragsdaten und Messdaten in einer Auftragsvorlage zu modellieren. Um diese Funktion nachzubilden, ist es bislang notwendig, mehrere Auftragsvorlagen zu definieren.

Durch folgende Erweiterung ist diese Unterscheidung zukünftig möglich:

- Die Applikationselemente `TplUnitUnderTestAttr`, `TplTestSequenceAttr` und `TplTestEquipmentAttr` werden um das Attribut *ObligatoryResult* erweitert. Der Wert legt fest, ob das jeweilige Attribut für Messdaten ein Pflichtattribut ist oder nicht. Das bisherige Attribut *Obligatory* ist damit nur noch für die Auftragsdaten gültig. Dadurch kann zwischen Auftragskontext und Messkontext unterschieden werden.

8.4 Anhang D: Zusammenhang ASAM ODS Basismodell, openMDM Applikationsmodell und VDA FDX Applikationsmodell am Beispiel Gummilager

Dies ist eine Ergänzung zu Abschnitt „Zusammenhang ASAM ODS Basismodell, openMDM Applikationsmodell und VDA FDX Applikationsmodell am Beispiel Gummilager“ der VDA-Empfehlung 5550.

Das Klassendiagramm in ***Abbildung 8-2*** stellt den Zusammenhang zwischen dem ASAM ODS Basismodell, dem openMDM Applikationsmodell und dem VDA FDX Applikationsmodell am Beispiel des Gummilagers dar. Es enthält eine Auswahl wichtiger Klassen aus dem ASAM ODS Basismodell und aus dem openMDM Applikationsmodell sowie Klassen aus dem VDA FDX Applikationsmodell zum Gummilager. Die abgebildeten Klassen repräsentieren ausgewählte Teile des Auftrags (Order) und der Datenlieferung (Delivery) von Messaufträgen. Die vertikale Ordnung von oben nach unten folgt im Wesentlichen dem Prozess der sukzessiven Befüllung mit Daten, d.h. der Reihenfolge in der die Daten vom Datenanforderer und Datenlieferant nacheinander eingeben werden.

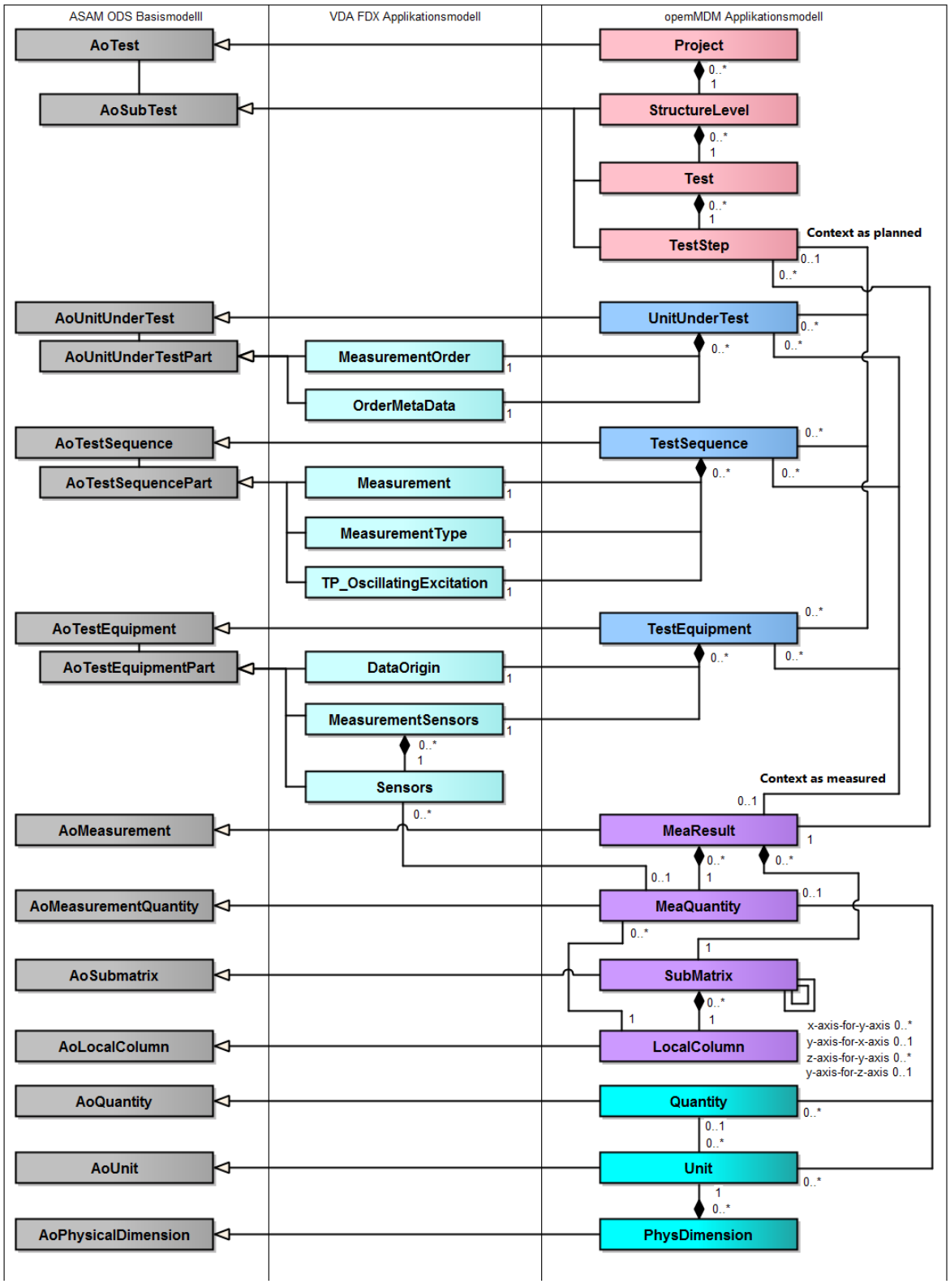


Abbildung 8-2 Zusammenhang ASAM ODS Basismodell, openMDM Applikationsmodell und VDA FDX Applikationsmodell am Beispiel des Gummilagers (Übersicht)

Die Organisation der Messaufträge mittels der Applikationselemente `Project, StructureLevel, Test, TestStep`

Die Klassen `Project`, `StructureLevel`, `Test` und `TestStep` liefern die Möglichkeiten zur Organisation von Messaufträgen und haben folgende Assoziationen:

- Eine ATFX-Datei beinhaltet 0..* Applikationselemente `Project` vom ASAM ODS Basistyp `AoTest`, welches die Wurzel zur Organisation von Messaufträgen bildet (dieses und die nachfolgenden Applikationselemente sind rot hinterlegt). Eine ATFX-Datei mit 0 Applikationselementen `Project` enthält damit noch keinen Messauftrag.
- Ein Applikationselement von `Project` umfasst 0..* Applikationselemente `StructureLevel` vom ASAM ODS Basistyp `AoSubTest`. Die Applikationselemente `Project` und `StructureLevel` organisieren Testvorhaben.
- Ein Applikationselement von `StructureLevel` umfasst 0..* Applikationselemente `Test` vom ASAM ODS Basistyp `AoSubTest`. Das Applikationselement `Test` repräsentiert ein Testvorhaben im Ganzen.
- Ein Applikationselement von `Test` umfasst 0..* Applikationselemente `TestStep` vom ASAM ODS Basistyp `AoSubTest`. Das Applikationselement `TestStep` repräsentiert den Testschritt eines solchen Testvorhabens.

Eine Festlegung zur Nutzung der Applikationselemente `Project` und `StructureLevel` ist nicht Gegenstand dieser Empfehlung, weil das fachliche VDA FDX Datenmodell dazu keine Aussage macht. Gegebenenfalls müssen zwischen dem Datenanforderer und dem Datenlieferanten entsprechende Regelungen gefunden werden. Ansonsten bleiben diese Applikationselemente unberücksichtigt, und alle über die Applikationselemente `Project` und `StructureLevel` erreichbaren Applikationselemente `TestStep` sind gemäß den nachfolgenden Beschreibungen zu berücksichtigen.

Die Klasse `TestStep` hat Assoziationen mit den Klassen `UnitUnderTest`, `TestSequence` und `TestEquipment`. Diese Assoziationen entsprechen den in der openMDM-Übersicht mit „Context as planned“ gekennzeichneten Assoziationen und sind somit im Rahmen der Bestellung relevant.

Die Daten zu einer Datenanforderung und Datenlieferung unter den Applikationselementen `UnitUnderTest, TestSequence, TestEquipment`

Die Klassen `UnitUnderTest`, `TestSequence, TestEquipment` spezifizieren über ihre Komponenten alle auftragsbezogenen Informationen, insbesondere zum Messobjekt, zu den Messgeräten und zum Testablauf.

- Ein Applikationselement `TestStep` verweist auf 0..1 bestimmtes Applikationselement `UnitUnderTest` vom ASAM ODS Basistyp `AoUnitUnderTest`, auf 0..1 bestimmtes Applikationselement `TestSequence` vom ASAM ODS Basistyp `AoTestSequence` und auf 0..1 bestimmtes Applikationselement `TestEquipment` vom ASAM ODS Basistyp `AoTestEquipment` (diese drei Applikationselemente sind blau hinterlegt). Es bildet somit das Bindeglied zwischen Messobjekt, Testablauf und Testmethode
- Ein Applikationselement `UnitUnderTest` umfasst 0..* für das spezifische openMDM Applikationsmodell (Gummilager, Dämpfer, etc.) zulässige Komponenten vom ASAM ODS Basistyp `AoUnitUnderTestPart`. Für das Gummilager ist das in ***Abbildung 8-2*** auszugsweise dargestellt anhand der Klassen `MeasurementOrder` und `OrderMetaData` (diese sind hellblau hinterlegt). Alle für das Gummilager zulässigen Komponenten von `UnitUnderTest` und deren Funktion sind weiter unten im ***Abbildung 8-3*** aufgeführt.
- Ein Applikationselement `TestSequence` umfasst 0..* für das spezifische openMDM Applikationsmodell zulässige Komponenten vom ASAM ODS Basistyp `AoTestSequencePart`. Für das Gummilager ist das in ***Abbildung 8-2*** auszugsweise dargestellt anhand der Klassen `Measurement`, `MeasurementType` und

`TP_OscillatingExcitation` (diese sind hellblau hinterlegt). Alle für das Gummilager zulässigen Komponenten von `Testsequence` und deren Funktion sind weiter unten in ***Abbildung 8-6*** aufgeführt.

- Das Applikationselement der Klasse `TP_OscillatingExcitation` steht für das konkrete *Messprogramm Oszillierende Anregung*.
- Ein Applikationselement `TestEquipment` umfasst 0..* für das spezifische openMDM Applikationsmodell zulässige Komponenten vom ASAM ODS Basistyp `AoTestEquipmentPart`. Für das Gummilager ist das in ***Abbildung 8-2*** auszugsweise dargestellt anhand der Klassen `DataOrigin` und `MeasurementSensors` (diese sind hellblau hinterlegt). Alle für das Gummilager zulässigen Komponenten von `Testequipment` und deren Funktion sind weiter unten im ***Abbildung 8-9*** aufgeführt.
- Ein Applikationselement `MeasurementSensors` umfasst 0..* Komponenten `Sensors` vom ASAM ODS Basistyp `AoTestEquipmentPart`, wobei die Klasse `MeasurementSensors` für das Messprogramm steht und die Klasse `Sensors` für einen Messwert der Messkurve (diese sind hellblau hinterlegt).

Die Funktionsdaten zu einer Datenanforderung und Datenlieferung in den Applikationselementen `MeaResult, MeaQuantity, SubMatrix und LocalColumn`

Die Klassen `MeaResult`, `MeaQuantity`, `SubMatrix` und `LocalColumn` spezifizieren die Messdaten:

- Ein Applikationselement `TestStep` umfasst 0..* Applikationselemente `MeaResult` vom ASAM ODS Basistyp `AoMeasurement`, wobei die Klasse `MeaResult` für ein Messergebnis als Ganzes steht (dieses ist violett hinterlegt).
- Ein Applikationselement `MeaResult` umfasst 0..* Applikationselemente `MeaQuantity` vom ASAM ODS Basistyp `AoMeasurementQuantitiy`, wobei die Klasse `MeaQuantity` nur für eine bestimmte Messgröße des Messergebnisses steht – `MeaQuantity` selbst enthält keine Messwerte (dieses ist violett hinterlegt).
- Ein Applikationselement `MeaQuantity` verweist auf 0..1 bestimmtes Applikationselement `Sensors` (Zeit, Geschwindigkeit, Weg, Kraft, Winkel, Frequenz etc.) mit dem die Messung vorgenommen werden soll, bzw. wurde. In `Sensors` wird neben der Art des Sensors auch die Achse definiert auf der die zu messende Größe aufgetragen wird (z.B. x-Achse, y-Achse.)
- Ein Applikationselement `MeaQuantity` verweist auf 0..1 bestimmtes Applikationselement `Quantity` vom ASAM ODS Basistyp `AoQuantity`, das die gemessene Größe spezifiziert (Zeit, Geschwindigkeit, Weg, Kraft, Frequenz etc.) (dieses ist türkis hinterlegt). In `Quantity` wird neben der zu messenden bzw. der gemessenen Größe auch der zughörige Datentyp z.B. Floating definiert.
- Ein Applikationselement `MeaQuantity` verweist auf 0..1 bestimmtes Applikationselement `Unit` vom ASAM ODS Basistyp `AoUnit`, das die zu messende bzw. die gemessene Einheit spezifiziert (z.B. Hertz, Kilohertz, Newton, Kilonewton, Meter, Millimeter, m/s, m/s^2) (dieses ist türkis hinterlegt).
- Ein Applikationselement `Unit` verweist auf 1 bestimmtes Applikationselement `PhysDimension` vom ASAM ODS Basistyp `AoPhysicalDimension`, das zu messende bzw. gemessene physikalische Größe spezifiziert (Zeit, Geschwindigkeit, Länge, Kraft Frequenz, Dimensionslos) (dieses ist türkis hinterlegt).
- Ein Applikationselement `MeaResult` umfasst 0..* Applikationselemente `SubMatrix` vom ASAM ODS Basistyp `AoSubmatrix`, welche das Messergebnis strukturieren (dieses ist violett hinterlegt). Dadurch ist es möglich auch ein mehrdimensionales

Messergebnis über verschiedene Messgrößen abzubilden, beispielsweise wenn eine periodische Anregung sowohl in der x- als auch in der y-Richtung erfolgt.

- Ein Applikationselement `SubMatrix` umfasst 0..* Applikationselemente `LocalColumn` vom ASAM ODS Basistyp `AoLocalColumn`, wobei ein Applikationselement `LocalColumn` die eigentlichen Messdaten einer bestimmten Messgröße enthält (dieses ist violett hinterlegt), denn `LocalColumn` verweist auf genau ein Applikationselement `MeaQuantity`.
- Das Applikationselement `SubMatrix` besitzt noch zwei reflexive Assoziationen, d.h. Assoziationen mit sich selbst. Damit kann das Messergebnis z.B. im mehrdimensionalen Raum noch weiter strukturiert werden. Dieser Aspekt wird hier jedoch nicht weiter ausgeführt.

