FACE™ 3.0, 3.1, and 3.2 Guidance

Specific to

MagicDraw and Cameo

Revision B

2/15/2024

Table of Contents

Figures

Tables

Colophon

Record of Changes

Revision	Date	Number of Figure, Table, or Paragraph	A* M D	Title or Brief Description
New	1/13/2023			Initial Release
A	1/19/2024		A	Approved for public release: distribution unlimited
B	2/15/2024		A	Army PEO Aviation Control No: 004-23 Distribution Statement A - "Approved for public release: distribution unlimited"

*A-Added, M-Modified, D-Deleted

1 Introduction

There are many sources of information about the Future Airborne Capability Environment (FACE™), which include the FACE Technical Standard as well as guidance on modeling per the FACE Standard. This document does not reproduce that information, rather it addresses the specifics applicable to producing FACE models with MagicDraw (MD)/Cameo.

Notes:

1. This document only addresses modeling per FACE 3.0, 3.1 and 3.2. The earlier versions of FACE (e.g. 2.0, 2.1, 2.1.1) are not addressed by this document.
2. This document is applicable to both MagicDraw and Cameo. These two modeling tools are very similar, and thus MagicDraw is used to indicate both tools in the remainder of this document.
3. Cameo 2021 was used for the modeling examples/figures.

1.1 Reference Documents

The reference documentation is listed in **Error! Reference source not found.** This is typically used as a s tarting point to become familiar with FACE modeling. Reference [1] provides a table of contents and links to the published FACE documentation. This provides a comprehensive view of the available documentation. It is important to first become familiar with the Technical Standards ([2], [3], and [4]) and then use the guidance documents ([5], [6], [7]) to learn how to apply the Standards to modeling your system. Currently, there is not a guidance document for FACE 3.1, but because FACE 3.0, 3.1, and 3.2 are very similar, most of the FACE 3.0 guidance information applies to FACE 3.1 and 3.2.

Table 1 - Reference Documents

Ref. ID	Document	Date/Revision	Links/File-Name where Applicable
[1]	Document & Tools Published Documents	N/A	https://www.opengroup.org/face/docsandtools
[2]	FACE™ Technical Standard, Edition 3.0	Release 2017-814	https://publications.opengroup.org/c17c
[3]	FACE™ Technical Standard, Edition 3.1	Published by The Open Group, July 2020	https://publications.opengroup.org/c207
[4]	Open Universal Domain Description Language (Open UDDL), Edition 1.0	Jan. 2019	https://publications.opengroup.org/s190
[5]	Reference Implementation Guide for FACE™ Technical Standard, Edition 3.0, Volume 1, 2, and 3		https://publications.opengroup.org/guides/face/g209

[6]	Reference Implementation Guide for FACE™ Technical Standard, Edition 3.1, Volume 1, 2, and 3		Currently being developed.
[7]	MagicDraw MTI Instructions for Allowing Non-FACE(tm) Elements to have Relationships to FACE Elements.pdf	9/23/2022	Delivered with this document .\docs
[8]	Shared Data Model Governance Plan, Edition 3.1	February 2020	https://www.opengroup.org/face/docsandtools 1. Select "FACE Technical Standard Edition 3.1" 2. Select "Shared Data Model Governance Plat, Edition 3.1" Note – The 3.1 version covers previous versions (i.e. 3.0, 2.1…).

1.2 Reference Tooling

Reference Tooling that is sited in this document and typically used during the modeling process is listed in Table 2.

Table 2 - Reference Tooling

Ref. ID	Tooling	Date/Revision	Links/File-Name where Applicable
[20]	FACE Conformance Test Suite(s)	N/A	https://www.opengroup.org/face/conformance-testsuites
[21]	MagicDraw / Cameo (NoMagic) Model Tool Integration (MTI) for FACE™ 3.0 Data Modeling	v2021_08_0_A	https://www.isis.vanderbilt.edu/face
[22]	MagicDraw / Cameo (NoMagic) Model Tool Integration (MTI) for FACE™ 3.0 Data Modeling	v2022_03_1	Not publicly available
[23]	MagicDraw / Cameo (NoMagic) Model Tool Integration (MTI) for FACE™ 3.1 Data Modeling	v2022_03_1_A	https://www.isis.vanderbilt.edu/face
[24]	FACE30_31_MagicDraw_RedirectReferences	v2022.09.0_Beta	Not publicly available
[25]	Later tool versions	TBD	Later tool versions may be posted at https://www.isis.vanderbilt.edu/face after the release of this document. Therefore, always check the above link to obtain the latest versions.

1.3 Reference Models

The models used to illustrate the modeling process are listed in Table 3.

Table 3 - Reference Models

Ref. ID	Model	Date/Revision	Links/File-Name where Applicable
[40]	FACE30_MD18.5_Space_Shuttle_Tanks.mdzip FACE30_MD18.5_SDM_3_0_2.mdzip	N/A	Delivered with this document: .\models\FACE30
[41]	FACE30_MD18.5_Space_Shuttle_Tanks.face		Delivered with this document: .\models\FACE30
[42]	FACE31_MD18.5_Space_Shuttle_Tanks.mdzip FACE31_MD18.5_SDM_3_1_2.mdzip		Delivered with this document: .\models\FACE31
[43]	FACE31_MD18.5_Space_Shuttle_Tanks.face		Delivered with this document: .\models\FACE31

1.4 Reference Scripts

The scripts referenced by this document are listed in Table 4.

Table 4 - Reference Scripts

Ref. ID	Model	Date/Revision	Links/File-Name where Applicable
[60]	run_dmvt.bat	N/A	Delivered with this document .\scripts\DMVT_DIG_IDEAL_Invocations
[61]	run_dig.bat	N/A	“”
[62]	run_ideal.bat	N/A	“

2 CTS Applications Invoked Individually During Modeling Phase

One approach is to complete a model (i.e. USM) and then submit it to the Conformance Test Suite (CTS) [20] for validation purposes. The disadvantage of this approach is that errors are not caught early in the model-development phase. Also, it is often desirable to view the code that would be generated by the queries and templates as the model is being developed.

The CTS applications of interest during the modeling phase are shown in 5.

Table 5 - CTS Application Relevant to the Modeling Phase

Application Invoked by CTS	Purpose
DMVT/DACT	Checks a model
DIG	Generates IDL
IDEAL	Generates header code

The following sections describe how to invoke these applications from a command line. A prerequisite is that the CTS [20] is installed because these applications are part of CTS.

