Improving PDM Testability through Standards Harmonization

KC Morris, NIST[*]*, USA*

1. Introduction

Two emerging standards in the area of Product Data Management (PDM) support interoperability among PDM systems and between PDM systems and their clients. The Object Management Group's PDM Enablers [4] standard supports system interoperability using a server-based architecture where the interoperating systems could be either two PDM systems or a PDM system and a client application. Within the STEP (formally known as ISO 10303: the Standard for the Exchange of Product Model Data) community work is underway on a standard for the exchange of Product Management Data. One result of these efforts is the PDM Schema developed by PDES Inc. and ProSTEP. [5] The PDM Schema is based on several of the application specific standards emerging within STEP including the first formally adopted one, Application Protocol 203[6], and harmonizes the PDM data definitions shared amongst those standards. The Schema supports the interchange of PDM data between systems. The PDM Schema and the Enablers play different roles in an enterprise-wide software system yet consistency between them is important to allowing smooth operation of the system.[1] Furthermore, consistency between the standards supports testing efforts in that the artifacts of testing one type of interface can be reused in testing the other type. Specifically the data used to test the PDM Schema can be used in defining scenarios for testing the Enablers. Vice versa, the scenarios for testing the Enablers can be used in developing tests based on the PDM Schema. [7]

The National Institute of Standards and Technology (NIST) is contributing to these efforts through the Testability of Interaction-Driven Manufacturing Systems (TIMS) project. The TIMS project is investigating how the two standards—the PDM Enablers and PDM Schema—could be tested in concert to ensure compatibility. Essential to this activity is a mapping between the standards. This paper describes the initial mapping used and our approach to documenting the mapping in a programmatic form. The mapping is documented using the EXPRESS-X language [2], also work in progress within the STEP community. The complete mapping can be found at the TIMS project web site (http://www.mel.nist.gov/msid/tims/). The remainder of this paper describes briefly how the mapping is being used in the TIMS project, illustrates features of the mapping that add to its complexity, and concludes with recommendations on the approach to documenting the relationship between standards.

2. Mapping Approach

The mapping is captured as a series of views on the PDM Schema. Each view contains the data needed by one or more objects accessible using the PDM Enablers interfaces. (The terms "Enablers" and "PDM Schema" are used hereafter to refer to the two specifications.) The views are then traversed via the Expresso Toolkit [3] which processes EXPRESS-X and represents the views in the Common LISP Object System (CLOS). Calls to the Enablers are generated by a LISP program. For a given data set represented using the PDM Schema the software generates calls to the Enablers which reproduce the system state at the time that the data was produced.

The mapping between the PDM concepts found in the Enablers and PDM Schema is contained in its own EXPRESS-X SCHEMA_VIEW called the pdm_enablers_view which references the EXPRESS SCHEMA, pdm_schema, containing the PDM Schema. The pdm_enablers_view contains the twenty views described in the table below. The mapping covers the Enablers modules PdmResponsibility, PdmProductStructureDefinition, PdmDocumentManagement and portions of the PdmFoundation module as needed by the others. Some of these views support more than one object interface. Others support a portion of an object interface. The views shown in bold face in the table are contained only in the pdm_enablers_view and will be discussed in the section on **Abstract Views**.