Hinweis: Die im Anhang B vorliegende aktuelle Übersicht zum openMDM Applikationsmodell stellt die für das Verständnis der Messkurve und ihren Messwerten notwendigen Zusammenhänge nur unzureichend dar. Diese Zusammenhänge werden in mit den Klassen `MeasurementSensors`, `Sensors` und `MeaQuantity` und ihren Assoziation berücksichtigt.

Fehler! Verweisquelle konnte nicht gefunden werden. zeigt die spezifischen Informationen der jeweils 5 pplikationselemente `Sensors`, `MeaQuantity`, `Quantity`, `Unit` und `PhysDimension` für das *Messprogramm Statische Kraft-Weg Kennlinie* eines Gummilagers. Für das Messprogramm werden fünf Sensorelemente ausgewählt, ein Zeitsensor, ein Sensor der die Anzahl der Hystereseschleifen misst, ein Kraftsensor, ein Wegsensor und ein Geschwindigkeitssensor, d.h. es gibt 5 Instanzen von Sensors. Über die entsprechenden 5 Instanzen von `MeaQuantity` und die zugehörigen Beziehungen werden dann die gewünschten 5 Datentypen (5 Instanzen von `Quantity`), 5 Einheiten (5 Instanzen von `Unit`) und über `Unit` die physikalischen Größen (5 Instanzen von `PhysDimension`) für die Definition der Messung (Anforderung) und für die tatsächlich durchgeführte Messung (Lieferung) den Sensoren zugeordnet.

Tabelle 8-1 Applikationselemente Sensors, MeaQuantity, Quantity, Unit und PhysDimension für das Messprogramm Statische Kraft-Weg-Kennlinie im Falle des Gummilagers (CurveForceDisplacementStatic).

Nr.	**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*

Die in ***Abbildung 8-2*** dargestellten Assoziationen zwischen den Klassen `MeaResult` und `UnitUnderTest`, `TestSequence` und `TestEquipment` entsprechen den in der openMDM-Übersicht (siehe Anhang) mit „Context as measured“ gekennzeichneten Assoziationen, welche der Datenlieferung (Delivery) entsprechen.

Diese Assoziationen spezifizieren das tatsächlich verwendete Messobjekt (`UnitUnderTest`), den tatsächlich durchgeführten Testablauf mit seinem tatsächlich durchgeführten Testprogramm (`TestSequence`), die tatsächlich angewandte Testmethode mit der tatsächlich aufgenommenen Messkurve (`TestEquipment`) sowie den Ist-Datensatz. Bezüglich der `UnitUnderTest`, `TestSequence` und `TestEquipment` untergeordneten Komponenten gelten die vorangegangenen Beschreibungen zur Beauftragung.

Sollte ein Unterschied zwischen Datenlieferung und Bestellung vorliegen, müssen entsprechende Applikationselemente `UnitUnderTest`, `TestSequence` und `TestEquipment` angelegt werden, auch wenn sich nur eine ihrer untergeordneten Komponenten ändert.

Eine fachliche Vertiefung des Themas „Zusammenhang ASAM ODS Basismodell, openMDM Applikationsmodell und VDA FDX Applikationsmodell am Beispiel Gummilager“ findet sich in Anhang E: Die Komponenten der Applikationselemente `UnitUnderTest`, `TestSequence` und `TestEquipment` und ihr Zusammenhang mit den Kategorien des VDA FDX Applikationsmodells und in Anhang F: Die modellgetriebenen Aspekte des openMDM Applikationsmodells.

Eine technische Vertiefung des Themas „Zusammenhang ASAM ODS Basismodell, openMDM Applikationsmodell und VDA FDX Applikationsmodell am Beispiel Gummilager“ findet sich in Anhang G: Präsentation der modellgetriebenen Aspekte des openMDM4 Applikationsmodells in ATFX und Anhang H: Das XML Schema des ASAM ATFX-Datenaustauschformats zu finden.

8.5 Anhang E: Die Komponenten der Applikationselemente `UnitUnderTest`, `TestSequence` und `TestEquipment` und ihr Zusammenhang mit den Kategorien des VDA FDX Applikationsmodells

Dies ist eine Ergänzung zu Abschnitt „Zusammenhang ASAM ODS Basismodell, openMDM Applikationsmodell und VDA FDX Applikationsmodell am Beispiel Gummilager“ der VDA_Empfehlung_5550.

Die Komponenten des Applikationselements `UnitUnderTest`

Das Klassendiagramm in *Fehler! Verweisquelle konnte nicht gefunden werden.* stellt alle für das ummilager zulässigen Komponenten des Applikationselements `UnitUnderTest` dar. Diese Komponenten stellen alle Ausprägungen des ASAM ODS Basistyps `AoUnitUnderTestPart` dar.

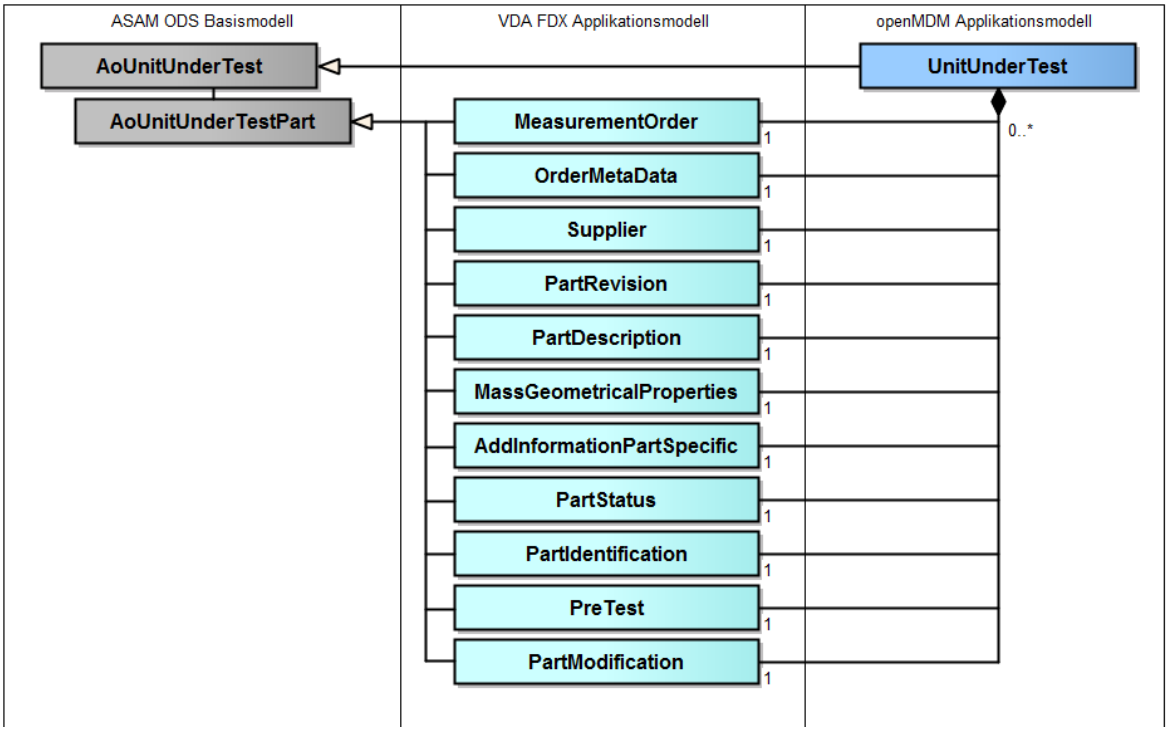


Abbildung 8-3 Zulässige Komponenten des Applikationselements UnitUnderTest am Beispiel des Gummilagers

Die Klasse UnitUnderTest beinhaltet über die ihr zugeordneten Komponenten die Informationen zu den Hauptkategorien Messauftrag, Auftragsmetadaten, Bauteilspezifische Zusatzinformationen und Prüfling.

Das Klassendiagramm in ***Abbildung 8-4*** stellt die im VDA FDX Datenmodell vorhandenen Haupt- und Subkategorien (diese sind beige hinterlegt) den im openMDM Applikationsmodell vorhandenen Komponenten zu UnitUnderTest (diese sind hellblau hinterlegt) gegenüber. Ein gestrichelter Pfeil ordnet eine Haupt- bzw. Subkategorie des VDA FDX Datenmodells (siehe Abschnitt 3.8) der entsprechenden Komponente des Applikationselements UnitUnderTest beim openMDM Applikationsmodell zu. Das Bild zeigt auch, dass die hierarchische Struktur in Haupt- und Subkategorien im openMDM nicht abgebildet wird.

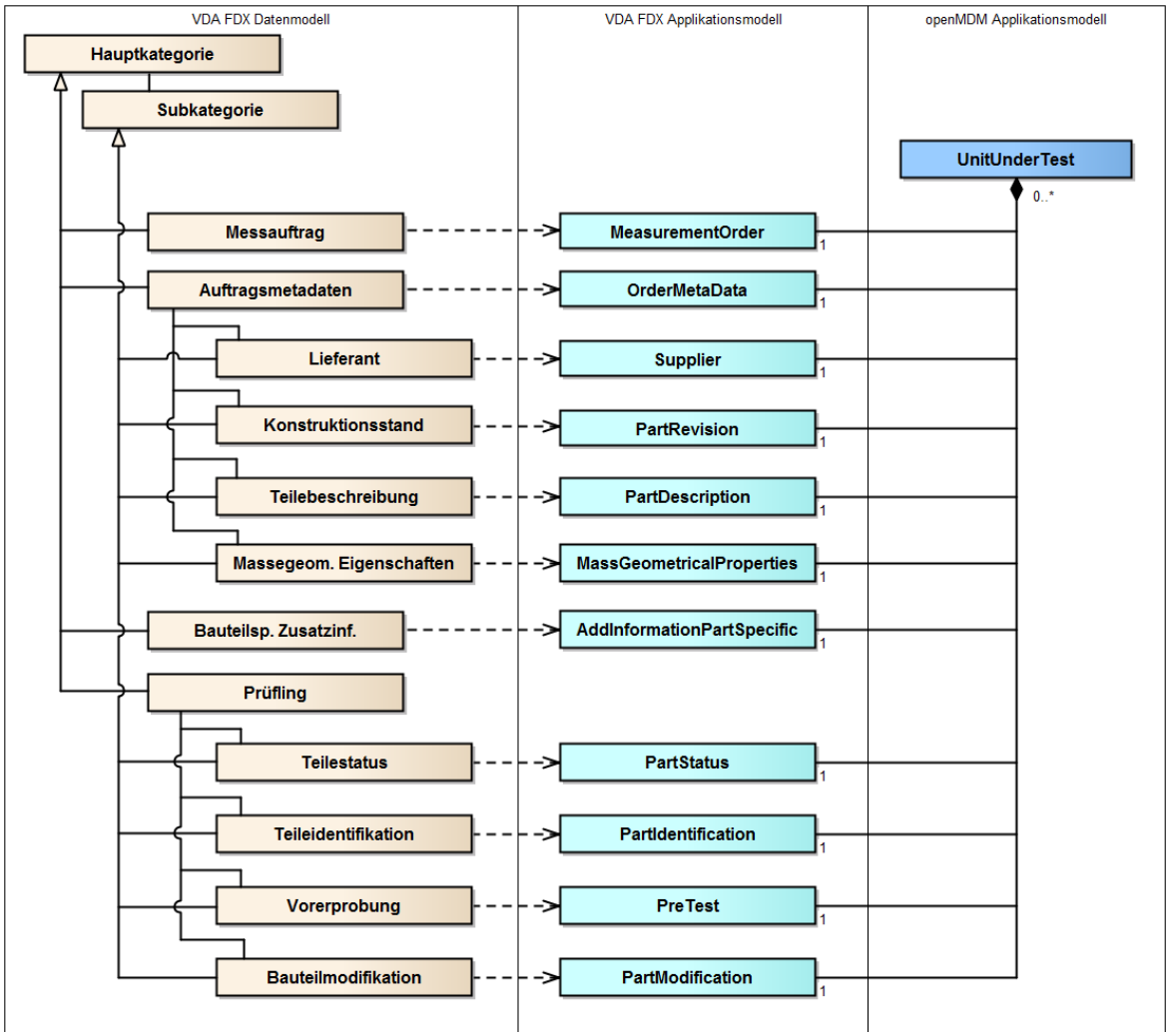


Abbildung 8-4 Zusammenhang der Haupt- und Subkategorien des VDA FDX Datenmodells mit den Komponenten des Applikationselements UnitUnderTest beim VDA FDX Applikationsmodells am Beispiel des Gummilagers

Die Attribute der Haupt- und Subkategorien finden sich entsprechend in den Applikationselementen des openMDM Applikationsmodells wieder. Im Klassendiagramm in ***Abbildung 8-5*** sind dies beispielsweise für die Subkategorie Messauftrag, die auf das VDA-FDX-Applikationselement `MeasurementOrder` abgebildet wird, die *Attribute CustomerIdentifcationNumber, OrderNumber, RequesterName, Splittability* und *SubOrderNumber.*

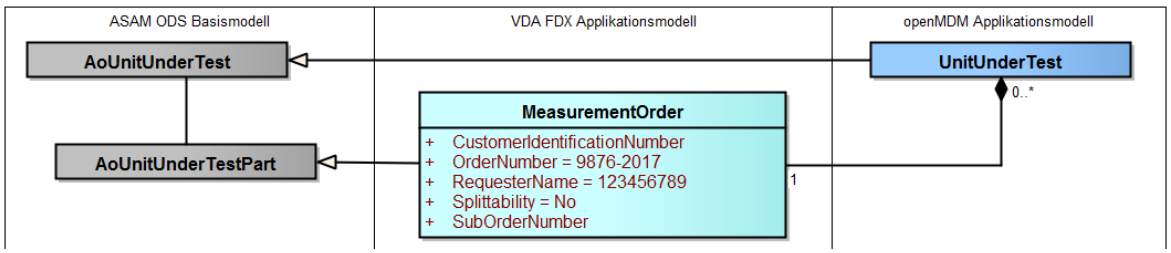


Abbildung 8-5 Attribute des Applikationselements MeasurementOrder und mögliche Werte

Hinweis: In ***Abbildung 8-5*** werden mögliche Werte des Applikationselements `MeasurementOrder` mittels der Default-Werte der UML gedeutet. Diese Darstellung ist zwar nicht konform zur UML, aber sie wurde aus Gründen der Vereinfachung gewählt.

Hinweis: Das openMDM Applikationsmodell lässt für die genannten untergeordneten Applikationselemente zu, dass diese nicht nur einfach, sondern auch mehrfach zur gleichen Klasse auftreten können (Mehrfachinstanziierung der gleichen Klasse). Dieser Fall tritt z.B. bei der Auswahl einer *Vorkonditionierung* mit dem gleichen Testprogramm wie die eigentliche *Messung* auf. Um die beiden Testprogramme, welche unter dem gleichen `TestStep` bzw. ihrer `TestSequence` liegen, unterscheiden zu können, ist es erforderlich, diese am Attribut *Measurement.TestProgram* bzw. *PreCondition.TestProgram* zu identifizieren.

