PLMXQuery: Towards a Standard PLM Querying Approach

Mohamed-Foued Sriti, Philippe Boutinaud

PLMXQuery: Towards a Standard PLM Querying Approach

Mohamed-Foued Sriti[1], Philippe Boutinaud[2]

[1]Imam Muhammad Ibn Saud Islamic University, Computer Science Department, Riyadh, Kingdom of Saudi Arabia
mfsriti@ccis.imamu.edu.sa
[2]CADeSIS, R&D Department, Courbevoie, France
pboutinaud@cadesis.com

Abstract. Experience with data exchange standards has shown considerable limitations in their integration in commercial tools and their interoperability. The current trend is to use XML as a mean of exchanging data. We present in this paper a new standard approach for querying and exporting data from PLM in XML format using XPath/XQuery languages. An abstraction effort and an adapter implementation were required to make PLM content as usable as XML document. Afterward, the resulting XML content could be directly reused or converted to another format.

Keywords: PLM, Product Data Exchange, Data Semantics, XML, XQuery.

1 Introduction

Early engineering systems were composed from a set of tools that didn't have a significant impact in the market [1]. Over the years, companies have been increasingly equipped with CAD systems, and, naturally, adopted systems were different from a company to another. Engineers, who are users of these heterogeneous systems, were paralyzed by the inability of sharing and exchanging data related to their products because each system has a proprietary and incompatible data representation format. To comply with this situation, the main proposed solutions were in form of data exchange standards. First standards were intended to exchange bi-dimensional and tri-dimensional geometric data. Then, the need to exchange went beyond this, since there were several applications that manage and exchange data throughout product lifecycle.

Besides, through several industrial experiences in PLM (Product Lifecycle Management) integration and customization, we found that there is a considerable redundant work from a project to another in particular for the following applications:

- Reorganization of data and products
- Improvement of transfer and data exchange
- Automatic documentation generation

Usually, these applications need to export or to exchange data. In all cases, provided solutions were specific every time and didn't always include exchange standards. Surely, the best way is to use data exchange standards, but experience with them has shown considerable limitations in their integration in commercial tools and their interoperability. Recently, several researches have been turned to XML strategies, either by exchanging data in a native XML format or by providing an XML representation of the exchange standards.

In this paper, we define a new standard approach for querying and exporting data from PLM in XML format. The paper is organized as follows. In Section 2, we present data exchange technologies, their limitations and the new developed approaches. In Section 3, we present our proposed PLMXQuery approach and we discuss its implementation in Section 4. We conclude in Section 5.

2 State of the art

To bring new products to the market faster, with more profit and lower cost, companies need to share and exchange data quickly, automatically and safely [2]. Moreover, according to [3], one of the main PLM goals is to simplify communication and data exchange between collaborators and industrial partners. In fact, data exchange is the building block on which we can find various applications such as data sharing and migration, applications integration, and much more knowledge reuse.

Actually, PLM supports many distributed activities and different systems to create, store and share data related to products. This, according to [4], requires robust mechanisms for data exchange and integration. Since PLM data are handled by heterogeneous systems using proprietary formats, in order to exchange data, the first approach that comes to mind is to implement translators between each pair of systems. Considering the huge number of existing systems, it will be worth to find another way specifying an open, common and neutral data representation: exchange standard.

This principle has given rise to many exchange standards in engineering (i.e. IGES, VDA, SET, STEP, etc.) These standards are supposed to be independent from any system or application providers. Naturally, first standards concerned geometric data handled by CAD/CAM systems. According to [5], CAD standards had significantly reduced translation costs and improved customers and suppliers interactions. Then, normalization was expanded to cover all product aspects throughout its lifecycle [6].

On the one hand, data defined by the geometry has limited number of basic concepts (point, line, curve, surface, etc) which are common to physical product aspects. Thus, international normalization efforts are focused on this kind of data. But on the other hand, it is less conceivable to standardize data related to the product lifecycle because these data are represented using different concepts from a system to another. Some works like [7], [8], [9] and many others tried to propose standards for exchange data related to product lifecycle. Nevertheless, it is still difficult to specify an exchange standard common to all existing systems around the product lifecycle. Even if we try, standardization will be a long and complex process [10].

In general, some standards are mature, such as STEP, but they still suffer from:

- Slow integration in commercial tools [3].
- Lack of interoperability among standards [5], [11].
- Large exchange files and significant processing time [4].
- Solutions are specific to their domains and communities [10], [12].

To overcome some of these limitations, there is strong tendency to use XML as a mean to capture and transform data [11]. For example, to represent STEP files in XML [13], PDML (Product Data Markup Language) is proposed [12]. Another example, PLMXML [9], is used to represent products' lifecycle related data.