2.1 DMVT/DACT Invocation

An iterative approach to model validation is as follows:

1. Develop portions of the model
2. Use the Model Tool Interchange (MTI) [21-23] to export the model to a .face file
3. Invoke the DMVT/DACT [60] to check the model
4. Make corrections as needed
5. If model not complete, go to step 1

Note – Before running `run_dmvt.bat` [60], you must edit the "set …" lines with paths/models applicable to your machine and your model. Also, at the end of `run_dmvt.bat` is a commented-out example of how to invoke `-help`. A good starting point is to read the `-help`.

2.2 DIG Invocation

As previously mentioned, the DIG generates IDL. The DIG is invoked by `run_dig.bat` [61].

Note – Before running `run_dig.bat` [61], you must edit the "set …" lines with paths/models applicable to your machine and your model. Also, at the end of `run_dig.bat` is a commented-out example of how to invoke `-help`. A good starting point would be to read the `-help`.

2.3 IDEAL Invocation

As previously mentioned, the IDEAL generates header code. This is invoked by `run_ideal.bat` [62].

Note – Before running `run_ideal.bat` [62], you must edit the "set …" lines with paths/models applicable to your machine and your model. Also, at the end of `run_ideal.bat` is a commented-out example of how to invoke `-help`. A good starting point would be to read the `-help`.

3 MD Model Organization Capabilities

MagicDraw provides capabilities that make it easy to organize your model in a variety of ways. The capabilities that are specifically leveraged by this document are discussed in this section. An experienced MD user can skip this section.

3.1 SmartPackages Overview

A SmartPackage contains references (i.e. pointers) to MD packages/elements. This is best illustrated by the example shown in Figure 1. In this example, "SDM_3_1_2 <<SmartPackage>>" contains the reference, "SDM_3_1_2 <<FACEDataModel>>", that points to "SDM_3_1_2 <<FaceDataModel>>" directly under ArchitectureModels package. A <<SmartPackage>> contains only references, which means that it would not contain elements.

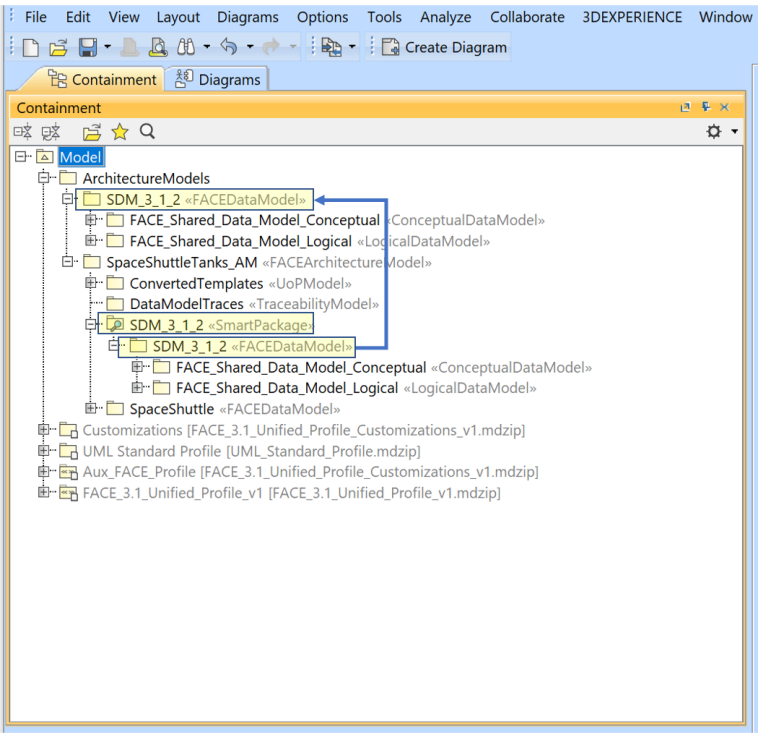

Figure 1 – SmartPackage Referencing SDM in a Different Folder

3.2 MD Model and Element IDs

Each MD model has a unique ID and each element within each MD model has a unique ID. This is important to know because if you are refactoring a model into two or more models, you would need to reset these IDs. Later sections have examples of this process, but this section just presents how to reset the IDs.

To navigate to the "**Project Properties**" panel in MD, select **File** → "**Project Properties**", which results in the panel shown in Figure 2.

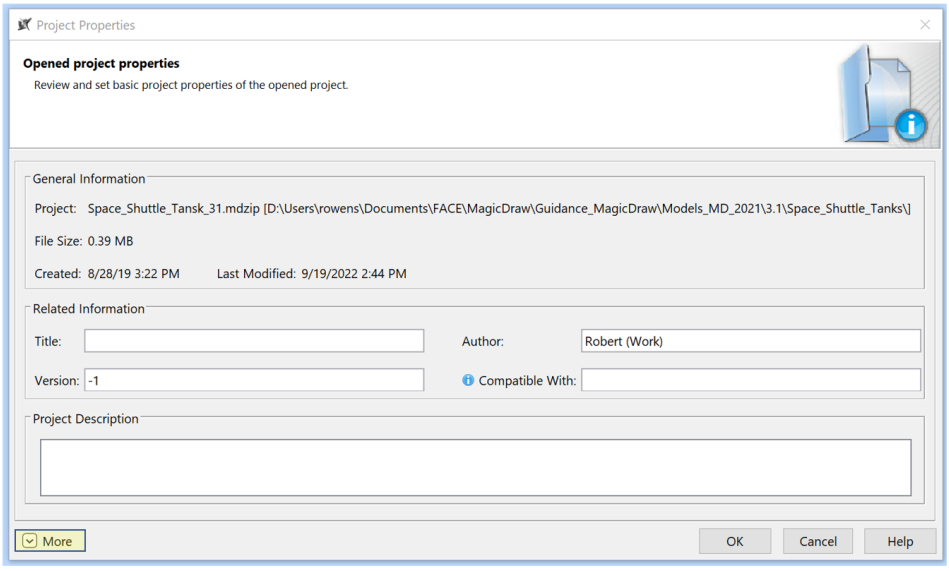

Figure 2 – Project-Properties Panel

In the Project-Properties Panel, selecting "**More**" and then selecting the "**Advanced**" tab results in the panel shown in Figure 3.