Die Komponenten des Applikationselements `TestSequence`

Das Klassendiagramm in ***Abbildung 8-6*** stellt alle für das Gummilager zulässigen Komponenten des Applikationselements `TestSequence` dar. Diese Komponenten sind alle Ausprägungen des ASAM ODS Basistyps `AoTestSequencePart`.

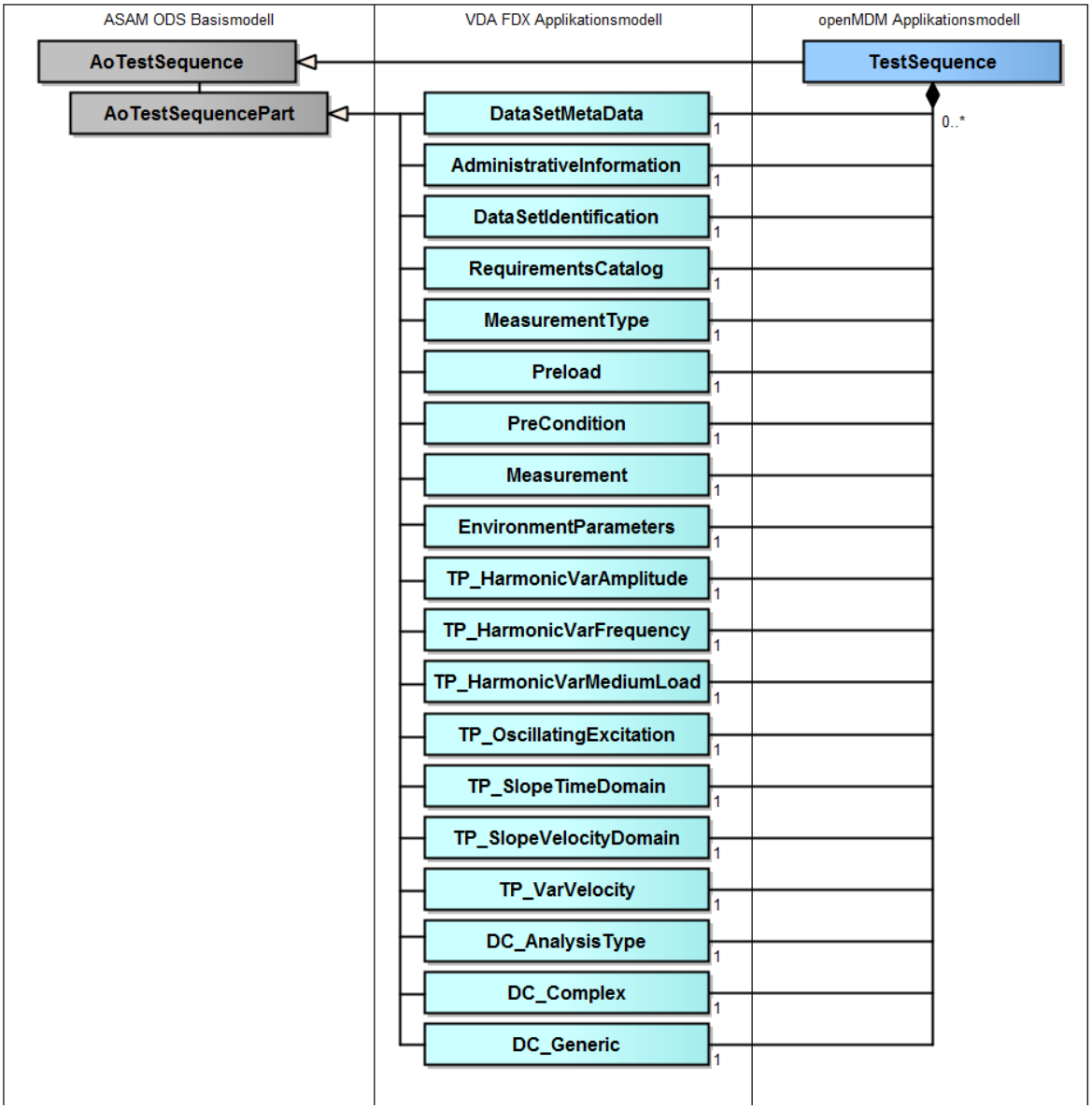


Abbildung 8-6 Zulässige Komponenten des Applikationselements TestSequence am Beispiel des Gummilagers

Die Klasse `TestSequence` beinhaltet über die ihr zugeordneten Komponenten (diese sind hellblau hinterlegt) die Informationen zu den Hauptkategorien *Datensatzmetadaten*,

Messgerätebetriebsgrößen und *Abgeleitete Kenngrößen*. Analog zu ***Abbildung 8-4*** stellt das Klassendiagramm in ***Abbildung 8-7*** die im VDA FDX Datenmodell vorhandenen Haupt- und Subkategorien (diese sind beige hinterlegt) den im openMDM Applikationsmodell vorhandenen Komponenten des Applikationsmodells `TestSequence` (diese sind hellblau hinterlegt) gegenüber. Das Bild zeigt auch hier, dass die hierarchische Struktur in Haupt- und Subkategorien im openMDM nicht abgebildet wird.

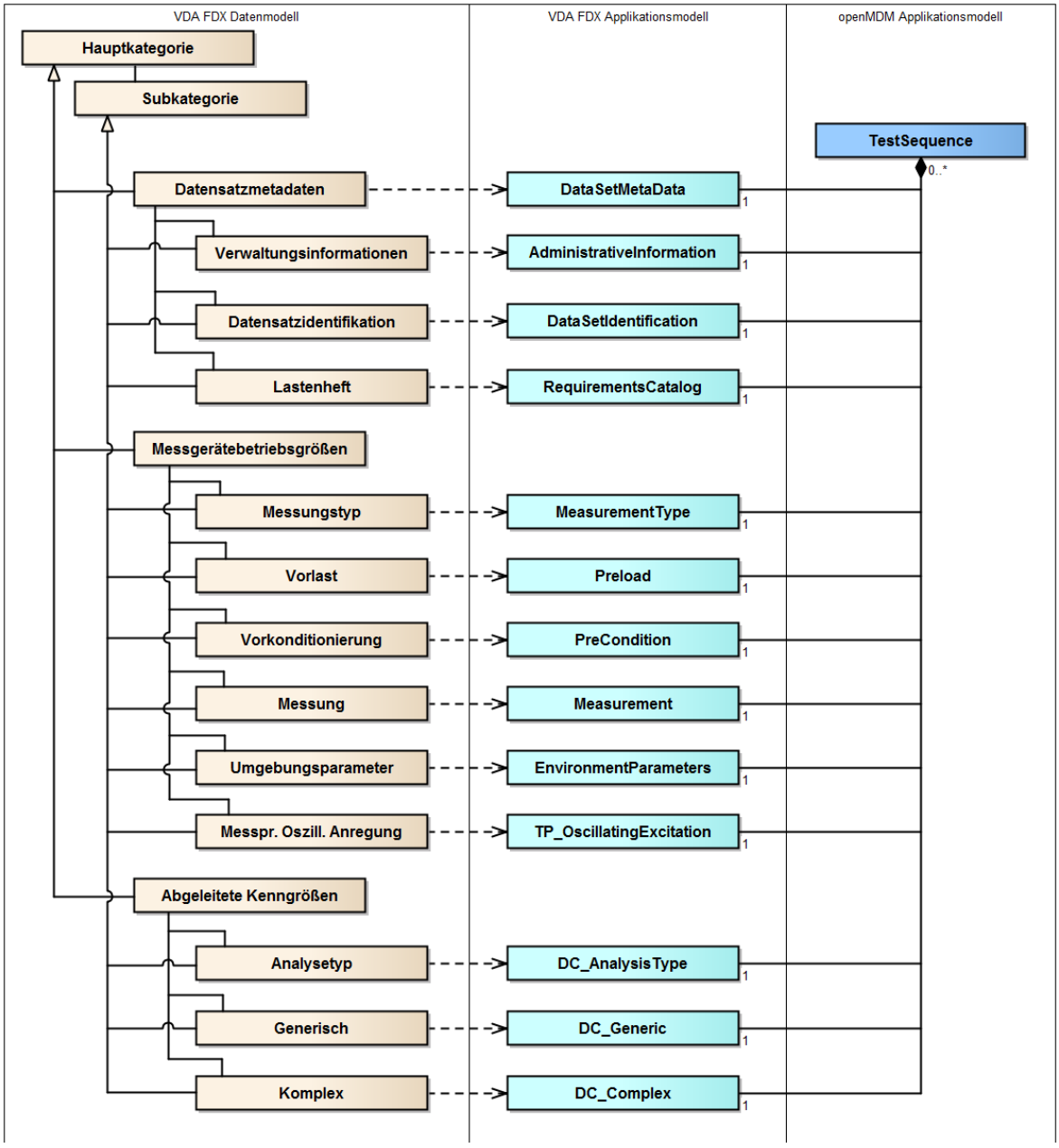


Abbildung 8-7 : Zusammenhang der Haupt- und Subkategorien des VDA FDX Datenmodells mit den Komponenten des Applikationselements TestSequence beim VDA FDX Applikationsmodell am Beispiel des Gummilagers

Die Attribute der Haupt- und Subkategorien finden sich entsprechend in den Applikationselementen des VDA-FDX Applikationsmodells wieder. Im Klassendiagramm in ***Abbildung 8-8*** sind dies beispielsweise für die Subkategorie *Messungstyp*, die auf das openMDM-Applikationselement `MeasurementType` abgebildet wird, die Attribute *DirectionOfMotion*, *MeasurementDynamics*, *Preconditioning*, *Preload*, *SamplingRate*, *SpatialDirection*, *TimeChannelType* und *TimeIncrement*.

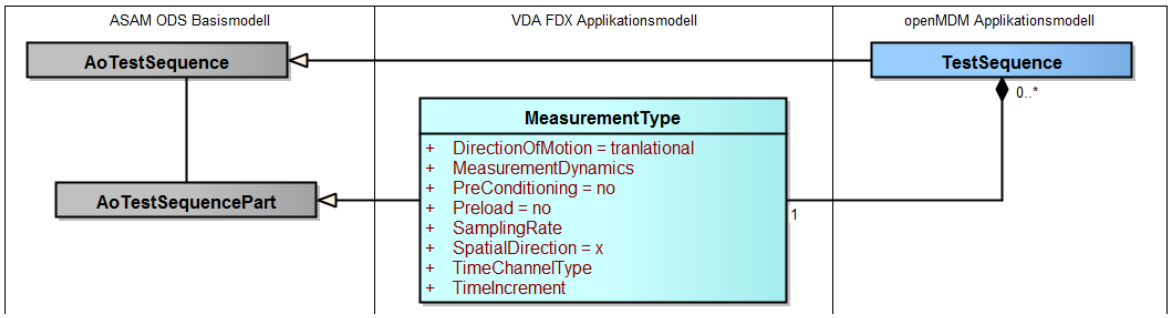


Abbildung 8-8 Attribute des Applikationselements MeasurementType und mögliche Werte

Hinweis: Für ***Abbildung 8-8*** gilt der gleiche Hinweis zu den Default-Werten der UML wie nach ***Abbildung 8-5***.

Die Komponenten des Applikationselements `TestEquipment`

Das Klassendiagramm in ***Abbildung 8-9*** stellt alle für das Gummilager zulässigen Komponenten des Applikationselements `TestEquipment` dar. Diese Komponenten sind alle Ausprägungen des ASAM ODS Basistyps `AoTestEquipmentPart`.

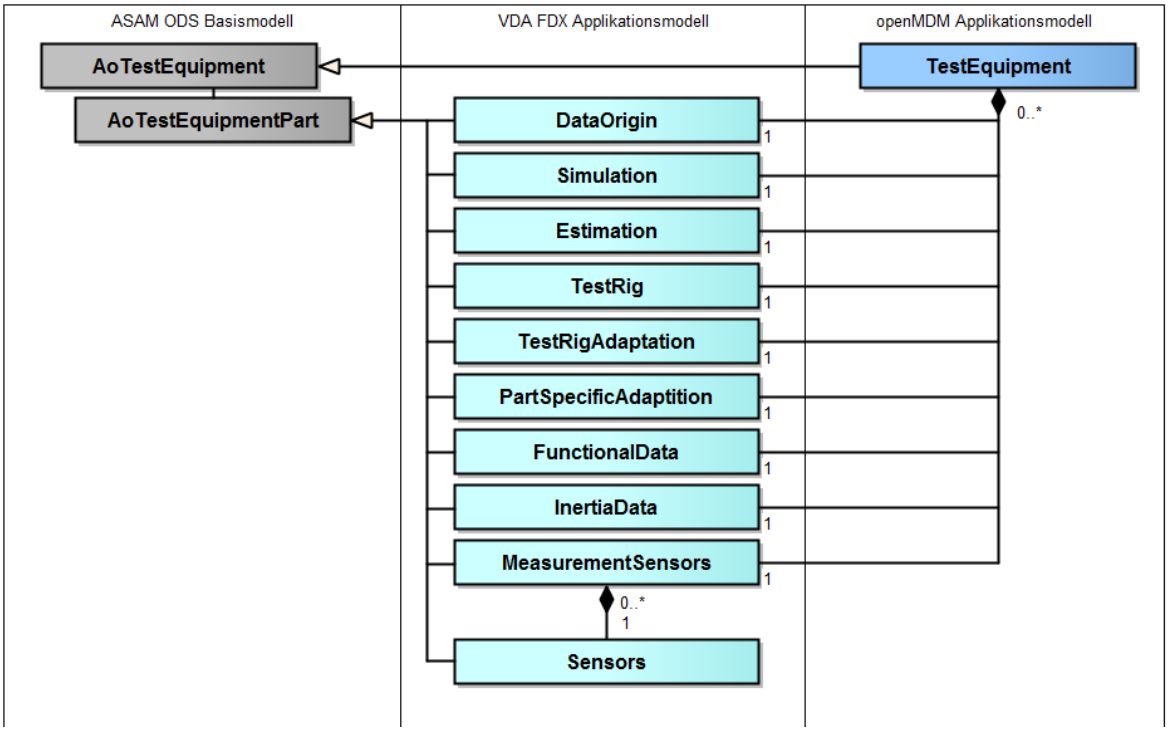


Abbildung 8-9 Zulässige Komponenten des Applikationselements TestEquipment am Beispiel des Gummilagers