Beside of data exchange based on translators and standards, we identify a third approach which is based on application integration. This approach defines a process on which several autonomous, distributed and heterogeneous data sources are integrated into one source and they are associated to a global schema [14]. By using this approach, data can be exchanged smoothly throughout the predefined mappings between every local schema of each source and the global schema. Application integration can be based on databases, data files [2] (in text format (CSV) or standard format (STEP, XML, etc.)), or semantic web ontologies [10], [15], [16]. Obviously, data integration is almost implemented to integrate applications sharing the same data [12]. It should be thought “a priori” and requires a significant investment. Also, this approach is not easily scalable since it’s mandatory to re-integrate different applications if we need to exchange data other than defined at the initial integration.

A new approach has emerged based on remote services invocation. In this context, the OMG has defined PDM Enablers [17], specifying a set of interfaces providing functionalities that give a simple access to PLM data in standard way. Few years after, the OMG has also developed PLM services [18], an advanced and extended version of PDM Enablers based on Web Services technology. In fact, the current trend in PLM market is that several software providers give access to all or a part of their API functions through Web Services.

3 PLMXQuery approach

Regarding the different approaches previously exposed, we can discern that there is always need to a more standard and flexible strategy for exchanging data between PLM applications. In fact, we can benefit from effective aspects of previous experiences.

In this section, we present PLMXQuery, a new developed approach which defines a standard way to access and explore PLM content, in order to exchange data, without resorting to a concise specification of limited set of concepts (the case of exchange standards) or a full integration of applications.

3.1 Definition

The idea of PLMXQuery approach is to make the PLM content seen as XML document. Then, we can get benefit from XML related technologies, in particular XPath and XQuery, to query (browse and export) PLM content in a standard way. Afterward, resulting (exported) XML content could be reused directly or converted to another format. We can easily use this approach to exchange data punctually (i.e. migration) or permanently (i.e. integration). Also, we can export all the content of the PLM or only a thoroughly selected part of its content.

However, to achieve this idea we are facing a modeling challenge which is to make an abstraction effort on PLM content to make its exploitation possible as XML document. Next, we will show how to overcome this challenge using a standard mapping rules between PLM objects and XML elements. After that, we will present XPath and XQuery languages and the way they will work in our context.

3.2 Mapping rules

In order to represent their products through different systems and lifecycle steps, companies need to specify a methodical and functional structures that reflecting data organization depending to each company. For this purpose, every PLM system should implement a data model. The later usually defines basic components: types of objects managed by the PLM system, their properties and relationships. Therefore, a product can be represented inside a PLM system as a set of interconnected objects.

Table 1. Standard mapping rules between PLM objects and XML nodes.

PLM object	XML node
object type	node name
object attributes	node attributes
object relations	children nodes
relation type	child node name + direction attribute
relation attributes	child node attributes
linked objects (by relation)	child node children

Within the PLMXQuery approach, every PLM component is mapped to a corresponding XML element[1] according to the standard mapping rules, summarized in Table 1, which are applicable to any PLM system. The following example is an illustration:

```
<Item id="00013" name="weldpoint">
   <IMAN_reference direction="forward">
      <Text id="00017" name="specs"/>
   </IMAN_reference>
</Item>
```

[1] The following concepts are used alternatively: XML element and XML node.

This is an XML representation of a PLM object of type *Item*, its attributes with associated values, and a directional relation (*IMAN_reference*) from this first object to another object of type *Text*.

3.3 Querying using XPath and XQuery

XPath and XQuery[2] are two standard languages working on XML structures and we'll use them to extract data from a PLM system. XPath is a filtering language based on a context node to designate a portion of an XML document. In reference to the previous XML example, the following one shows how to access the attribute *id* of element of type *Text* related to context node (not always the root).

```
./IMAN_reference/Text/@id
```

XPath could be incorporated inside other XML languages, especially XQuery. The later is an XML query language in which we can declare functions and use conditional and loop statements. The following is an example of a query that uses an XPath expression (text between "(:" and ":)" specifies a comment).

```
xquery version "1.0";
(: Get a reference to the context object :)
declare variable $root := .;
(: Get a set of items associated to the context :)
declare variable $related := $root/IMAN_reference/Text;
(: Beginning of the resulting XML :)
<root>{
(: Process related items one by one :)
for $item in $related
   return <text>{$item/@id}</text>}
</root>
```

By applying this query on the XML example, it will browse the XML content starting from the context node, then filter and list all paths of interest using XPath expression and then, returns the following XML:

```
<root>
   <text>0017</text>
</root>
```

However, XPath and XQuery operate on an abstract data model named XDM[3]. This feature allows the underlying model to be anything other than a structured XML content; PLM content in our case study. Thus, we should implement PLM to XML mapping rules to make the mentioned languages work on PLM through this abstraction. This is what we'll undertake in the following section.