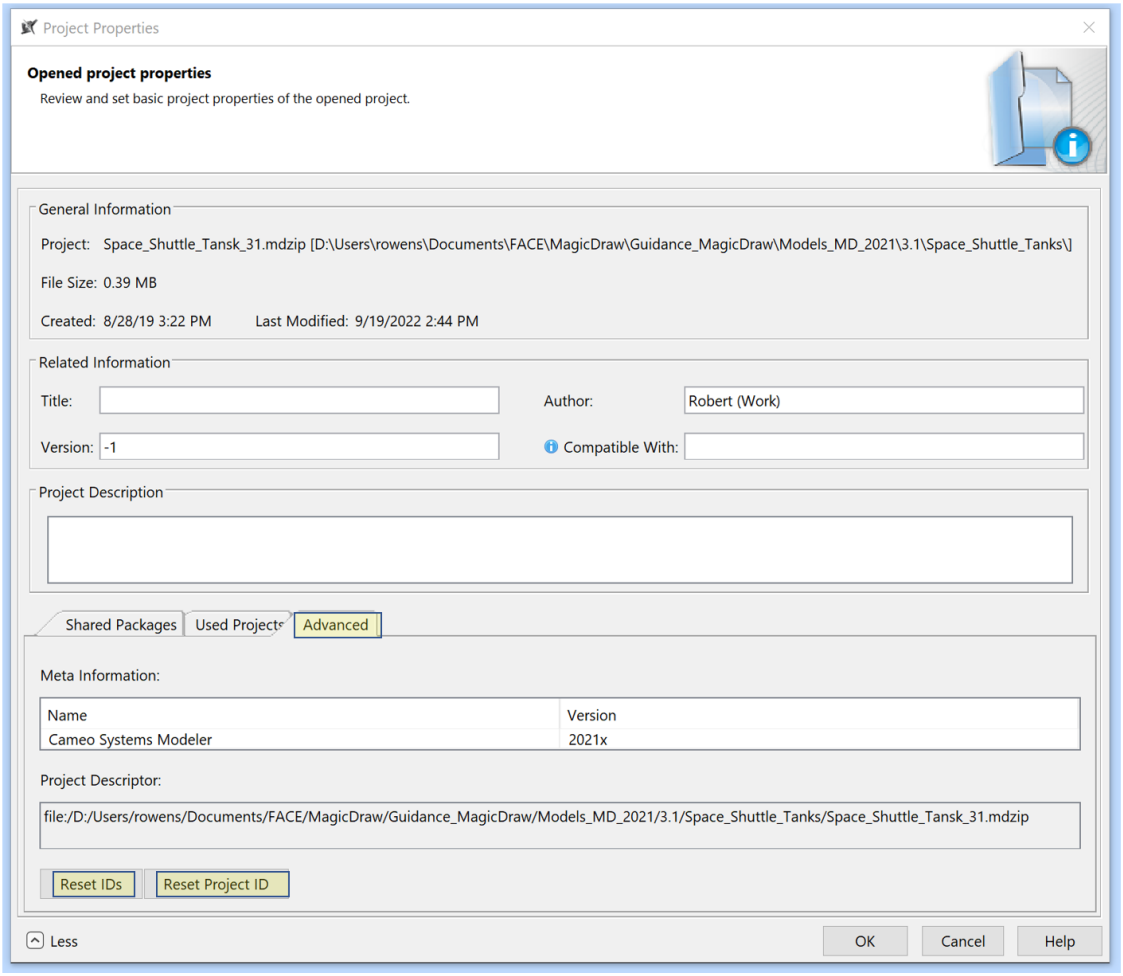

Figure 3 – Project-Properties Panel Expanded via More

3.2.1 Reset MD Model ID

Referring to Figure 3, to reset the model ID select "**Reset Project ID**". It is important to note that a 'model name' does not uniquely identify a model within MD. The "**Project ID**" uniquely identifies a model. For example, if you copied `A.mdzip` to `B.mdzip` and tried to use both `A.mdzip` and `B.mdzip` in `C.mdzip`, an error would result.

3.2.2 Reset MD Element IDs

Referring to Figure 3, to reset the model element IDs, select "**Reset IDs**". Section 3.2.1 describes the reason for resetting the "**Project ID**". Those same reasons apply to resetting the element IDs.

3.3 MD Use File

Selecting **File** → "**Use Project**" → "**Use Local Project**" results in the Figure 4 panel. Selecting `SDM_3_1_2.mdzip` and "**Finish**" uses the model in the currently opened MD model. Only the packages that are shared in `SDM_3_1_2.mdzip` are visible to the parent model. To share packages, with `SDM_3_1_2.mdzip` open, select **File** → "**Share Packages**" and follow the prompts.