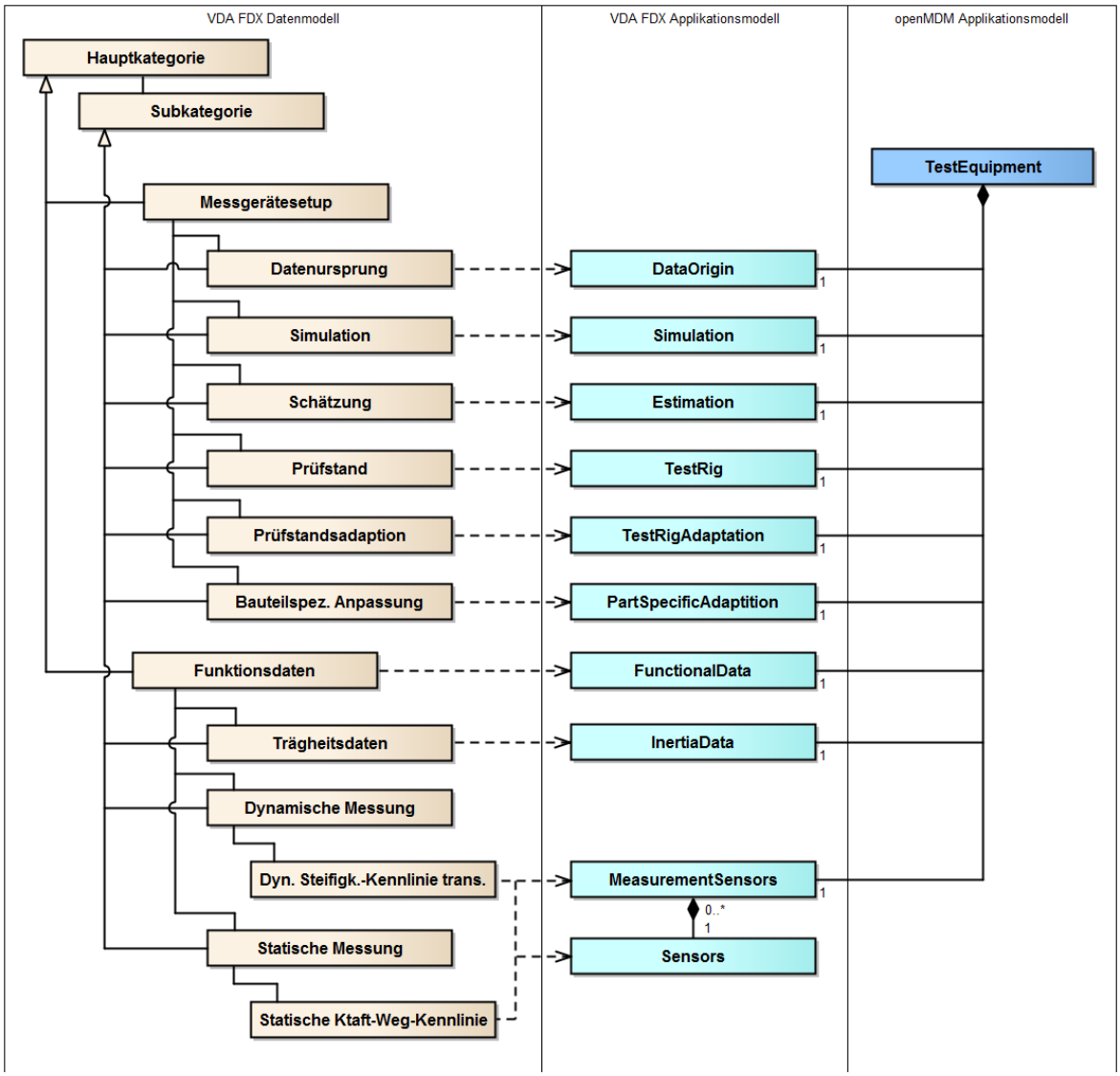


Abbildung 8-10 Zusammenhang der Haupt- und Subkategorien des VDA FDX Datenmodells mit den Komponenten des Applikationselements TestEquipment beim VDA FDX Applikationsmodell am Beispiel des Gummilagers

Die Attribute der Haupt- und Subkategorien finden sich entsprechend in den Applikationselementen des openMDM Applikationsmodells wieder. Im Klassendiagramm in ***Abbildung 8-11***. ist dies beispielsweise für die Subkategorie *Datenursprung*, die auf das Applikationselement `DataOrigin` abgebildet wird, das Attribut *Data Generation*.

Für die Subkategorien *Statische Messung* und *Dynamische Messung* der Hauptkategorie *Funktionsdaten* wird für jedes einzelne Attribut des VDA FDX Applikationsmodells jeweils ein eigenes Applikationselement angelegt. Im Klassendiagramm in ***Abbildung 8-11*** wird beispielsweise in der Subkategorie *Statische Messung* die *Messkurve Statische Kraft-Weg Kennlinie* auf das Applikationselement `MeasurementSensors` abgebildet, und ihre Messwerte werden jeweils auf ein Applikationselement `Sensors` abgebildet.

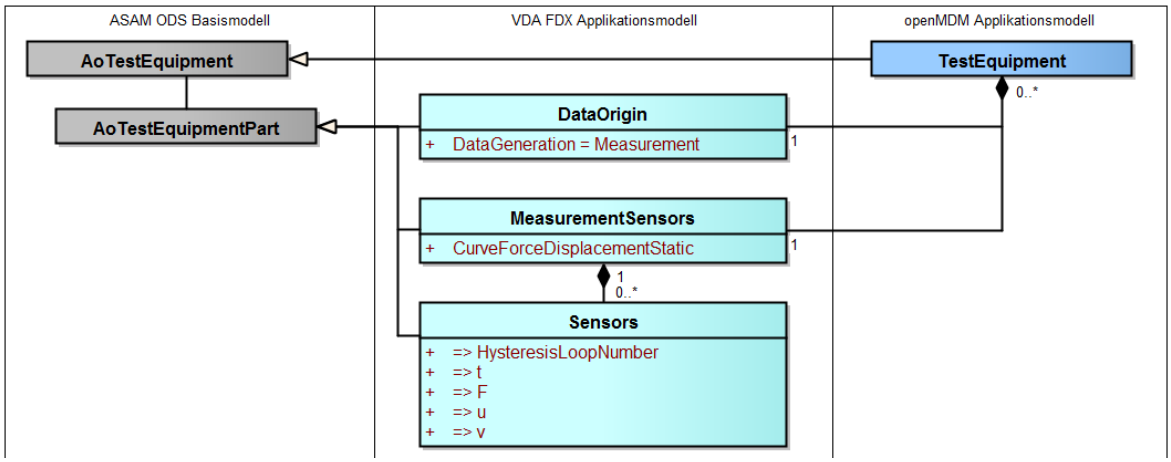


Abbildung 8-11 Attribute der Applikationselemente DataOrigin, MeasurementSensors und Sensors und mögliche Werte

Hinweis: Für ***Abbildung 8-11*** gilt der gleiche Hinweis zu den Default-Werten der UML wie nach ***Abbildung 8-5***.

Hinweis: In ***Abbildung 8-11*** wird die Messkurve als Attribut 'CurveForceDisplacementStatic' des Applikationselements `MeasurementSensors` gedeutet. Auf gleiche Weise werden die Messwerte als Attribute '=> HysteresisLoopNumber','=> t','=> F','=> u' und '=> v' des Applikationselements `Sensors` gedeutet. Diese Darstellung ist zwar nicht konform zur UML, sie wurde aber aus Gründen der Vereinfachung und Vereinheitlichung mit den Darstellungen in ***Abbildung 8-8*** und ***Abbildung 8-5*** gewählt. In Anhang F: Die modellgetriebenen Aspekte des openMDM Applikationsmodells und darin insbesondere in Fehler! Verweisquelle konnte nicht gefunden werden., ***Abbildung 8-13*** und ***Abbildung 8-14*** wird auf die Instanziierung von Applikationselementen ausführlich eingegangen.

8.6 Anhang F: Die modellgetriebenen Aspekte des openMDM Applikationsmodells

Dies ist eine Ergänzung zu Abschnitt „Zusammenhang ASAM ODS Basismodell, openMDM Applikationsmodell und VDA FDX Applikationsmodell am Beispiel Gummilager“ der VDA_Empfehlung_5550.

Die modellgetriebenen Aspekte des openMDM Applikationsmodells

Die modellgetriebenen Aspekte des openMDM Applikationsmodells sind primär für den Fachadministrator und sekundär für den Datenanforderer relevant, aber nicht für den Datenlieferanten, weil dieser ausschließlich mit einem spezifischen openMDM Applikationsmodell umgeht.

Die in den bisherigen Abschnitten beschriebenen Applikationselemente können in zwei Gruppen eingeteilt werden.

Die erste Gruppe bilden diejenigen Applikationselemente, die allen spezifischen openMDM Applikationsmodellen gemeinsam sind, also typischerweise die Klassen `UnitUnderTest`, `TestSequence`, `TestEquipment`, `MeaResult`, `MeaQuantity`, `SubMatrix` oder `LocalColumn`. Diese Anwendungselemente werden in den mit „openMDM Applikationsmodell“ überschriebenen Schwimmbahnen dargestellt (diese sind blau bzw. violett hinterlegt).

Die zweite Gruppe bilden die für ein spezifisches openMDM Applikationsmodell vorgesehenen Applikationselemente. Diese Klassen sind Komponenten der Klassen `UnitUnderTest`, `TestSequence` und `TestEquipment`, beispielsweise im Falle des Gummilagers die Klasse `MeasurementOrder` als Komponente von `UnitUnderTest`, die Klasse `MeasurementType` als Komponente von `TestSequence`, und die Klasse `DataOrigin` als Komponente von `TestEquipment`. Diese Anwendungselemente werden in den mit „VDA FDX Applikationsmodell“ überschriebenen Schwimmbahnen dargestellt (diese sind hellblau hinterlegt).

Hierbei können Applikationselemente der zweiten Gruppe durchaus in verschiedenen spezifischen openMDM Applikationsmodellen vorkommen, beispielsweise die oben genannten Klassen `MeasurementOrder`, `MeasurementType` und `DataOrigin` in den VDA FDX Applikationsmodellen zum Stoßdämpfer, Stützlager, Zusatzfeder, Gummilager, Aufbaufeder, Abkoppelelement oder Stabilisator. Dies ist auch nicht überraschend, da diese und andere Klassen allgemeine und modellübergreifende Informationen beschreiben.

Daneben gibt es in der zweiten Gruppe aber auch Applikationselemente, die nur in bestimmten spezifischen openMDM Applikationsmodellen vorkommen, beispielsweise die Klasse `PartSpecificAdaption` als Komponente von `TestEquipment` in den beiden openMDM Applikationsmodellen zum Stoßdämpfer und zur Zusatzfeder, oder die Klasse `TP_VarVelocity` als Komponente von `TestSequence` aktuell ausschließlich im openMDM Applikationsmodell zum Dämpfer.

Zudem ist es bei der zweiten Gruppe von Applikationselementen durchaus üblich, dass bestimmte Attribute der Applikationselemente nur in bestimmten spezifischen openMDM Applikationsmodellen vorkommen. Beispielsweise gibt es den Messwert *=> CurrentFeed* des *Messprogramms Oszillierende Anregung* aktuell nur beim VDA FDX Applikationsmodell zum Stoßdämpfer.

Diese beiden Aspekte, zum einen, dass bestimmte Applikationselemente nur in bestimmten spezifischen openMDM Applikationsmodellen und zum anderen, dass in diesen Applikationselementen nur bestimmte Attribute vorkommen können, werden mit dem Begriff „Bauteilgültigkeit“ zusammengefasst.

Diese „Bauteilgültigkeit“ darf nicht mit der in Abschnitt 4.3.1 „Erweiterungen zum openMDM Applikationsmodell“ beschriebenen Steuerung der Sichtbarkeit von Komponenten und Attributen verwechselt werden. Die Steuerung der Sichtbarkeit erlaubt die Definition von Bedingungen, bei deren Erfüllung ein hier beschriebenes Template zur Anwendung kommt beziehungsweise bei Hinfälligkeit der Bedingungen die Wirkung dieses Templates wieder zurückgenommen wird.

Die folgenden Abschnitte beschreiben die wesentlichen modellgetriebenen Aspekte des openMDM Applikationsmodells, die der Ausprägung eines spezifischen openMDM Applikationsmodells dienen. Diese erlauben insbesondere die Festlegung, welche Testschritte eines zusammenhängenden

Testvorhabens durchzuführen sind, welche Applikationselemente der zweiten Gruppe in die Testschritte aufgenommen werden, und welche Attribute dieser Applikationselemente für die Belegung mit Daten zur Verfügung stehen.

Das Klassendiagramm in *Abbildung 8-12* stellt die Zusammenhänge der modellgetriebenen Aspekte des openMDM Applikationsmodells dar. In Templates und Katalogen wird definiert, welche Applikationselemente und welche Attribute dieser Applikationselemente für ein spezifisches openMDM Applikationsmodell, z.B. Gummilager, zulässig sind. Die hier neu vorgestellten Klassen repräsentieren diese Templates und Kataloge und sind in den zugehörigen Klassendiagrammen gelb beziehungsweise grün hinterlegt.

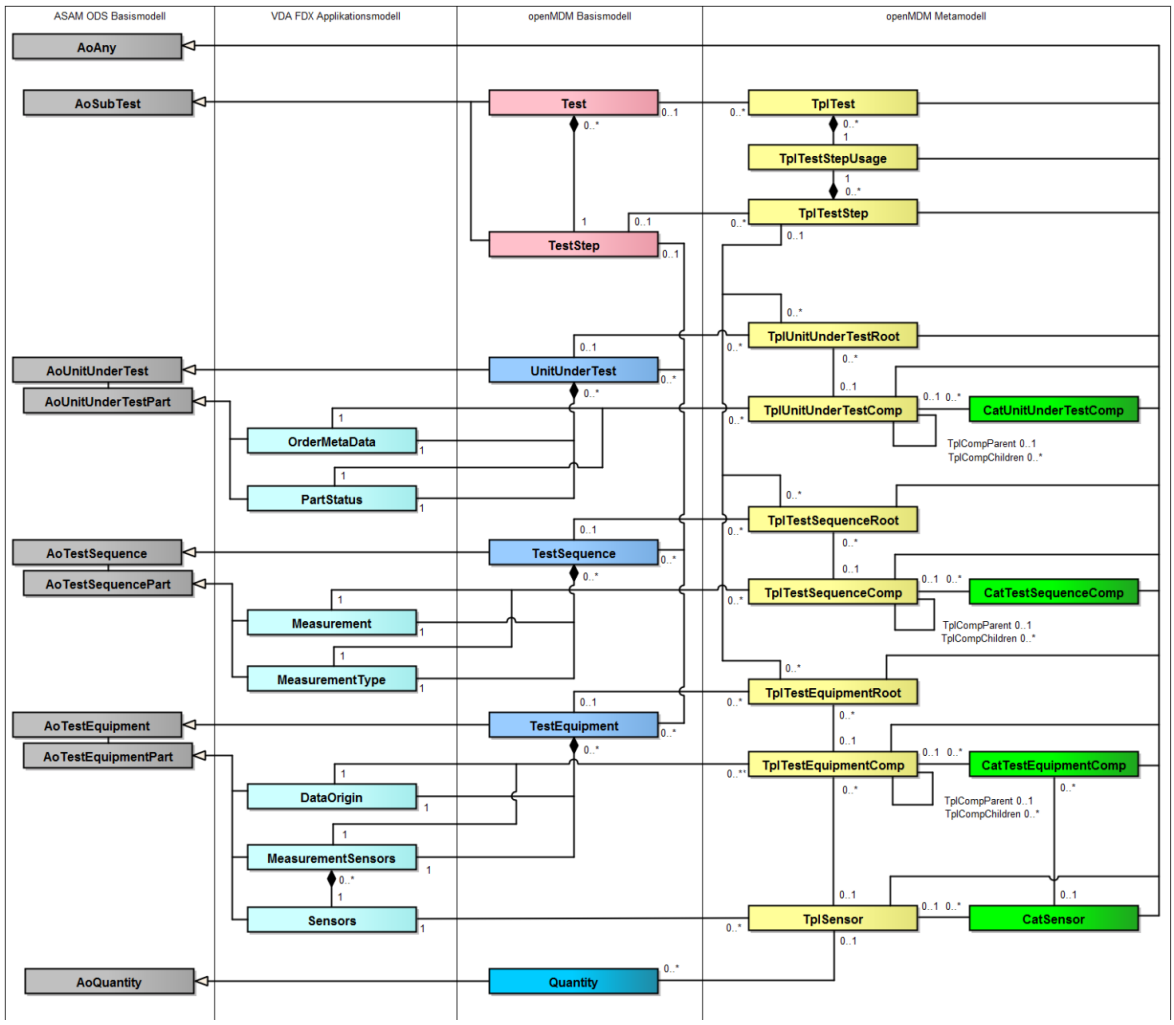


Abbildung 8-12 Modellgetriebene Aspekte des openMDM Applikationsmodells und Verwendung von Templates und Katalogen und Zusammenhang mit dem ASAM ODS Basismodell und dem VDA FDX Applikationsmodell am Beispiel des Gummilagers

Alle der hier neu vorgestellten Klassen für Templates und Kataloge sind vom ASAM ODS Basistyp `AoAny`.