Table 1: Views of the PDM Schema supporting the PDM Enablers

View	Enabler	PDM Schema Entity
parts	PartMaster	product
standard_parts	PartMaster.standard_part	product product_category_relationship
part_categorizations	PartMaster	product_related_product_category product
part_revisions	PartRevision PartData PartDataIteration	product_definition_formation
assemblies	PartStructure NextAssemblyUsageOccurrence PromissoryUsageOccurrence MakeFromUsage	product_definition_relationship (includes Next_assembly_usage_occurrence as subtype)
alternates	Alternate	alternate_product_relationship
substitutes	Substitute	assembly_component_usage_subst itute
part_document_relationship	PartDocumentRelationship	applied_document_reference **document_product**
applied_assignments	--abstract--	applied_person_and_organization_ assignment product product_definition_formation
design_supplier_relationship	DesignSupplierRelationships	**applied_assignments**
part_supplier_relationship	PartSupplierRelationships	**applied_assignments**
object_owners	ObjectOwner	**applied_assignments**
oc	--abstract--	**applied_assignments**
object_creators	ObjectCreator	**oc**
person_party	Person	person personal_address
organization_party	Organization	organization organizational_address
document_product	--abstract--	document_product_association product_definition
document_masters	DocumentMaster	**document_product**
document_revisions	DocumentRevision	**document_product**
documents_with_files	DocumentIteration DocumentFileRelationship	**document_product** product_definition_with_associated_documents
files	File	document

The basic approach to mapping Enablers onto the PDM Schema is to create a view representing each of the objects in the Enablers. An EXPRESS-X VIEW construct captures the data needed from the PDM Schema in order to describe the Enablers' object. At a minimum the VIEW contains a FROM clause and a SELECT clause. The FROM clause identifies the entity or entities to use as the source of the view; the SELECT clause identifies the attributes from the entity and associates names to be used by the view. The alternates view below illustrates this approach.

```
VIEW  alternates
(*  This view supports the PDM Enabler object
**    Alternate
*)
FROM  apr : pdm_schema.alternate_product_relationship;
SELECT
      apart:  string :=apr.alternate.id;
      bpart:  string :=apr.base.id;
      name: string :=apr.name;
      description:  string  :=apr.definition;
      basis:  string  :=apr.basis;
END_VIEW;
```

The attribute "apart" gathers its data from the product entity. This data is used to identify the part master objects which are connected through the alternate object.

3. Types of Complexity

Despite the similarity in the subject matter of the two specifications, the mapping from the PDM Schema to the Enablers is not without complexity. In fact none of the views reflect a one-to-one mapping of a single entity type with all of its instances to a single object in the Enablers. Seven categories of complexity arise in the mapping as will be described below.

Implicit Joins

In the PDM mapping, the views substitutes and alternates (depicted above) map the extents for a single entity type to a single instance of an object type; however, they pull in data values from other entity types that are used to establish the connections between the objects. In database terminology this is known as an "implicit join." Thus even these most basic views, which map one entity type to one object type, actually involve multiple entity types.

Selection

Another category of relatively simple views in the mapping contains those views resulting from a straight forward selection of instances from the extent of a single entity type. This type of view is depicted for parts below:

```
VIEW parts
(*   This view supports the PDM Enabler objects:
**    PartMaster.
*)
FROM
      p: pdm_schema.product;
WHERE (
(*  when the product is not a document  *)
      SIZEOF (QUERY
      (c <* EXTENT ('pdm_schema.product_related_product_category')
      | (c.name = 'document') AND (p IN c.products))) = 0
);
SELECT
(*  PartMaster attributes  *)
```

```
      part_name : string := p.id    ;
      short_description : string := p.name;
      long_description : string :=  p.description;
END_VIEW;
```

The PDM Schema represents both products and documents as products but the Enablers specification clearly separates these types of data. The parts view implements this concept by restricting the instances of the entity type product to only those products that are not documents. In EXPRESS-X the WHERE clause is used to express the restriction. In order to implement this restriction the product_related_product_category ENTITY is used. Thus both an implicit join and a selection are applied to the data.