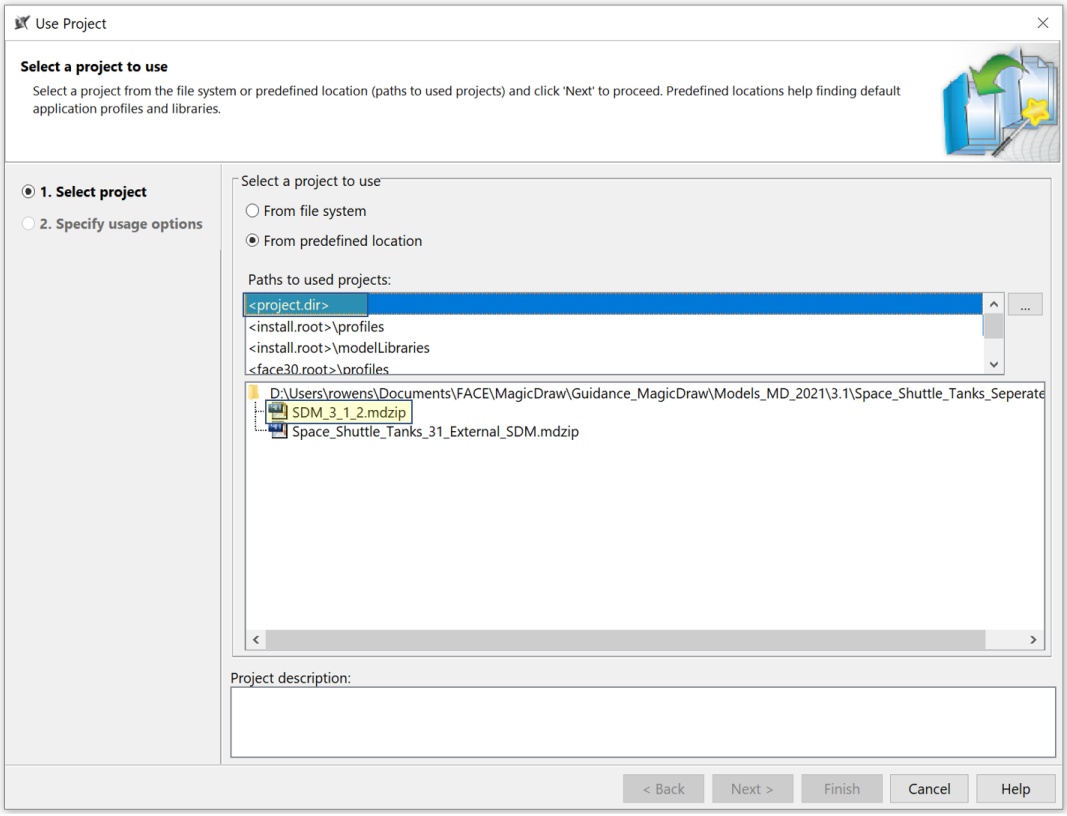

Figure 4 – Use-Project Panel

4 Model Structure

An example model structure is shown Figure 5. Arguably, this is a preferred approach to factoring models. Notice that model, `Main_Model.mdzip`, uses `DSDM_01.mdzip` and `SDM_3_1_2.mdzip`. A key concept is the 'master representation of a model', which is the version of the model that you would edit to drive subsequent changes. In this example, the master representations of the model are:

```
o  Main_Model.mdzip
o  DSDM_01.mdzip
o  SDM_3_1_2.face  // NOT the .mdzip, the .face file is the
   SDM master
```

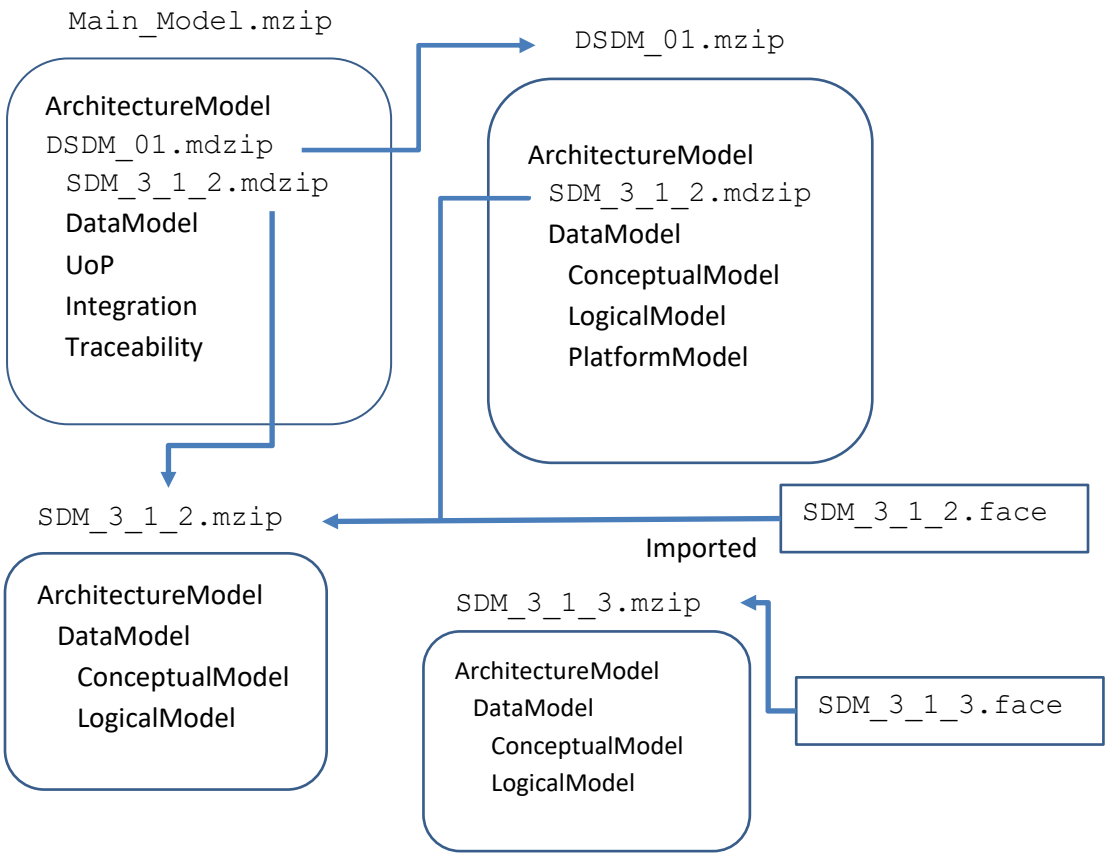

Figure 5 - Example Model Structure

4.1.1 Update Model - Master Representation: MD Model

Referring to Figure 5, models `Main_Model.mdzip` and `DSDM_01.mdzip` are master representations. The built-in MD capabilities can be used to handle the case where there is concurrent editing of the DataModel, UoP, Integration, and/or Traceability models. For example, the MD Project-

Merge capability supports merging two models where two copies of the model were edited independently.

4.1.2 Update Model - Master Representation: .face File

Referring to Figure 5, models `SDM_3_1_2.face` and `SDM_3_1_3.face` are master representations. Notice that the SDM is used in `Main_Model.mdzip` and `DSDM_01.mdzip`. For this case, upgrading to a new SDM requires the RedirectReferences Plugin [24]. The process entails first upgrading `DSMM_01.mdzip` to use the new SDM and then upgrading `Main_Model.mdzip` to use the new SDM. The process follows:

1. Import `SDM_3_1_3.face` via MTI [21-23] into `SDM_3_1_3.mdzip`. `SDM_3_1_3.mdzip` would initially (before the import) be a coped version of the starter model (e.g. `FACE31StarterModel.mdzip`), which is delivered with the MTI zip file. "**Reset IDs**" and "**Reset Project ID**" per Sections 3.2.1 and 3.2.2. Failure to reset the IDs will cause errors when using the SDM in a multi-level hierarchy.
2. Use (MD File "**Use Project**"…) `SDM_3_1_3.mdzip` in `DSDM_01.mdzip`
3. Use RedirectReferences Plugin [24] to redirect the references in `DSDM_01.mdzip` from `SDM_3_1_2.mdzip` to `SDM_3_1_3.mdzip`
4. Delete `SDM_3_1_2.mdzip` from `DSDM_01.mdzip`
5. Save `DSDM_01.mdzip`
6. Repeat steps 2 thru 5 substituting `Main_Model.mdzip` for `DSDM_01.mdzip`

Notes:

1. MagicDraw supports a model (e.g. `SDM_3_1_2.mdzip`) appearing multiple times in a hierarchy. In the case of `Main_Model.mdzip`, `SDM_3_1_2.mdzip` is used directly in `Main_Model.mdzip`, but is also used indirectly via `DSDM_01.mdzip`. If a model is used multiple times, it will only appear one time in the containment tree (e.g. the containment tree of `Main_Model.mdzip`).
2. The same SDM version must be used throughout the hierarchy. It is necessary during the updating phase for a MD model to reference two SDMs (e.g. `SDM_3_1_2.mdzip` and `SDM_3_1_3.mdzip`); however, after the completion of the updates, there should be one and only one SDM in the hierarchy. From a general perspective, having two different SDMs in the hierarchy would cause problems when using the MTI [21-23] for exports. SmartPackages could be used to allow two different SDMs in a MD model, but this would be very confusing and error prone.
3. The above discussion would apply to DSDMs if the master representation of the DSDM was the DSDM `.face` file.