Um die ausschließlich für die modellgetriebenen Aspekte des openMDM Applikationsmodells vorgesehenen Applikationselemente von der ersten Gruppe der Applikationselemente, die allen spezifischen openMDM Applikationsmodellen gemeinsam sind, in *Abbildung 8-12* einfach unterscheiden zu können, werden die ersteren in der Schwimmbahn 'ASAM ODS Basismodell' und die zweiteren in der Schwimmbahn 'openMDM Metamodell' angeordnet.

Die folgende Fehler! Verweisquelle konnte nicht gefunden werden. zeigt im Sinne eines bjektdiagramms am Beispiel des Gummilagers das Szenario eines Testvorhabens mit den Templates und Katalogen für einen Messauftrag mit zwei dynamischen und einer quasistatischen Messung:

Tabelle 8-2 Szenario eines Testvorhabens mit den Templates und Katalogen für einen Messauftrag am Beispiel des Gummilagers

Test / Instanz	TestStep / Instanz	<> / Instanz	Cat<>Comp / Instanz	CatSensor / Instanz	TplTest / TplTestStepUsage	TplTestStep / TplTestStepUsage	Tpl<>Root	Tpl<>Comp / Ebene 1	Tpl<>Comp / Ebene 2	Tpl<>Comp / Ebene 3	TplSensor	Cat<>Comp / Klasse	CatSensor / Klasse
Gummilager					Gummilager								
	Dynam. 0,05					Dynam. 1							
<>: UnitUnderTest		Gummilager 001					Gummilager						
			PartIdentification					PartIdentification				PartIdentification	
			PartStatus						PartStatus			PartStatus	
									PreTest			PreTest	
			PartModification						PartModification			PartModification	
			OrderMetaData					OrderMetaData				OrderMetaData	
			MeasurementOrder						MeasurementOrder			MeasurementOrder	
			Supplier						Supplier			Supplier	
			PartRevision						PartRevision			PartRevision	
			PartDescription						PartDescription			PartDescription	
			Mass.Geom.Prop.						Mass.Geom.Prop.			Mass.Geom.Prop.	
			AddInform.PartSpec.					AddInform.PartSpec.				AddInform.PartSpec.	
<>: TestSequence		Dynam. 0,05					Dynam.Harm.Var.Freq.						
			MeasurementType					MeasurementType				MeasurementType	
									Preload			Preload	
									PreCondition				
										TP_PC_Harm.Var.Freq.		TP_Harm.Var.Freq.	
			Measurement						Measurement			Measurement	
			TP_Harm.Var.Freq. 0,01							TP_Harm.Var.Freq.		TP_Harm.Var.Freq.	
			Environm.Param.					Environm.Param.				Environm.Param.	
			DataSetMetaData					DataSetMetaData				DataSetMetaData	
			Admin.Inform.						Admin.Inform.			Admin.Inform.	
			DataSetIdentification						DataSetIdentification			DataSetIdentification	
			RequirementsCatalog						RequirementsCatalog			RequirementsCatalog	
<>: TestEquipment		Curve.Displ.Freq.Trans.					Curve.Displ.Freq.Trans.						
			DataOrigin					DataOrigin				DataOrigin	
									Simulation			Simulation	
									Estimation			Estimation	
			TestRig						TestRig			TestRig	
									TestRigAdaptation			TestRigAdaptation	
			InertiaData					InertiaData				InertiaData	
			FunctionalData						DynamicMeasurement			FunctionalData	
			Curve.Displ.Freq.Trans.							Curve.Displ.Freq.Trans.		MeasurementSensors	
				=> Freq						=>	Freq		Sensors
				=> C						=>	C		Sensors
				=> LossAngle						=>	LossAngle		Sensors
				=> um						=>	um		Sensors
				=> ua						=>	ua		Sensors
				=> Fm						=>	Fm		Sensors
				=> Fa						=>	Fa		Sensors
				=> TE						=>	TE		Sensors
	Dynam. 0,1					Dynam. 2							
<>: UnitUnderTest		Gummilager 001											
<>: TestSequence		Dynam. 0,1					Dynam.Harm.Var.Freq.						
			MeasurementType					MeasurementType				MeasurementType	
									Preload			Preload	
									PreCondition			PreCondition	
										TP_PreCon._Harm.Var.Freq.		TP_Harm.Var.Freq.	
			Measurement						Measurement			Measurement	
			TP_Harm.Var.Freq. 0,05							TP_Harm.Var.Freq.		TP_Harm.Var.Freq.	
			Environm.Param.					Environm.Param.				Environm.Param.	
			DataSetMetaData					DataSetMetaData				DataSetMetaData	
			Admin.Inform.						Admin.Inform.			Admin.Inform.	
			DataSetIdentification						DataSetIdentification			DataSetIdentification	
			RequirementsCatalog						RequirementsCatalog			RequirementsCatalog	
<>: TestEquipment		Curve.Displ.Freq.Trans.											
	Quasistat.					Quasistat.							
<>: UnitUnderTest		Gummilager 001											
<>: TestSequence		Quasistatisch					OscillatingExcitation						
			MeasurementType					MeasurementType				MeasurementType	
									Preload			Preload	
									PreCondition			PreCondition	
										TP_PreCon._Oscill.Excit.		TP_Oscill.Excit.	
			Measurement						Measurement			Measurement	
			TP_Oscill.Excit.							TP_Oscill.Excit.		TP_Oscill.Excit.	
			Environm.Param.					Environm.Param.				Environm.Param.	
			DataSetMetaData					DataSetMetaData				DataSetMetaData	
			Admin.Inform.						Admin.Inform.			Admin.Inform.	
			DataSetIdentification						DataSetIdentification			DataSetIdentification	
			RequirementsCatalog						RequirementsCatalog			RequirementsCatalog	
<>: TestEquipment		ForceDisplacem.					ForceDisplacem.						
			DataOrigin					DataOrigin				DataOrigin	
									Simulation			Simulation	
									Estimation			Estimation	
			TestRig						TestRig			TestRig	
			TestRigAdaptation						TestRigAdaptation			TestRigAdaptation	
			InertiaData					InertiaData				InertiaData	
			FunctionalData						StaticMeasurement			FunctionalData	
			ForceDisplacem.							ForceDisplacem.		MeasurementSensors	
				=> t						=>	t		Sensors
				=> v						=>	v		Sensors
				=> u						=>	u		Sensors
				=> F						=>	F		Sensors
				=> Hy.Loo.Num.						=>	Hy.Loo.Num.		Sensors

Die Tabelle gliedert sich jeweils für jeden Testschritt in drei horizontale Abschnitte für die Komponenten der Applikationselemente `UnitUnderTest`, `TestSequence` und `TestEquipment`. Mit dem Platzhalter „<>“ werden in den Spaltenüberschriften die zugehörigen Klassen der Applikationselemente

angeben. Beispielsweise werden für den Platzhalter „UnitUnderTest“ in den entsprechenden Spalten die Applikationselemente `UnitUnderTest`, `TplUnitUnderTestRoot`, `TplUnitUnderTestComp` und `CatUnitUnderTestComp` angegeben. Auf den Zeilen der Tabelle stehen dann die Namen der Instanzen dieser Applikationselemente. Im Text werden die Namen mit einfachen Anführungszeichen versehen.

Die Auswahl der Testschritte zu einem Testvorhaben mittels der Applikationselemente `TplTest`, `TplTestUsage`, `TplTestStep`

Die Klassen `TplTest`, `TplTestUsage`, und `TplTestStep` repräsentieren Templates für Testvorhaben. Die im vorigen Abschnitt beschriebenen Klassen `Test` und `TestStep` repräsentieren die Testvorhaben, die aus diesen Templates hervorgehen.

Ein konkreter Fall wäre eine Messung des Gummilagers mit drei Testschritten. Die ersten beiden Testschritte sollen dynamische Messungen mit unterschiedlichen Amplituden sein, der dritte Testschritt soll eine quasistatische Messung sein.

Die Multiplizitäten der Klassen `TplTest`, `TplTestUsage` und `TplTestStep` entsprechen einer attributierten N:M-Beziehung. Die Klasse `TplTest` repräsentiert das Template für ein Testvorhaben im Ganzen. Die Klasse `TplTestStepUsage` repräsentiert die Auswahl eines Templates für einen Testschritt und die Klasse `TplTestStep` das Template eines Testschritts.

Das Applikationselement `TplTest` umfasst 0..* Applikationselemente `TplTestUsage`. Die Klasse `TplTestUsage` repräsentiert die Auswahl eines Templates für einen Testschritt. Hierbei verweist das Applikationselement `TplTestUsage` auf ein bestimmtes Applikationselement `TplTestStep`.

Bei Vorgabe einer Instanz von `TplTest` werden über die Kette der Assoziationen zwischen den Klassen `TplTest`, `TplTestStepUsage` und `TplTestStep` entsprechend viele Instanzen von `TplTestStep` gefunden. Für die Instanz von `TplTest` wird eine Instanz der Klasse `Test` und für jede der so gefundenen Instanz von `TplTestStep` wird eine Instanz der Klasse `TestStep` angelegt und der Instanz von `TplTest` zugeordnet.

Im Szenario des Testvorhabens nach **Fehler! Verweisquelle konnte nicht gefunden werden.** ist ein pplikationselement `TplTest` (Name ‘Gummilager‘) erforderlich. Wegen der zwei dynamischen und einer quasistatischen Messung sind drei Applikationselemente `TplTestUsage` erforderlich, die jeweils ein Applikationselement `TplTestStep` (Namen ‘Dynamisch 1‘,‘Dynamisch 2‘ und ‘Quasistatisch‘) auswählen.

Die Auswahl der Applikationselemente zu einem Testschritt mittels der Applikationselemente `TplUnitUnderTestRoot`, `TplTestSequenceRoot`, `TplTestEquipmentRoot`

Dabei können die Templates entsprechend der Hierarchie der Hauptkategorien *Messauftrag*, *Auftragsmetadaten*, *Bauteilspezifische Zusatzinformationen*, *Prüfling*, *Datensatzmetadaten, Messgerätebetriebsgrößen*, *Abgeleitete Kenngrößen*, *Messgerätesetup* und *Funktionsdaten* sowie ihrer Subkategorien angeordnet werden.

Die Klasse `TplUnitUnderTestRoot` repräsentiert das Template, aus der die in den Testschritt aufzunehmenden Komponenten der Klasse `UnitUnderTest` hervorgehen. Die Klasse `TplTestSequenceRoot` repräsentiert das Template, aus der die in den Testschritt aufzunehmenden Komponenten der Klasse `TestSequence` hervorgehen. Die Klasse `TplTestEquipmentRoot` repräsentiert das Template, aus der die in den Testschritt aufzunehmenden Komponenten der Klasse `TestEquipment` hervorgehen.

Hierzu verweist eine Instanz der Klasse `TplTestStep` auf jeweils 0..1 Instanzen der Klasse `TplUnitUnderTestRoot`, `TplTestSequenceRoot` und `TplTestEquipmentRoot`.

Eine Instanz der Klasse `TplUnitUnderTestRoot` verweist auf 0..* Instanzen der Klasse `TplUnitUnderTestComp`. Die Klasse `TplUnitUnderTestComp` repräsentiert eine Auswahl der Klasse einer Komponente der Klasse `UnitUnderTest`. Hierzu verweist eine Instanz der Klasse `TplUnitUnderTestComp` auf eine Instanz der Klasse `CatUnitUnderTestComp`. Die Klasse

`CatUnitUnderTestComp` repräsentiert den Katalog der Klassen von Applikationselementen, die als Komponenten der Klasse `UnitUnderTest` verfügbar sind.

Eine Instanz der Klasse `TplUnitUnderTestComp` kann auch reflexiv auf 0..* andere Instanzen der gleichen Klasse `TplUnitUnderTestComp` verweisen. Dadurch kann ein Baum von Instanzen der Klasse `TplUnitUnderTestComp` aufgespannt werden. Damit können die Templates `TplUnitUnderTestComp` so angeordnet werden, dass sie die Hauptkategorien *Messauftrag*, *Auftragsmetadaten*, *Bauteilspezifische Zusatzinformationen* und *Prüfling* und die entsprechenden Subkategorien hierarchisch organisieren.

Im Szenario des Testvorhabens nach **Fehler! Verweisquelle konnte nicht gefunden werden.** verweist das pplikationselement `TplTestStep` ‘Dynamisch 1‘ auf das Applikationselement `TplUnitUnderTestRoot` ‘Gummilager‘, das einen Baum von Applikationselementen `TplUnitUnderTestComp` aufspannt. Hierzu verweist das Applikationselement `TplUnitUnderTestRoot` ‘Dynamisch 1‘ auf drei Applikationselemente `TplUnitUnderTestComp` ‘PartIdentification‘, ‘OrderMetaData‘ und ‘AddInformationPartSpecific‘. Das Applikationselement `TplUnitUnderTestComp` ‘PartIdentification‘ verweist auf drei Applikationselemente `TplUnitUnderTestComp` ‘PartStatus‘, ‘PreTest‘ und ‘PartModification‘. Das Applikationselement `TplUnitUnderTestComp` ‘OrderMetaData‘ verweist auf drei Applikationselemente `TplUnitUnderTestComp` ‘MeasurementOrder‘, ‘Supplier‘, ‘PartRevision‘, ‘PartDescription‘ und ‘MassGeometricalProperties‘. Das Applikationselement `TplUnitUnderTestComp` ‘AddInformationPartSpecific‘ verweist auf ein Applikationselement `AddInformationPartSpecific` ‘AddInformationPartSpecific‘.