Unions

In some cases, we have two distinct sets of instances from the PDM Schema that represent a single collection of objects in the Enablers. In these cases we can use the EXPRESS-X PARTITION construct to combine the union of the two sets into a single set. In the mapping this is done for the VIEW organization_party which unites all organizations with their address or with blank addresses if none are contained in the dataset. The alternative to using a PARTITION in this situation is either not including organizations without addresses or looking up each address individually in the test generation source code which processes the mapping. The former alternative is insufficient because some organizations may be omitted from the generated tests corrupting the results. The consequence of the latter is that, by not capturing all the semantics of the mapping, it is subject to inconsistent implementation.

```
VIEW  organization_party
(*  This view supports the PDM Enabler object
**    Organization
*)
PARTITION waddr  :
(**  organizations that have addresses  **)
FROM  org  : pdm_schema.organization;
      addr : pdm_schema.organizational_address;
WHERE
      (org IN addr.organizations);
SELECT
      name : string := org.name;
      organization_type : string := org.description;
      phone_number : string := addr.telephone_number;
      street_address : string := addr.street_number + addr.street +
addr.postal_box; city : string := addr.town; state : string :=
addr.region;
      country : string := addr.country;
      mail_code : string := addr.postal_code;
      e_mail : string := addr.electronic_mail_address;
      id : string := org.id;
PARTITION woaddr  :
(**  organizations that do not have addresses  **)
FROM  org : pdm_schema.organization;
WHERE (
      (SIZEOF (QUERY
      (a <* EXTENT('pdm_schema.organizational_address')
      | (org IN  a.organizations))) = 0   )
);
SELECT
      name : string := org.name;
      organization_type : string := org.description;
      phone_number : string := '';
      street_address : string := '';
      city : string := '';
      state : string := '';
```

```
      country : string := '';
      mail_code : string := '';
      e_mail : string := '';
      id : string := org.id;
END_VIEW;
```

Multiple views for one object

In some cases multiple views were used to capture all the data needed for one object. For example, the three views—parts, standard_parts, and part_categorizations—each contain the information needed for the Enabler's PartMaster object. The data is derived from the entities product, product_category_relationship, and product_related_product_category.

The standard_parts view distinguishes those objects that are standard. In the Enablers standard parts are distinguished as an attribute of the PartMaster object which is set to true. To simplify the EXPRESS-X these flags are set using the LISP code by traversing the list of standard parts only after the PartMaster objects have been created. This could have been expressed directly in EXPRESS-X but would have resulted in a more complex EXPRESS-X schema.

Another reason to divide these views was that the combined view created a three way join (three entities in the FROM clause) which was not processed efficiently. This problem could be factored out when the EXPRESS-X standard is adopted and more robust implementations appear.

Mapping dependence on data values

While the mapping is represented in meta-data, i.e. at the schema level, it is not entirely independent of the data. The difference in the representation of products and documents, as well as, the views standard_parts, files, applied_assignments and its derivatives, illustrates how the semantics of the data in the case of the PDM Schema are represented at the meta-level in the case of the Enablers. The file view (shown below) contains a clear example of this difference of representation level.

```
VIEW  files
(*   This view supports the PDM Enabler objects:
**    File
*)
FROM
      df:  pdm_schema.document;
WHERE  (
(*  The first 2 cases are instances of document_file and are
conventions for using the PDM Schema  *)
      (df.kind.product_data_type = 'digital')
      OR
      (df.kind.product_data_type = 'physical')
(* the third case is an instance of document and is established in the
AP 203 Usage Guide *)
      OR
      (df.kind.product_data_type = 'cad_filename')
 );
SELECT
      filename :string := df.id;
      type : string := df.kind.product_data_type;
END_VIEW;
```

The file view is a selection of those instances of the document entity that have been assigned certain values for the product_data_type of the instance. Differing conventions were found in the data for values of the product_data_type attribute that indicate that the instance is indeed referring to a "computer file" in the traditional sense. Specifically the usage conventions for the PDM Schema state that the value 'digital' or 'physical' would be used for the product_data_type attribute; however, the AP 203 Usage Guide recommends using the term 'cad_filename.' Additionally, with AP 203 only the supertype document is populated, whereas, the PDM Schema usage conventions call for the subtype document_file to be used.