4.1.3 Non-Supported Updates

The case of upgrading an existing MD Model (i.e. `.mdzip`) based on importing, via MTI [21-23], an updated .face file is not supported. In other words, if a MD Model is populated via importing a `.face`

file and later that `.face` file is updated, then reimporting the updated `.face` file is not supported. This is because the MTI does not support an import update mode. Importing a `.face` file a second time results in two copies of the imported model in the MD model.

5 Refactoring a Model to Another Organization

Often it is desirable to refactor a MD model from all the content being in one MD model to some of the content being in two or more MD models. For example, in Figure 6, the SDM is internal to the MD model. It would be desirable for the SDM to reside in a separate MD model as shown in Figure 7.

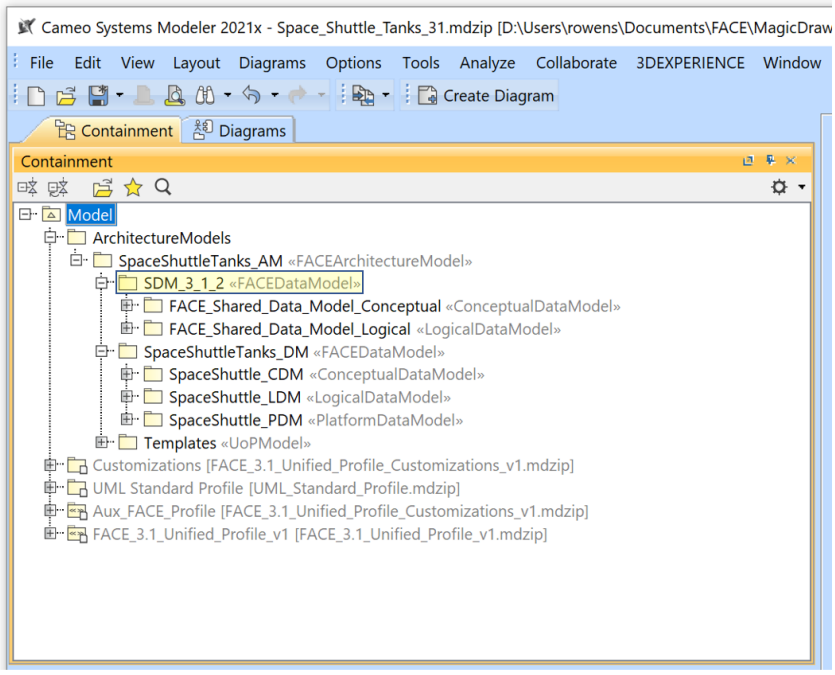

Figure 6 – SDM Content Internal to the MD Model

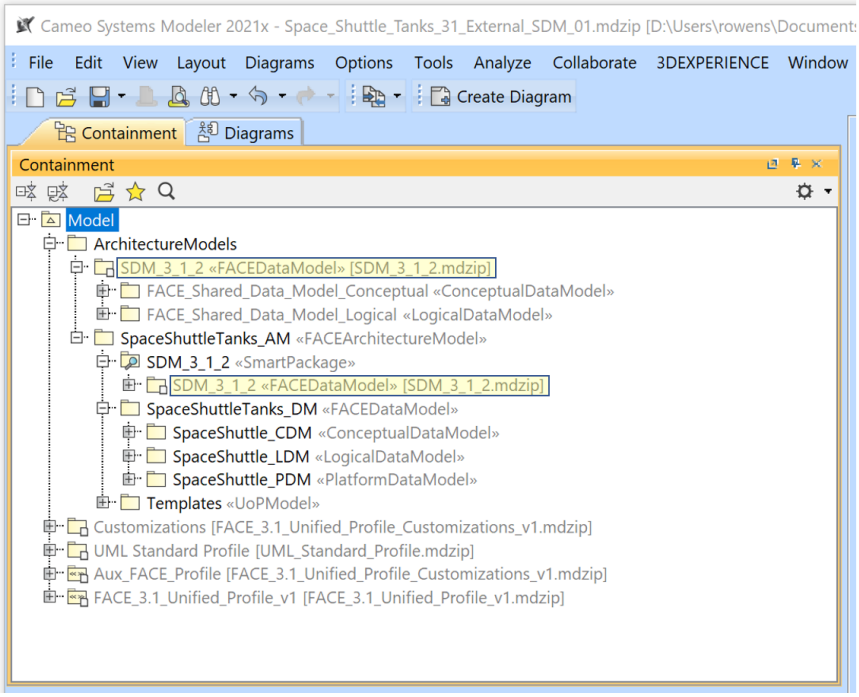

Figure 7 – SDM Content External to the MD Model

The steps to convert the model in Figure 6 to the model in Figure 7 is as follows:

1. Copy the original model (i.e. Figure 6 model) to a model named `SDM_3_1_2.mdzip`
2. Using `SDM_3_1_2.mdzip` model perform the following steps:
    a. Reset the MD Model ID and Element IDs per Sections 3.2.1 and 3.2.2
    b. Delete the non-SDM content (i.e. SpaceShuttleTanks_DM, Templates)
    c. Move package SDM_3_1_2 to be directly under ArchitectureModels
    d. Share (right-click on SDM_3_1_2 "**Project Usages**" → "**Share Packages…**" → …) SDM_3_1_2
    e. Save `SDM_3_1_2.mdzip`
3. Open the original model and:
    a. Use File `SDM_3_1_2.mdzip` per Section 3.3.
    b. Use RedirectReferences Plugin [24] to redirect the references from package SDM_3_1_2<<FACEDataModel>> to package `SDM_3_1_2.mdzip`
    c. Delete package SDM_3_1_2 <<FACEDataModel>>
    d. Create SmartPackage (right-click on **SpaceShuttleTanks_AM** → "**Create Element**" → "**SmartPackage**" → …) as shown in Figure 7.