Hinweis: Das Szenario nach **Fehler! Verweisquelle konnte nicht gefunden werden.** bildet hinsichtlich der emplates die Datenkonstellation aus einem spezifischen openMDM Applikationsmodells zum Gummilager ab, die bei der Erstellung dieses Dokuments zur Verfügung stand. Der Baum der Applikationselemente `TplUnitUnderTestComp` beinhaltet zwar alle Hauptkategorien *Messauftrag*, *Auftragsmetadaten*, *Bauteilspezifische Zusatzinformationen* und *Prüfling* und ihre Subkategorien, die Struktur des Baums weicht jedoch von der Struktur der Haupt- und Subkategorien des VDA FDX Datenmodells ab (vergleiche hierzu ***Abbildung 8-4***). Beispielsweise ist das Applikationselement `TplUnitUnderTestComp` ‘MeasurementOrder‘ unter dem Applikationselement `TplUnitUnderTestComp` ‘OrderMetaData‘ organisiert, während das VDA FDX Datenmodell eine eigene Hauptkategorie *Messauftrag* besitzt.

Für die so gefundene Instanz der Klasse `CatUnitUnderTestComp` wird ein Applikationselement der Klasse `UnitUnderTest` angelegt. Für jede so gefundene Instanz der Klasse `CatUnitUnderTestComp` wird ein Applikationselement mit der entsprechenden Klasse angelegt und als Komponente dieser Instanz von `UnitUnderTest` zugeordnet.

Am Beispiel des Gummilagers wäre es für ein Applikationselement `MeasurementOrder` als Komponente von `UnitUnderTest` erforderlich, dass ausgehendend vom Applikationselement `TplUnitUnderTestRoot` der Baum der Applikationselemente `TplUnitUnderTestComp` eines enthält, dass auf ein bestimmtes Applikationselement `CatUnitUnderTestComp` verweist. Dieses Applikationselement `CatUnitUnderTestComp` entspräche der Klasse `MeasurementOrder` als Komponente von `UnitUnderTest`.

Diese Beschreibung gilt entsprechend für die Klassen `TplTestSequenceRoot`, `TplTestSequenceComp` und `CatTestSequenceComp` für die in den Testschritt aufzunehmenden Komponenten der Klasse `TestSequence` beziehungsweise für die Klassen `TplTestEquipmentRoot`, `TplTestEquipmentComp` und `CatTestEquipmentComp` für die in den Testschritt aufzunehmenden Komponenten der Klasse `TestEquipment`.

Für die so gefundene Instanz der Klasse `CatTestSequenceComp` wird ein Applikationselement der Klasse `TestSequence` beziehungsweise für die Instanz der Klasse `CatTestEquipmentComp` wird ein Applikationselement der Klasse `TestEquipment` angelegt.

Im Beispiel des Gummilagers würde ein Applikationselement `TplTestSequenceComp` zu einem Applikationselement `Measurement` und ein Applikationselement `TplTestEquipmentComp` zu einem Applikationselement `DataOrigin` führen.

Ebenso können die Templates `TplTestSequenceComp` so angeordnet werden, dass sie die Hauptkategorien *Datensatzmetadaten*, *Messgerätebetriebsgrößen* und *Abgeleitete Kenngrößen* und ihre Subkategorien hierarchisch organisieren, ebenso die Templates `TplTestEquipmentComp` mit den Hauptkategorien *Messgerätesetup* und *Funktionsdaten* und ihren Subkategorien.

Für die beiden dynamischen Messungen im Szenario des Testvorhabens nach **Fehler! Verweisquelle onnte nicht gefunden werden.** verweist das Applikationselement `TplTestStep` 'Dynamisch 1' auf das Applikationselement `TplTestSequenceRoot` 'HarmonicVarFrequency' und für die quasistatische Messung verweist das Applikationselement `TplTestStep` 'Quasistatisch' auf das Applikationselement `TplTestSequenceRoot` 'OscillatingExcitation'. Diese Applikationselemente `TplTestSequenceRoot` spannen jeweils einen Baum von Applikationselementen `TplTestSequenceComp` auf.

Hinweis: Auch hier beinhaltet der Baum der Applikationselemente `TplTestSequenceComp` zwar alle Hauptkategorien *Datensatzmetadaten*, *Messgerätebetriebsgrößen* und *Abgeleitete Kenngrößen* und ihre Subkategorien. Die Struktur des Baums weicht jedoch von der Struktur der Haupt- und Subkategorien des VDA FDX Datenmodells ab (vergleiche hierzu ***Abbildung 8-7***). Beispielsweise ist das Applikationselement `TplTestEquipmentComp` 'TP_HarmonicVarFrequency' unter dem Applikationselement `TplUnitUnderTestComp` 'Measurement' organisiert, während das VDA FDX Datenmodell die Subkategorie *Messprogramm Oszillierende Anregung* besitzt.

Für die beiden dynamischen Messungen im Szenario des Testvorhabens nach **Fehler! Verweisquelle onnte nicht gefunden werden.** verweist das Applikationselement `TplTestStep` 'Dynamisch 1' auf das Applikationselement `TplTestEquipmentRoot` 'CurveDisplacementFrequencyTranslational' und für die quasistatische Messung verweist das Applikationselement `TplTestStep` 'Quasistatisch' auf das Applikationselement `TplTestSequenceRoot` 'ForceDisplacement'. Diese Applikationselemente `TplTestEquipmentRoot` spannen jeweils einen Baum von Applikationselementen `TplTestEquipmentComp` auf.

Hinweis: Auch hier beinhaltet der Baum der Applikationselemente `TplTestEquipmentComp` zwar alle Hauptkategorien *Messgerätesetup* und *Funktionsdaten* und ihre Subkategorien, die Struktur des Baums weicht jedoch von der Struktur der Haupt- und Subkategorien des VDA FDX Datenmodells ab (vergleiche hierzu ***Abbildung 8-10***). Beispielsweise ist das Applikationselement `TplTestEquipmentComp` 'DataOrigin' unmittelbar unter dem Applikationselement `TplTestEquipmentRoot` 'CurveDisplacementFrequencyTranslational' organisiert, während das VDA FDX Datenmodell die Subkategorie *Datenursprung* unter der Hauptkategorie *Messgerätesetup* organisiert.

Nicht benötigte Applikationselemente wie beispielsweise die Komponente `PreTest` des Applikationselements `UnitUnderTest`, die Komponenten `Prelaod` und `PreCondition` des Applikationselements `TestSequence`, und die Komponenten `Simulation` und `Estimation` des Applikationselements `TestEquipment` werden im Szenario nach **Fehler! Verweisquelle konnte nicht efunden werden.** durch leere Felder abgebildet.

Bildet das Applikationselement `TplTestEquipmentComp` das Template eines Applikationselements zu den Subkategorien *Statische Messung* und *Dynamische Messung* aus der Hauptkategorie *Funktionsdaten*, dann müssen weitere Applikationselemente die Messwerte der Messkurve beschreiben. Im Beispiel des Gummilagers gehört das durch `CatTestEquipmentComp` bestimmte Applikationselement `MeasurementSensors` zu einer der Subkategorien *Statische Messung* und *Dynamische Messung*.

Um die Messwerte der Messkurve zu bestimmen verweist eine Instanz der Klasse `TplTestEquipmentComp` auf 0..* Instanzen der Klasse `TplSensor`. Die Klasse `TplSensor` repräsentiert die Auswahl der Klasse des Applikationselements, das einen Messwert beschreiben soll. Hierzu verweist eine Instanz der Klasse `TplSensor` auf eine Instanz der Klasse `CatSensor`. Die

Klasse `CatSensor` repräsentiert den Katalog der Klassen von Applikationselementen, mit denen Messwerte beschrieben werden.

Die aus `CatSensor` hervorgegangenen Applikationselemente sind Komponenten des Applikationselements, das aus dem Applikationselement `CatTestEquipmentComp` hervorgegangen ist. In diesem Sinne sind die aus `TplSensor` hervorgegangenen Applikationselemente die Unterkomponenten des Applikationselements `TestEquipment`.

Für jede so gefundene Instanz der Klasse `TplSensor` wird eine Instanz der Klasse entsprechend `CatSensor` angelegt und als Komponente der `CatTestEquipmentComp` entsprechenden Instanz zugeordnet. Im Beispiel des Gummilagers sind es der Anzahl der Anwendungselemente `TplSensor` entsprechend viele Applikationselemente `Sensors` als Komponenten des Applikationselements `MeasurementSensors`.

Die Klassen `CatUnitUnderTestAttr`, `CatTestSequenceAttr` und `CatTestEquipmentAttr` sowie `CatSensorAttr` (***Abbildung 8-1***) repräsentieren Kataloge von Attributen, die überhaupt in den Klassen der Komponenten der Klassen `UnitUnderTest`, `TestSequence` und `TestEquipment` sowie in den Klassen der Unterkomponenten der Klasse `TestEquipment` verfügbar sind.

Die Klassen `TplUnitUnderTestAttr`, `TplTestSequenceAttr` und `TplTestEquipmentAttr` sowie `TplSensorAttr` (***Abbildung 8-1***) repräsentieren die Templates, mit denen die in die Komponenten der Klassen `UnitUnderTest`, `TestSequence` und `TestEquipment` sowie in die Unterkomponenten der Klasse `TestEquipment` aufzunehmenden Attribute auswählt werden.

Für diese Auswahl umfasst das Anwendungselement `TplUnitUnderTestComp` 0..* Anwendungselemente `TplUnitUnderTestAttr`, das Anwendungselement `TplTestEquipmentComp` 0..* umfasst Anwendungselemente `TplTestEquipmentAttr` und das Anwendungselement `TplTestEquipmentComp` umfasst 0..* Anwendungselemente `TplTestEquipmentAttr`.

Um ein Attribut zu bestimmen verweist das Anwendungselement `TplUnitUnderTestAttr` auf 0..1 bestimmtes Anwendungselement `CatTestEquipmentComp`, das Anwendungselement `TplUnitTestEquipmentAttr` verweist auf 0..1 bestimmtes Anwendungselement `CatTestEquipmentAttr`, und das Anwendungselement `TplTestSequenceAttr` verweist auf 0..1 bestimmtes Anwendungselement `CatTestSequenceAttr` sowie das Anwendungselement `TplSensorAttr` verweist auf 0..1 bestimmtes Anwendungselement `CatSensorAttr`.

Mit den Klassen `ValueList` und `ValueListValue` vom ASAM ODS Basistyp `AoParameterSet` und `AoParameter` (***Abbildung 8-1***) kann der Wertevorrat eines Attributs auf bestimmte Aufzählungstypen eingeschränkt werden. Dabei repräsentiert die Klasse `ValueList` eine Aufzählung und die Klasse `ValueListValue` einen Aufzählungstyp der Aufzählung. Hierzu umfasst das Applikationselement `ValueList` 0..* Applikationselemente `ValueListValue`.

Zur Angabe der Aufzählung verweisen die Applikationselemente `CatUnitUnderTestAttr`, `CatTestSequenceAttr`, `CatTestEquipmentAttr` und `CatSensorAttr` auf 0..1 bestimmtes Anwendungselement `ValueList`.

Abbildung des VDA FDX Datenmodells auf das VDA FDX Applikationsmodell

Das folgende Objektdiagramm in ***Abbildung 8-13*** vergleicht das VDA FDX Datenmodell mit dem VDA FDX Applikationsmodell anhand das modellübergreifenden Szenarios einer typischen Bestellung und Datenlieferung (siehe ***Abbildung 3-10*** und ***Abbildung 3-11***):

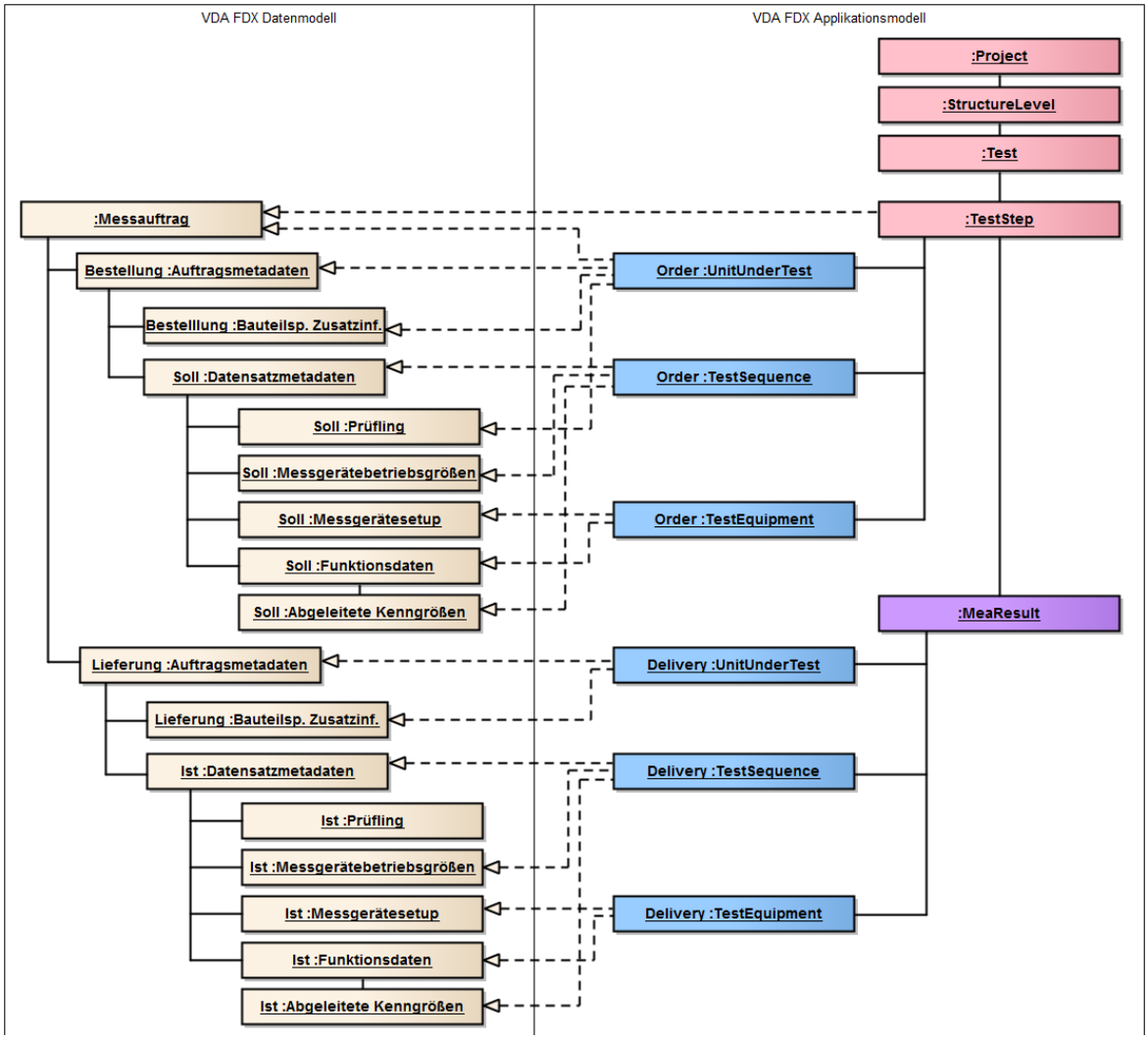


Abbildung 8-13 Szenario für die Abbildung des VDA FDX Datenmodells auf das VDA FDX Applikationsmodell

Der Name eines Objekts setzt sich einem Präfix und einem davon mit einem Doppelpunkt getrennten Postfix zusammen.