The implication of this use of data to convey information that is meta-data in the other specification is that the mapping between the two specifications may be more volatile than the specifications themselves. Furthermore the mapping can not be finalized without regard for usage conventions on the data and as such can not be completely specified for specifications with informal usage guidelines.

Abstract views

The schema contains three examples of *abstract views* that are distinguished by bold type face in the table. The term abstract views is used to distinguish views that do not directly support an object in the Enablers but rather support other views which directly support the objects. Both the document_product and applied_assignment views use the EXPRESS-X construct of PARTITION to represent set unions that consolidate instances. In these cases, the union consolidates instances are related in the PDM Schema through different paths in the product, product_definition, and product_definition_formation triad used in STEP to represent a specific configuration of a product. This abstract view simplifies the views that use it, thereby clarifying them, by encapsulating a fundamental difference between the two specifications.

The document_product view shown below identifies the product, version, and document information associated with a given document. The three partitions provide the access path to the product and product version information from the three different underlying types of the SELECT type product_or_formation_or_definition used in the document_product_equivalence ENTITY. (An example of the use of the document_product VIEW is shown in the following section in the part_document_relationship VIEW.)

```
VIEW document_product
PARTITION p1  :
  FROM  d : pdm_schema.document_product_association;
       pd: pdm_schema.product_definition;
WHERE  (
       (pd = d.related_product)
  );
SELECT
       ip : product := pd.formation.of_product;
       ipdf : product_definition_formation := pd.formation;
       idoc : document := d.relating_document;
       master_id : string := pd.formation.of_product.id;
       rev_id : string := pd.formation.id;
PARTITION p2  :
  FROM      d : pdm_schema.document_product_equivalence;
       p:    pdm_schema.product;
       pdf:  pdm_schema.product_definition_formation;
WHERE  (
       (p = d.related_product)
       AND
       (pdf.of_product = p)
  );
SELECT
       ip : product := p;
       ipdf : product_definition_formation := pdf;
       idoc : document := d.relating_document;
       master_id : string := p.id;
       rev_id : string := pdf.id;
PARTITION p3  :
  FROM  d : pdm_schema.document_product_equivalence;
       pdf:  pdm_schema.product_definition_formation;
WHERE  (
       (d.related_product = pdf)
  );
SELECT
       ip : product := pdf.of_product;
       ipdf : product_definition_formation := pdf;
```

```
      idoc : document := d.relating_document;
      master_id : string := pdf.of_product.id;
      rev_id : string := pdf.id;
END_VIEW;
```

Incomplete specification of mapping for processing convenience

The final complexity category has to do with incomplete specification of the mapping. This category, in fact, removes complexity from the mapping at the cost of adding complexity to understanding the mapping. An example of such a convenience is in the view part_document_relationship shown below.

```
VIEW part_document_relationship
(*   This view supports the PDM Enabler objects:
**    PartDocumentRelationship
*)
FROM
      ar:  pdm_schema.applied_document_reference;
      dp:  pdm_enablers_view.document_product;
WHERE  (
      dp.idoc = ar.assigned_document
  );
SELECT
      master_id:  string := dp.master_id;
      rev_id: string := dp.rev_id;
(*  pvs is the set of document_reference_items which must be filtered
**   for product versions
*)
      pvs:  set [1:?] OF document_reference_item := ar.items;
END_VIEW;
```

In this example the attribute pvs does not directly represent what is needed by the Enabler's PartDocumentRelationship interface. The pvs short cut actually addresses two complexity features. The first is that the PDM Schema represents the relationship of a part with a document or set of documents through a multi-valued attribute, the set "items," whereas the Enablers express that same relationship as multiple instances of an object relating the part and the document. While this difference is easy enough to comprehend and encode in a programming language, it is not as easily expressed in EXPRESS-X.