    Note – You must Drag SDM_3_1_2 <<FACEDataModel>> `[SDM_3_1_2.mdzip]` to the SmartPackage.
    e. Save the original model

6 Modeling Specific FACE Constructs

This section presents examples of modeling particular constructs with the FACE UAF profile.

6.1 Associations

An example Association modeled with the FACE UAF profile is shown in Figure 8. In many profiles there is a line directly from LevelProvider<<Conceptual Association>> to LevelSensor<<ConceptualEntity>>, but with the FACE UAF profile there is an intermediary element sensor<<ConceptualParticipant>>. The sensor<<ConceptualParticipant>> tag participatingEntity points to LevelSentor<<ConceptualEntity>>. Otherwise, Association models are as would be expected.

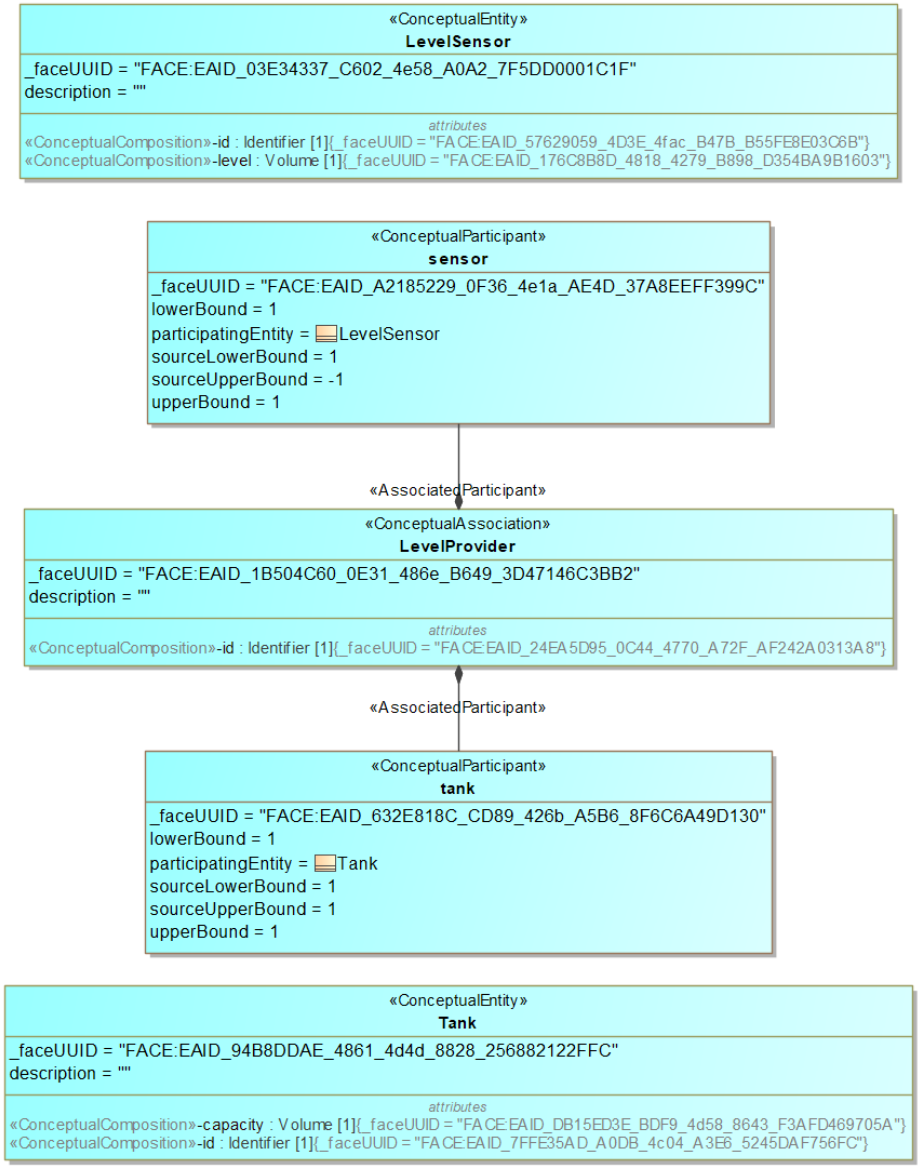

Figure 8 – Association Example

6.2 Enums

An example Enum model is shown in Figure 9 and in models [40] and [42]. Notice that an <<EnumerationConstraint>> is depicted. An <<EnumerationConstraint>> is optional, but in this case is provided to limit the enumeration literals to a subset of the literals in <<LogicalValueTypes>>.

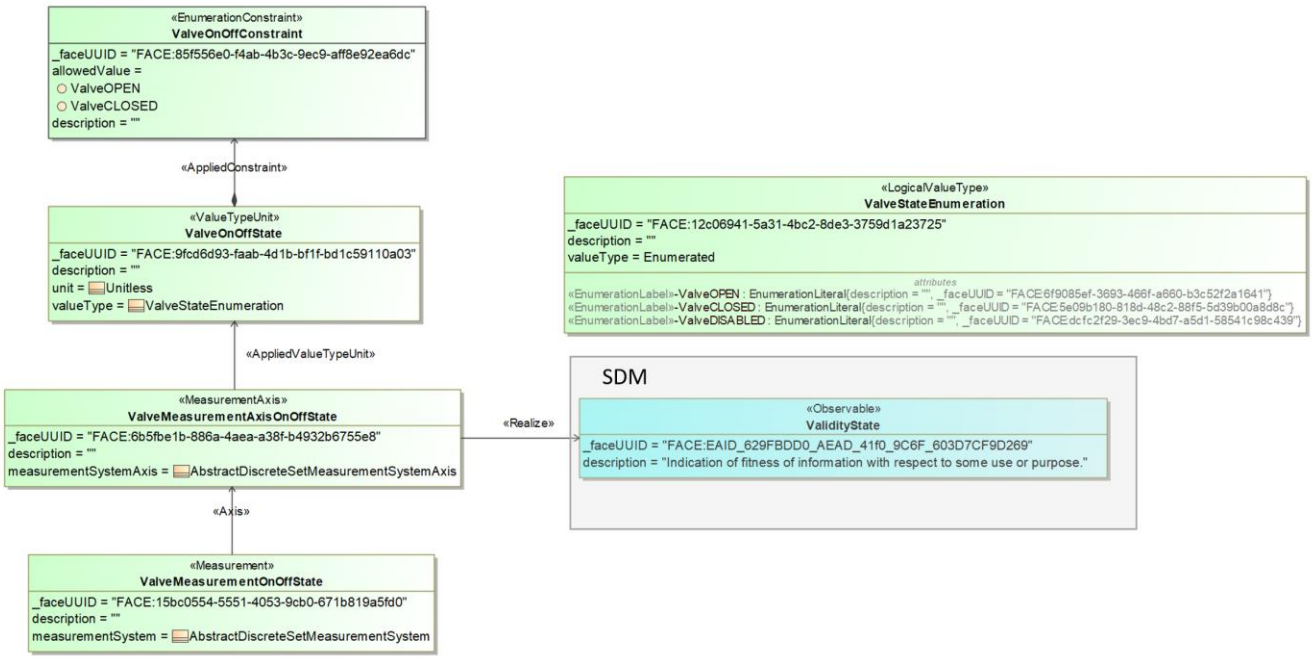

Figure 9 – Enum Example

6.3 Extending SDM Measurements

The document "Shared Data Model Governance Plan, Edition 3.1" [8] has the following statement:

> "A Basis Element (defined in the appendices) is a type of Data Model element that must reside in the SDM. Basis Elements are managed by the SDM CCB. Non-Basis Elements are not required to reside in the SDM, but can be added if there is potential for reuse. Once added to the SDM, non-Basis Elements are managed by the SDM CCB.
>
> > Note: "Basis Element" is not to be confused with similarly named metatypes in the FACE Data Model Language (i.e., Basis Entity in Edition 3.0 of the FACE Technical Standard)."

To enter a ticket to add to the SDM, use:

https://ticketing.facesoftware.org/

Notes:

1) Basis Element per Standard - There are separate lists of Basis Elements for each Standard Edition (e.g. 2.0, 2.1, 3.0, and 3.1). See the specific appendices in [8] applicable to each edition.
2) Requesting Units Added to the SDM - In some case, the Units (e.g. `face.datamodel.logical.Unit` for FACE Technical Standard, Edition 3.1) defined in the

SDM are not sufficient for your modeling scenario. See the instructions above (Section 6.3) to enter a ticket to request additions to the SDM. A `face.datamodel.logical.Unit` is a Basis Element; and thus, you cannot create a Unit in your USM. You must use Units from the SDM.

7 Relationships Between Non-FACE Elements and FACE Elements

This section describes how to modify the customization to allow relationships between non-FACE elements and FACE elements. An example of using these types of relationship is also provided.

7.1 Customization Modifications to Allow Non-FACE Relationships

For the following versions and earlier versions of the MTI, the customizations prevented non-FACE elements from connecting to FACE elements:

```
FACE30_MagicDraw_MTI_v2022_03_1.zip
FACE31_MagicDraw_MTI_v2022_03_1_A.zip
```

Document [7] provides instructions on how to modify those customizations to allow non-FACE elements to connect to FACE elements.

7.2 Requirements Related to FACE Elements Example

This section provides an example of non-FACE elements related to FACE elements. The example pertains to a two-way relationship between a requirement and a FACE element. In this example, Figure 10 shows an overall view of the MD model, Figure 11 shows a closeup view of the containment tree, and Figure 12 shows a closeup view of the diagram. Notice that the <<requirement>> points to the <<LogicalAssociation>> via a <<trace> and that the <<LogicalAssociation>> points to the <<requirement>> via a <<satisfy>>.

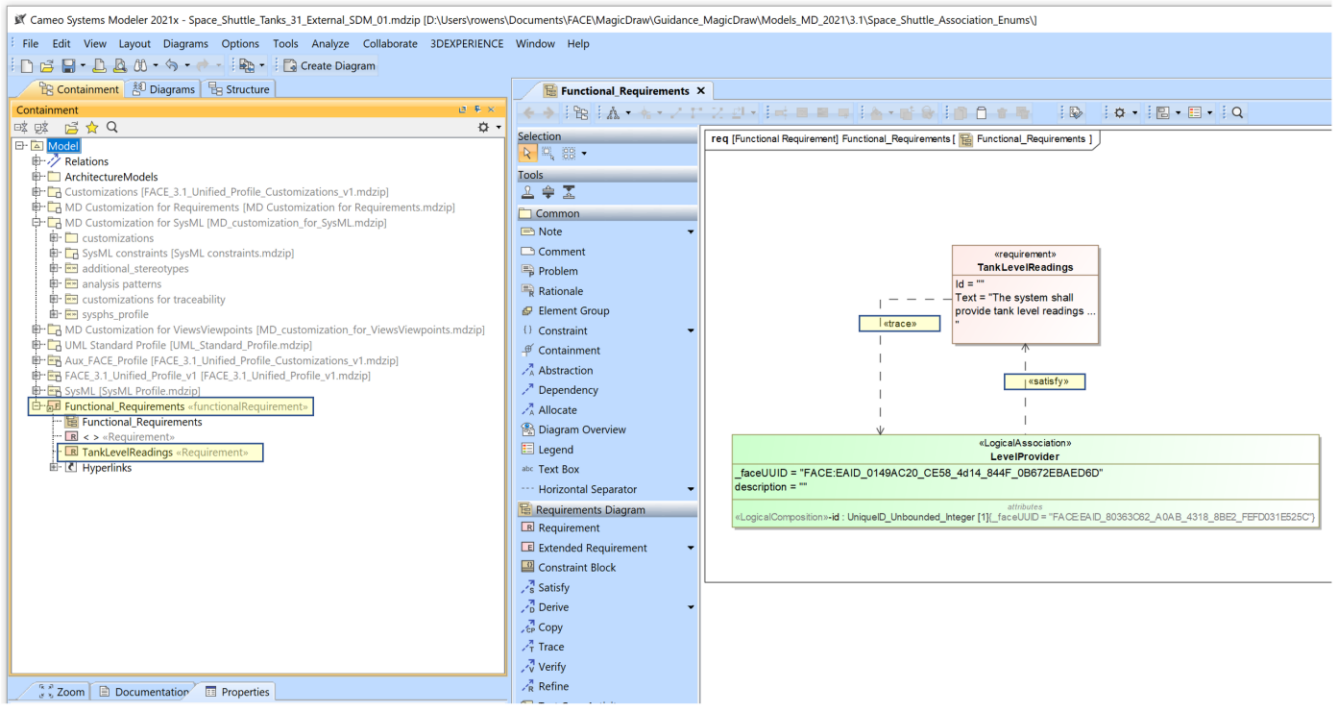

Figure 10 – Example of Requirements Relating to a FACE Element

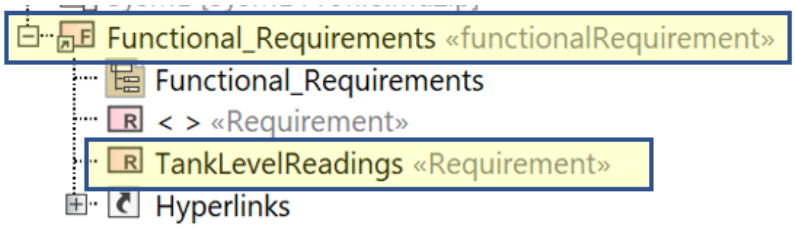

Figure 11 - Example of Requirements Relating to a FACE Element – Containment Tree