Das Präfix ordnet ein Objekt den Daten zur Bestellung oder den Daten zur Lieferung zu (in der Abbildung ´Bestellung´ und ´Lieferung´ für das Applikationsmodell des Fachkonzepts beziehungsweise ´Order´ und ´Delivery´ für das openMDM Applikationsmodell). Das Präfix ordnet ein Objekt den Soll- oder Ist-Funktionsdaten zu (in der Abbildung ´Soll´ / ´Ist´ für das Applikationsmodell des Fachkonzepts).

Der Postfix gibt die Klasse des Applikationselements an.

In diesem Szenario gibt es einen *Messauftrag*, der durch die mit ´Order´ gekennzeichneten Applikationselemente `TestStep` und `UnitUnderTest` realisiert wird. Die Daten der Bestellung verteilen sich über die mit ´Order´ gekennzeichneten Applikationselemente `UnitUnderTest`, `TestSequence` und `TestEquipment` und ihre Komponenten (in der Abbildung nicht dargestellt).

Zu diesem *Messauftrag* gibt es eine Datenlieferung, die durch ein Applikationselement `MeaResult` realisiert wird. Die Daten der Datenlieferung verteilen sich über die mit ´Delivery´ gekennzeichneten Applikationselemente `UnitUnderTest`, `TestSequence` und `TestEquipment` und ihre Komponenten.

Das Szenario eines spezifischen openMDM Applikationsmodells am Beispiel Gummilager

Das folgende Objektdiagramm in *Fehler! Verweisquelle konnte nicht gefunden werden.* zeigt ein ypisches Szenarios einer Bestellung und Datenlieferung am Beispiel Gummilager:

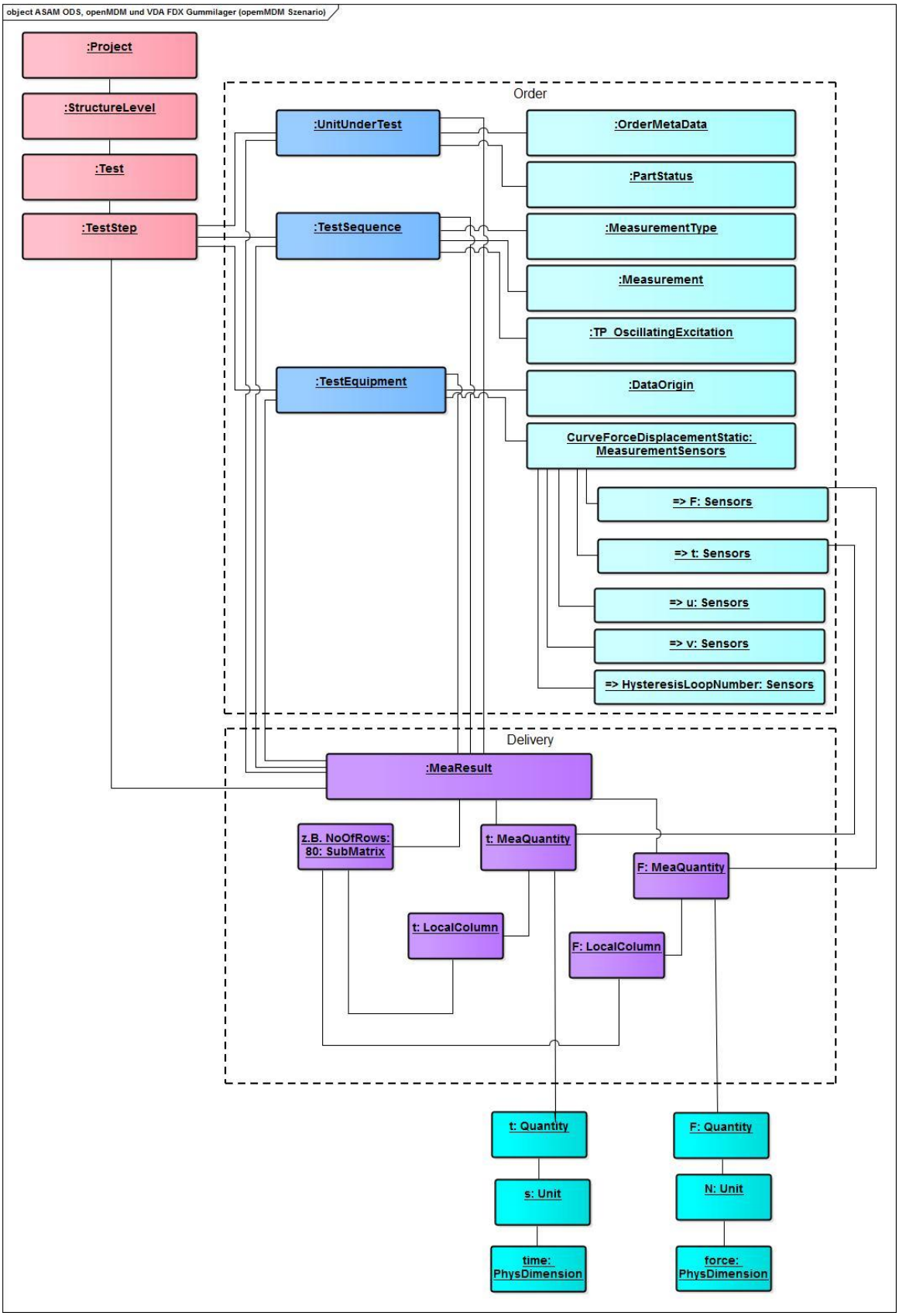


Abbildung 8-14 Szenario eines spezifischen openMDM Applikationsmodells am Beispiel Gummilager

Der Name eines Objekts setzt sich aus einem gegebenenfalls vorhandenen Präfix und einem davon mit einem Doppelpunkt getrennten Postfix zusammen.

Das Präfix prägt ein Applikationselement für einen bestimmten Anwendungszweck aus, falls die Klasse des Applikationselements an sich den Anwendungszweck nicht ausreichend beschreibt.

Der Postfix gibt die Klasse des Applikationselements an.

Das Applikationselement `TP_OscillatingExcitation` steht für die Subkategorie *Messprogramm Oszillierende Anregung*.

Das mit dem Präfix ‚CurveForceDisplacementStatic' ausgeprägte Applikationselement `MeasurementSensors` steht für das *Messprogramm Statische Kraft-Weg Kennlinie*. Die mit den Präfixen ‚=> t', ‚=> F', ‚=> u', ‚=> v' und ‚=> HysteresisLoopNumber' ausgeprägten Applikationselemente `Sensors` stehen für die einzelnen Messwerte.

Das Messergebnis wird in einem Applikationselement `MeaResult` zusammengefasst. Jeder Messwert wird jeweils über ein Applikationselement `MeaQuantity`, `SubMatrix` und `LocalColumn` erfasst.

Weil der Messauftrag durch den Datenlieferanten gemäß den Daten zur Bestellung durchgeführt wurde, das heißt es gibt zu den Hauptkategorien *Auftragsmetadaten, bauteilspezifische Zusatzinformationen, Datensatzmetadaten, Prüfling, Messgerätesetup, Messgerätebetriebsgrößen, Funktionsdaten und abgeleitete Kenngrößen* keine abweichenden Daten zur Lieferung, verweist das Applikationselement `MeaResult` auf die Applikationselemente `UnitUnderTest`, `TestSequence` und `TestEquipment` aus der Bestellung.

8.7 Anhang G: Präsentation der modellgetriebenen Aspekte des openMDM4 Applikationsmodells in ATFX

Dies ist eine Ergänzung zu Abschnitt „Zusammenhang ASAM ODS Basismodell, openMDM Applikationsmodell und VDA FDX Applikationsmodell am Beispiel Gummilager“ der VDA_Empfehlung_5550.

Präsentation der modellgetriebenen Aspekte des openMDM4 Applikationsmodells in ATFX

Das folgende gemischte Klassen-/Objektdiagramm in *Fehler! Verweisquelle konnte nicht gefunden erden.* zeigt wie die modellgetriebenen Aspekte des openMDM4 Applikationsmodells in das ASAM ODS Basismodell eingebettet werden:

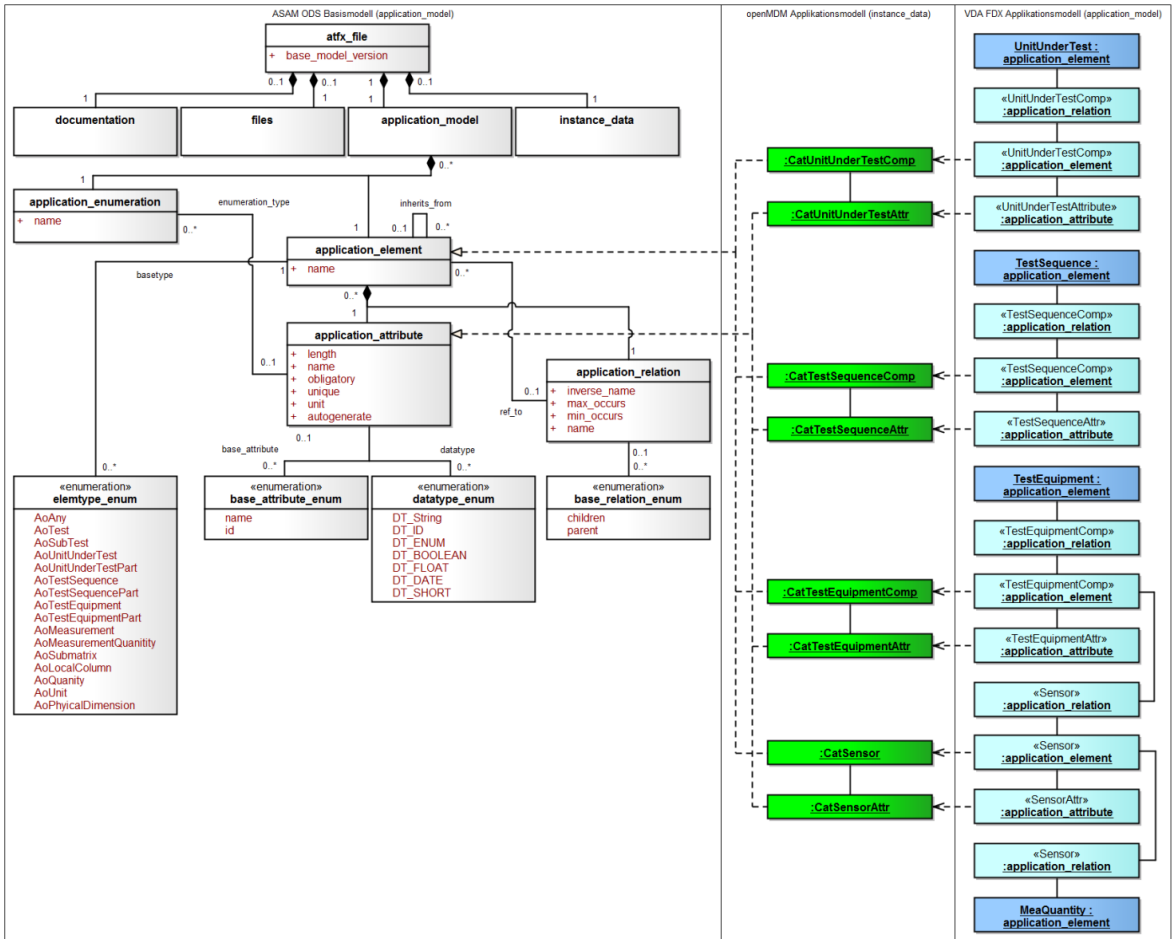


Abbildung 8-15 Die modellgetriebenen Aspekte des openMDM Applikationsmodells

Die Klassen `atfx_file`, `documentation`, `files`, `application_model` und `instance_data` repräsentieren die Hauptbestandteile einer ATFX-Datei (siehe die Schwimmbahn „ASAM ODS Basismodell (application_model)“ in *Fehler! Verweisquelle konnte nicht gefunden werden.*).

Die Sektion `application_model` beschreibt mit den Klassen `application_enumeration` und insbesondere `application_element` alle in der Sektion `instance_data` erlaubten Applikationselemente.

Die Klasse `application_model` repräsentiert dabei die Klasse eines Applikationselements, indem es ihre Attribute durch die Klasse `application_attribute` und die Beteiligung an einer Assoziation mit einem anderen Applikationselement durch die Klasse `application_relation` angibt. Die Assoziation `application_relation.ref_to` gibt das andere Applikationselement am Ende der Assoziation an. Eine nur in einer Richtung navigierbare Assoziation wird durch eine Instanz von `application_relation` angegeben; eine in beiden Richtungen navigierbare Assoziation zwischen zwei Applikationselementen wird durch zwei Instanzen von `application_relation` angegeben.

Mit den weiteren Klassen `elemtype_enum`, `base_attribute_enum`, `datatype_enum` und `base_relation_enum` werden Eigenschaften der Applikationselemente beschreiben, das heißt welchem ASAM ODS Basistyp ein Applikationselement angehört, welches ASAM ODS Basisattribut das Attribut eines Applikationselements realisiert, welchen Datentyp das Attribut eines Applikationselementes besitzt und welche Basisrelation die Relation eines Applikationselements realisiert.

Die Klassen `CatUnitUnderTestComp`, `CatTestSequenceComp`, `CatTestEquipmentComp` und `CatSensor` repräsentieren wie oben beschrieben die Kataloge der Klassen von Applikationselementen, die als Komponenten der Klasse `UnitUnderTest`, `TestSequence` und `TestEquipment` und Unterkomponenten der Klasse `TestEquipment` verfügbar sind.

Die Klassen `CatUnitUnderTestAttr`, `CatTestSequenceAttr`, `CatTestEquipmentAttr` und `CatSensorAttr` repräsentieren wie oben beschrieben die Kataloge der Attribute dieser Komponenten.

Die Klassen `CatUnitUnderTestComp`, `CatTestSequenceComp`, `CatTestEquipmentComp` und `CatSensor` sowie `CatUnitUnderTestAttr`, `CatTestSequenceAttr`, `CatTestEquipmentAttr` und `CatSensorAttr` selbst sind als `application_element` in der Sektion `application_model` modelliert.

Die Instanzen der Klassen `CatUnitUnderTestComp`, `CatTestSequenceComp`, `CatTestEquipmentComp` und `CatSensor` sowie `CatUnitUnderTestAttr`, `CatTestSequenceAttr`, `CatTestEquipmentAttr` und `CatSensorAttr` in der Sektion `instance_data` sind nun die materiellen Templates, zu denen in der Sektion `application_model` entsprechende Instanzen der Kassen `application_element`, `application_attribute` und `application_relation` anlegt werden, welche die Klassen der Komponenten der Klassen `UnitUnderTest`, `TestSequence` und `TestEquipment` beschreiben (siehe die Schwimmbahn „openMDM Applikationsmodell (instance_data)“ und die gestrichelten Pfeile in *Fehler! Verweisquelle onnte nicht gefunden werden.*).

Hierzu wird für jede Instanz der Klasse `CatUnitUnderTestComp`, `CatTestSequenceComp`, `CatTestEquipmentComp` und `CatSensor` in der Sektion `application_model` eine entsprechende Instanz der Kasse `application_element` angelegt (siehe die Schwimmbahn „VDA FDX Applikationsmodell (application_model)“ und die gestrichelten Pfeile in *Fehler! Verweisquelle onnte nicht gefunden werden.*).