The second complexity feature addresses a fundamental aspect of the relationship itself. In the PDM Schema the documents may be related to product_definition, product_definition_formation_relationship, product_definition_relationship, shape_aspect, shape_aspect_relationship. In practice we found only relationships to the product_definition and used that to map the documents to the part version; however, given a different data set which uses one of the other types of relationships our mapping would be undefined. The test generation code that uses this mapping to generate calls to the Enablers generates error messages in the uncovered cases. Similar to the above situation these messages would be difficult to convey in the EXPRESS-X.

Even given our assumption that the document was related to a product_definition entity encoding that assumption in EXPRESS-X would be more complicated than the equivalent LISP code that we use to process the set. Below is an example of pseudo-EXPRESS-X that would convey the meaning but these expressions are illegal EXRPESS-X and would need to be replaced by significantly more complicated functions.

```
(* -- assuming select type is product_definition
part_ids: set [1:0] OF string := ar.items[*].formation.of_product.id;
ver_ids:  set [1:0] OF string := ar.items[*].formation.id;
*)
```

4. Conclusion and Future Directions

When two standards are defined to be used in concert, such as the PDM Schema and the Enablers, there is clearly a need to define precisely and at a detailed level how they relate to each. The Enablers standard whole-heartedly references the STEP standards in this area (although not the PDM Schema itself but rather its precursor, AP 203.) One of the Enablers' modules supports importing of exchange files based on STEP. In this case the behavior of the import operation is not defined without such a mapping. Even with more granular access than file import in order to perform round-trip testing of implementations supporting both standards, a mapping is necessary to define expected results for such tests.

On the other hand, as has been shown above, such a mapping is subject to conventions on the use of data, and in some cases an exhaustive mapping may be impractical. Despite these shortcomings, something is better than nothing. Furthermore, formalizing the mapping definition can help to highlight where the shortcomings are and where they aren't, thereby simplifying the problem of implementing and testing against a standard.

Given this state of affairs we would recommend the EXPRESS-X approach to mapping combined with mapping guidelines which address how to handle difficult situations such as the need for extensibility and incomplete specification. EXPRESS-X is a useful means of capturing a mapping. It allows for formal declaration of the mapping while at the same time separating the semantic mapping for the standards from the implementation of a particular mapping system, such as our test generation system. To improve usability of this approach, conventions should be established which formalize how the complexity categories described above are handled.

One thing not addressed by the mapping presented here is coverage. By design the PDM Schema and the Enablers have different scopes. The mapping presented here only addresses a subset of their overlapping scopes. In doing so a little effort has been made to ensure that nothing has been left out of the mapping. The complete EXPRESS-X schema found at the TIMS homepage contains a couple views that are intended to uncover any instance data that may have been left out as the result of a selection in the mapping. The test generation software also checks in certain spots for uncovered data as was described in the section on incomplete specification. TIMS has also uncovered certain constructs such as the STEP exchange context information that does not map into an Enablers environment. Methods for automatically and more rigorously uncovering gaps in the mapping could be established based on a formal mapping technique such as EXPRESS-X.

5. References

1. David Starzyk, STEP and OMG Product Data Management Specifications: A Guide for Decision Makers, August 1999, http://www.omg.org/cgi-bin/doc?mfg/99-08-02.

2. ISO TC184/SC4/WG11 N078, EXPRESS-X Language Reference Manual, http://www.steptools.com/library/express-x/, 1999.

3. The Expresso Homepage, http://www.nist.gov/expresso, 1999.

4. PDM Enablers revised submission (including errata changes), http://www.omg.org/cgi-bin/doc?mfg/98-02-02, 1998.

5. Version 1.1 of the PDM Schema is a pre-normative specification available from http://www.pdm-if.org/, 1999.

6. ISO/DIS 10303-203:1994, Industrial automation systems and integration — Product data representation and exchange — Part 203: Configuration Controlled 3D Designs of Mechanical Parts and Assemblies. Available from ISO, http://www.iso.ch/.

7. KC Morris and David Flater, "Design of a Flexible, Integrated Testing System for STEP and OMG Standards,"National Institute of Standards and Technology, forthcoming.