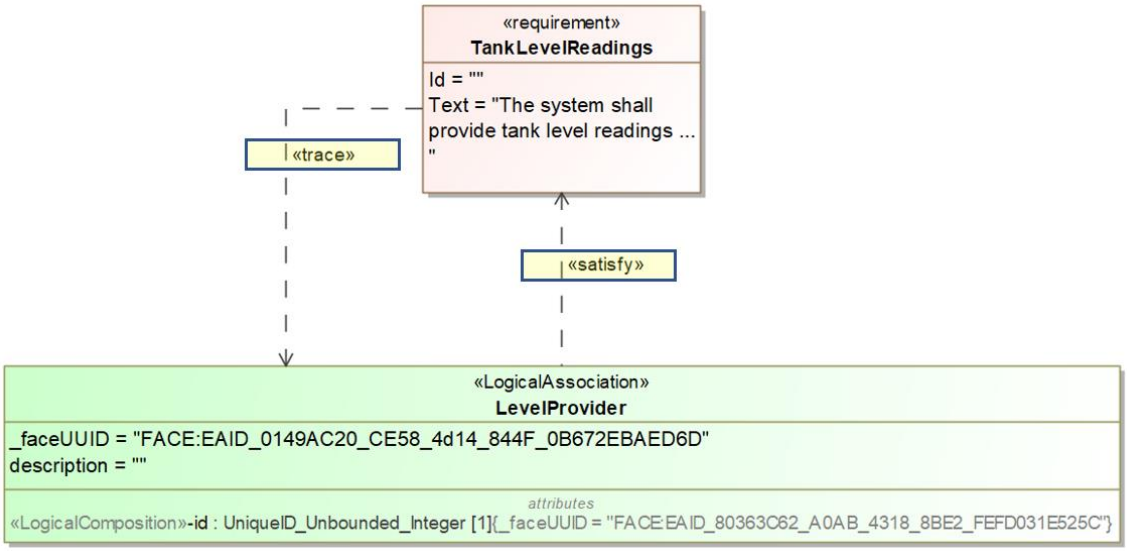

Figure 12 - Example of Requirements Relating to a FACE Element – Diagram

The Figure 10 model was created by first creating a “Requirement Diagram” as shown in Figure 13. It should be noted that a SysML license is necessary to create a “Requirement Diagram”. Once the diagram was created, the tool palette was used to create <<requirement>>. The <<LogicalAssociation>> was dragged onto the diagram from the FACE model. The final step was to add the <<trace>> and <<satisfy>> relationships from the tool palette.

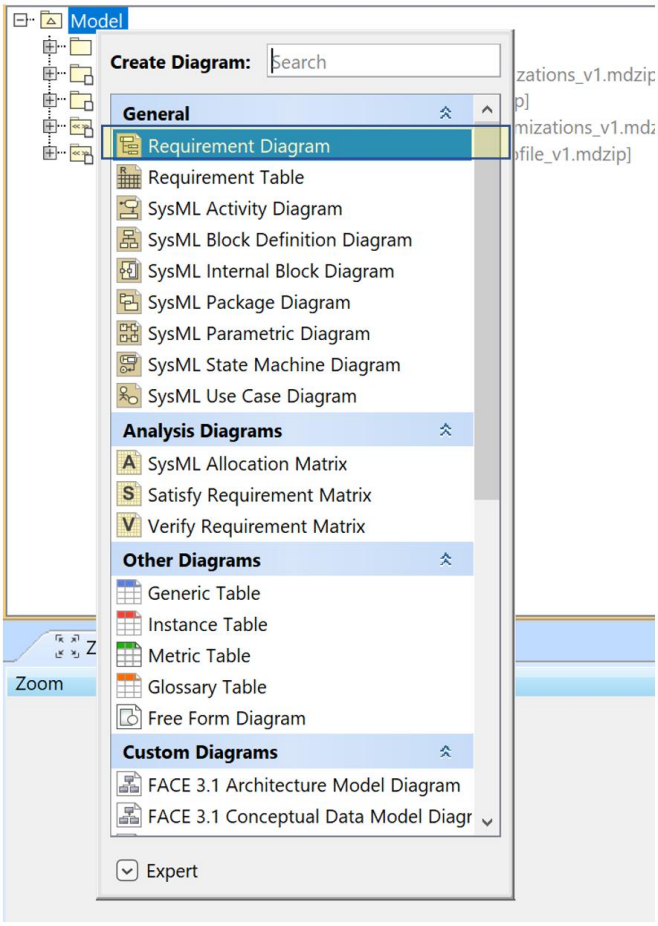

Figure 13 – Creation of Requirement Diagram

8 Acronyms

Acronym	Description
CDM	Conceptual Data Model
CSCI	Computer Software Configuration Item
CTS	Conformance Test Suite (CTS)
DID	Data Item Description
DIG	Data-model IDL Generator
DM	Data Model DM
DMVT	Data Model Validation Tool
DOD	Department of Defense
DSDM	Domain-Specific Data Model
EMOF	Essential Meta-Object Facility
FACE™	Future Airborne Capability Environment
IDL	Interface Definition Language
Java EE	Java Enterprise Edition
Java SE	Java Standard Edition
LDM	Logical Data Model
MD	MagicDraw
MIL-STD	Military Standard
MTI	Model Tool Interchange
MOF	Meta-Object Facility
MTI	Model Tool Interchange
NAVAIR	Naval Air Systems Command
OCL	Object Constraint Language
OMG	Object Management Group
OS	Operating System
PDM	Platform Data Model
POSIX	Portable Operating System Interface
RRR	Redirect References Plugin
SDD	Software Design Document
SDM	Shared Data Model
SDP	Software Development Plan
SMT	System Modeling Tool
SRS	Software Requirements Specification
STD	Software Test Description
STR	Software Test Report
UAF	Unified Architecture Framework
UDP	User Datagram Protocol
UI	User Interface
UoC	Unit of Conformance
UoP	Unit of Portability
XMI	Extensible Markup Language Metadata Interchange
XML	Extensible Markup Language
XSD	Extensible Markup Language Schema Definition