Ebenso wird für jede Instanz der Klasse `CatUnitUnderTestAttr`, `CatTestSequenceAttr`, `CatTestEquipmentAttr` und `CatSensorAttr` in der Sektion `application_model` eine entsprechende Instanz der Kasse `application_attribute` angelegt und der entsprechenden Instanz der Klasse `CatUnitUnderTestComp`, `CatTestSequenceComp`, `CatTestEquipmentComp` und `CatSensor` zugeordnet (siehe die Schwimmbahn „VDA FDX Applikationsmodell (application_model)“ und die gestrichelten Pfeile in *Fehler! Verweisquelle konnte icht gefunden werden.*).

Um auszudrücken, dass ein neues Applikationselement, das aus einer Instanz von `CatUnitUnderTestAttr`, `CatTestSequenceAttr` oder `CatTestEquipmentAttr` hervorgegangen ist, eine Komponente der Klasse `UnitUnderTest`, `TestSequence` oder `TestEquipment` ist, wird eine entsprechende Instanz der Kasse `application_relation` angelegt.

Um auszudrücken, dass ein neues Applikationselement, das aus einer Instanz von `CatSensor` hervorgegangen ist, eine Komponente der aus einem `CatTestEquipmentComp` hervorgegangenen Klasse ist, werden für eine in beiden Richtungen navigierbare Assoziation zwei Instanzen der Kasse `application_relation` angelegt.

Zusätzlich werden zwei Instanzen der Kasse `application_relation` angelegt, welche die in beiden Richtungen navigierbare Assoziation zwischen der Klasse `MeaQuantity` und der aus einer Instanz von `CatSensor` hervorgegangen Klasse verkörpert.

8.8 Anhang H: Das XML Schema des ASAM ATFX-Datenaustauschformats

Das XML Schema des ASAM ATFX-Datenaustauschformats wird von der ASAM definiert (ASAM ODS - Standards) und ist in den 4 XML Schema-Definitionen Schema.xsd, ASAM_HDTypes.xsd, HelperSchema.xsd und ODSBaseModelSpecSchema.xsd festgelegt.

Diese definieren insbesondere das XML Schema für den Rahmen des ASAM ATFX-Datenaustauschformats und insbesondere das XML Schema der Applikationselemente der Sektion `application_model`. Diese definieren aber nicht das XML Schema der Instanzen der Applikationselemente in der Sektion `instance_data`, weil die Applikationselemente erst im Rahmen der Ausprägung des ASAM ODS Basismodells entstehen.

Hierzu wird der ASAM ODS Checker beschrieben (Schema Validation of ATFX-Files According to ASAM ODS V5.3.0), mit dem die in der Sektion `application_model` beschriebenen `application_element` (siehe den vorangegangenen Abschnitt) in die entsprechenden XML Schema Definitionen (xsd) der Instanzen der Applikationselemente der Sektion `instance_data` übersetzt werden können.

Ein `application_attribute` zum `base_attribute_enum` ‚id' und dem `application_attribute.name` ‚Id' wird als `xsd:element` mit dem `type` `xsd:long` übersetzt. Eine `application_relation` als das Ende einer Assoziation wird als `xsd:element` mit dem `type` `t_reference` übersetzt. Der `xsd:simpleType` `t_reference` ist eine `xsd:union` von `xsd:integer` und `xsd:string`, das heißt in einem solchen Attribut werden alle `id` der Applikationselemente am gegenüberliegenden Ende der Assoziation in einer durch Leerzeichen getrennten Liste von Integer aufgezählt.

Der folgende Ausschnitt einer ATFX-Austauschdatei in ***Abbildung 8-16*** zeigt am Beispiel des VDA FDX Applikationsmodells zum Gummilager ausgewählte `application_attribute` und `application_relation` des `application_element` für das Applikationselement `UnitUnderTest` in der Sektion `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>
```

Abbildung 8-16: Ausschnitt der Sektion application_element einer ATFX-Austauschdatei am Beispiel des VDA FDX Applikationsmodells zum Gummilager für das Applikationselement UnitUnderTest

Der folgende Ausschnitt einer ATFX-Austauschdatei in *Fehler! Verweisquelle konnte nicht gefunden erden.* zeigt am Beispiel des VDA FDX Applikationsmodells zum Gummilager ausgewählte `application_attribute` und `application_relation` des `application_element` für das Applikationselement `MeasurementOrder` in der Sektion `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>
```

Abbildung 8-17 Ausschnitt der Sektion application_element einer ATFX-Austauschdatei am Beispiel des VDA FDX Applikationsmodells zum Gummilager für das Applikationselement `MeasurementOrder`

Der folgende Ausschnitt des XML Schema einer ATFX-Austauschdatei in ***Abbildung 8-18*** zeigt das Schemaelement zum Applikationselement `UnitUnderTest` in der Sektion `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>
```

Abbildung 8-18 Ausschnitt der Sektion instance_data einer ATFX-Austauschdatei am Beispiel des VDA FDX Applikationsmodells zum Gummilager für das Applikationselement `UnitUnderTest`

Der folgende Ausschnitt des XML Schema einer ATFX-Austauschdatei in ***Abbildung 8-19*** zeigt das Schemaelement zum Applikationselement `MesurementOrder` in der Sektion `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>
```

Abbildung 8-19 Ausschnitt der Sektion instance_data einer ATFX-Austauschdatei am Beispiel des VDA FDX Applikationsmodells zum Gummilager für das Applikationselement `MeasurementOrder`

Der folgende Ausschnitt einer ATFX-Austauschdatei in ***Abbildung 8-20*** zeigt am Beispiel eines anonymisierten Messauftrags des VDA FDX Applikationsmodells zum Gummilager ein Applikationselement `UnitUnderTest` in der Sektion `instance_data`:

```
  <instance_data>
    <UnitUnderTest>
      <Id>10004</Id>
      <Name>ABCDEFGHI</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>
```

Abbildung 8-20 Ausschnitt der Sektion instance_data einer ATFX-Austauschdatei am Beispiel des VDA FDX Applikationsmodells zum Gummilager für das Applikationselement `UnitUnderTest`

Der folgende Ausschnitt einer ATFX-Austauschdatei in ***Abbildung*** 8-21**Fehler! Verweisquelle konnte nicht gefunden werden.** zeigt am Beispiel eines anonymisierten Messauftrags des VDA FDX Applikationsmodells zum Gummilager ein Applikationselement `MeasurementOrder` in der Sektion `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>
```

Abbildung 8-21 Ausschnitt der Sektion `instance_data` einer ATFX-Austauschdatei am Beispiel des VDA FDX Applikationsmodells zum Gummilager für das Applikationselement `MeasurementOrder`

XML Schema freie Bearbeitung des ASAM ATFX-Datenaustauschformats

Zur XML Schema freien Bearbeitung des ASAM ATFX-Datenaustauschformats stehen entsprechende Java-Klassen zur Verfügung (openATFX download | SourceForge.net), mit denen beliebige ASAM ATFX-Dateien bearbeitet werden, also insbesondere unabhängig vom openMDM Applikationsmodell. Allerdings muss hierzu ein IDL interface definition file vorliegen (ASAM ODS - Standards), das in ein Java-Archiv zu übersetzt werden muss.

8.9 Anhang I: Konventionen zur Interpretation der UML-Diagramme und der zugehörigen textuellen Beschreibung

Dies ist eine Ergänzung zu Abschnitt „Erweiterungen zum openMDM Applikationsmodell “ und zu Abschnitt „Zusammenhang ASAM ODS Basismodell, openMDM Applikationsmodell und VDA FDX Applikationsmodell am Beispiel Gummilager“ der VDA_Empfehlung_5550.

Zur Interpretation der UML-Diagramme und der zugehörigen textuellen Beschreibung werden folgende Konventionen getroffen:

- In den Klassendiagrammen und Objektdiagrammen werden die Klassen und Objekte aus dem VDA FDX Datenmodell, dem ASAM ODS Basismodell, dem VDA FDX Applikationsmodell und dem openMDM Applikationsmodell durch 2 und gegebenenfalls 3 vertikale Schwimmbahnen unterschieden, welche das jeweilige Modell verkörpern.
- Zur leichteren Lesbarkeit sind die Klassen und Objekte mit Farben hinterlegt: Beige für das VDA FDX Datenmodell, grau für das ASAM ODS Basismodell, hellblau für das VDA FDX Applikationsmodell, sowie , rot, blau, violett, türkis, gelb und grün für das openMDM Applikationsmodell.
- Eine Vererbung zwischen zwei Klassen wird durch Linien mit einem nicht ausgefüllten Dreieck am Ende dargestellt. Die Richtung des Pfeils zeigt auf die allgemeinere Klasse.
- Eine Assoziation zwischen zwei Klassen wird durch eine durchgehende Linie dargestellt.

  - Eine Assoziation kann mit weiteren Informationen wie Rollen, Multiplizitäten und weiteren Zeichen gekennzeichnet sein.
  - Zur Vereinfachung sind die Enden der Assoziationen ausgeblendet, weil diese meistens den Namen der an der Assoziation beteiligten Klassen entsprechen (im nachfolgenden Beispiel würden die Enden „Structure Level“ und „Project“ lauten).
  - Eine Ausnahme bilden Situationen, bei denen zwei Klassen mit mehr als einer Assoziation in Beziehung stehen oder reflexive Assoziationen, bei denen die gleiche Klasse miteinander in Beziehung steht. In diesen Situationen müssen die beiden Enden der Assoziation unterschieden werden und sie werden deshalb angezeigt.
  - Die Multiplizitäten einer Assoziation werden durch ganze Zahlen bzw. Intervalle gekennzeichnet (n1 bzw. n1..n2 mit n1, n2 ganze Zahl oder * für beliebig viele). Die Multiplizität an einem Ende der Assoziation gibt für eine Instanz an diesem Ende an, wie viele Instanzen ihr auf der gegenüberliegenden Seite ihr zugeordnet sein können.
    Achtung: Diese Konvention hier ist nicht konform zur UML. Gegenüber der UML sind hier die Multiplizitäten „spiegelbildlich“ zu lesen.
- Die Multiplizitäten einer Assoziation können unterschiedliche Konstellationen der beteiligten Objekte charakterisieren:
  - Multiplizitäten in der Kombination 0..* und 1 finden sich nach einer Lesart der UML bei einer Konstellation, die eine Komposition eines Ganzen und seiner Teile beschreibt. Mit anderen Worten, ein untergeordnetes Objekt kann nur im Kontext eines anderen übergeordneten Objekts existieren.
    Um dies besonders hervorzuheben wird hier das mit der Multiplizität 0..* gekennzeichnete Ende der Assoziation zusätzlich mit einer ausgefüllten Raute versehen, und im Text wird darauf mit dem Begriff „umfasst“ verwiesen.
    Beispiel: Die in der Kombination 0..* und 1 vorliegenden Multiplizitäten der Assoziation zwischen der Klasse `Project` und der Klasse `StructureLevel` bedeuten also, dass 1 Instanz der Klasse `Project` 0, 1, oder * Instanzen der Klasse `StructureLevel` zugeordnet sein können, und dass 1 Instanz der Klasse `StructureLevel` genau 1 Instanz der Klasse `Project` zugeordnet ist. Im Text würde es also heißen dass eine Instanz der Klasse `Project` 0..* Instanzen der Klasse `StructureLevel` umfasst.
    Handelt es sich um ein untergeordnetes Applikationselement ist es im Sprachgebrauch dieses Dokuments eine Komponente des übergeordneten Applikationselements.
    Eine Spielart dieser Konstellation sind Assoziationen mit Multiplizitäten in der Kombination 0..* und 0..1 zwischen einer gegebenen Klasse mehreren anderen Klassen, wobei ein Objekt der gegebenen Klasse genau einem Objekt aus den anderen Klassen untergeordnet sein muss. Auch hier werden die mit der Multiplizität 0..* gekennzeichneten Enden der Assoziationen zusätzlich mit einer ausgefüllten Raute versehen. Ein Beispiel dieser Konstellation findet sich in Abschnitt 4.3.1 zur Steuerung der Sichtbarkeit von Komponenten und Attributen. Ein Applikationselement `ComponentVisibility` muss genau einem der Applikationselemente `TplUnitUnderTestComp`, `TplTestSequenceComp` oder `TplTestEquipmentComp` untergeordnet sein. Entsprechend muss ein `AttributeVisibility` genau einem der Applikationselemente `TplUnitUnderTestAttr`, `TplTestSequenceAttr` oder `TplTestEquipmentAttr` untergeordnet sein.
  - Dem gegenüber gibt es die Konstellation, in der Multiplizitäten in der Kombination 0..* und 0..1 vorliegen, wenn zwei voneinander unabhängige Objekte mit einer Assoziation in Beziehung stehen. Eine solche Konstellation wird in den UML-Klassendiagrammen nicht besonders hervorgehoben, aber im Text wird darauf mit den Begriffen „verweist“

und „verwendet“ verwiesen.
Beispiel: Die in der Kombination 0..1 und 0..* vorliegenden Multiplizitäten der Assoziation zwischen der Klasse `TestStep` und der Klasse `UnitUnderTest` bedeuten also, dass 1 Instanz der Klasse `TestStep` 0 oder 1 Instanzen der Klasse `UnitUndertest` zugeordnet sein kann und dass 1 Instanz der Klasse `UnitUnderTest` 0, 1, oder * Instanzen der Klasse `TestStep` zugeordnet sein können. Im Text würde es also heißen, dass eine Instanz der Klasse `TestStep` auf 0..1 Instanzen der Klasse `UnitUnderTest` verweist und dass eine Instanz der Klasse `UnitUnderTest` von 0..* Instanzen der Klasse `TestStep` verwendet werden kann.

- Eine weitere Konstellation ist die sogenannte attributierte N:M-Beziehung, die aus zwei Kompositionen besteht, die drei Klassen miteinander verbinden. Die Multiplizitäten der ersten und zweiten Klasse liegen in der Kombination 0..* und 1 und die Multiplizitäten der zweiten und dritten Klasse liegen in der Kombination 1 und 0..* vor. Ein untergeordnetes Objekt der zweiten und damit mittleren Klasse kann also nur im Kontext zweier übergeordnete Objekte der ersten und dritten Klasse existieren. Ein Beispiel dieser Konstellation findet sich in Anhang F: Die modellgetriebenen Aspekte des openMDM Applikationsmodells bei der Auswahl der Testschritte zu einem Testvorhaben mittels der Applikationselemente `TplTest`, `TplTestUsage`, `TplTestStep`. Das untergeordnete Objekt ist das Applikationselement `TplTestUsage` und die übergeordneten Objekte sind die Applikationselemente `TplTest` und `TplTestStep`.

- Die Einfärbung der Klassen in den folgenden Bildern entspricht der openMDM-Übersicht im Anhang B.
- Auf die Darstellung der Organisation der Klassen in die Pakete `Administration`, `Descriptive Data` und `Measurement`, wie es in den Übersichten von ASAM ODS (siehe Abbildung 4-3) und des openMDM Applikationsmodells (Anhang B) dargestellt ist, wird aus Gründen der Übersichtlichkeit verzichtet.
- Unter dem Begriff Applikationselement werden hier Instanzen der entsprechenden Klasse verstanden.