---

[2] http://www.w3.org/TR/xpath/ and http://www.w3.org/TR/xquery/

[3] http://www.w3.org/TR/xpath-datamodel/

4 PLMXQuery implementation

For the implementation of PLMXQuery, we propose an adapter for exporting data from PLM system in XML format. PLMXQuery adapter should be able to consider the PLM content as an XML document and to query it using an XQuery engine.

Usually, when working with XML document the query engine tries to browse the whole document to ensure its conformity (a well-formed XML document). This is inconceivable when XML document will be substituted by a PLM database. This situation leads to an important technical challenge which is to have reasonable performances; filtering PLM content and limiting the browsing to only objects of interest.

Next, we present a standard architecture for the adapter including the solution for the raised challenge and the selected XML platform. Then, we present a specific implementation of the adapter for Teamcenter PLM system.

4.1 Architecture

Since our approach considers PLM content as XML document queryable using XQuery language, the adapter should be able to implement defined mapping rules and incorporate an XQuery engine. A standard architecture for the adapter is proposed in **Error! Reference source not found.**. When we need to query the PLM content according to this architecture, we edit a query and then the adapter will: 1) load and compile the query, 2) execute the query on the PLM content, and 3) return the query results in XML format. In order to perform these different steps, the adapter will translate the PLM browsing and access functions (API) to those expected by the XQuery engine.

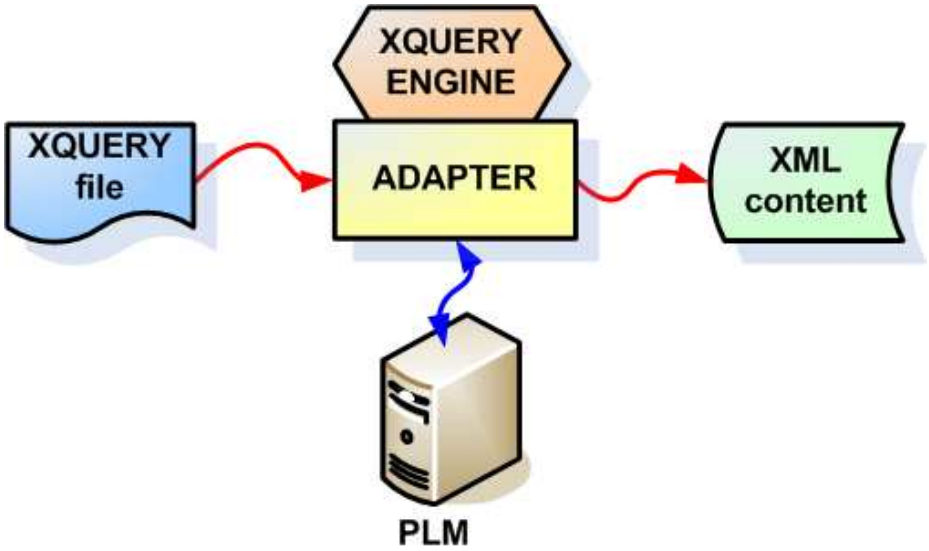

Fig. 1. PLMXQuery architecture.

Browsing limitation. One of the major selection criteria of the XML platform on which we will build the PLMXQuery adapter is to restrict the scope of visited PLM objects. To make this possible, we should be able to define the following functions through the selected platform:

- Access function: allows for quickly finding and selecting PLM objects using some of their properties. Then, a selected object could be used as a context node in the query.
- Filter function: limits browsing only on PLM object types, relation types and attribute names that we are interested in.

Selected platform. There are several existing XML platforms endowed with XQuery engine. To build the adapter, we need to reuse the most appropriate platform that fits the different PLMXQuery requirements expressed by the following major criteria:

- Adaptability: clear/documented implementation of the XDM abstraction model.
- Performance: restriction and limitation of the visited objects' number.
- Extensibility: definition of additional functions adapted to the underlying model.
- Cost: license, free, etc.

The Table 2 illustrates a summary of the compared platforms against these criteria.

Table 2. Comparison of existing XQuery platforms.

Plateform	Adaptability	Performance	Extensibility	Cost
Altova XML suite	×	?	×	×
DataDirect XQuery	✓	?	×	×
Jaxen	✓	×	×	✓
Jaxp	✓	×	×	✓
Xalan	✓	×	×	✓
QtXmlPatterns	✓	✓	×	×
Saxon	✓	✓	✓	✓

In this table, we mention if a platform includes the concerned feature ("✓") or not ("×"), and if there were difficulties to test or to get information on the feature ("?"). So, it is clear from this comparison table that Saxon is the most appropriate platform.

4.2 Teamcenter XQuery Adapter

Teamcenter XQuery is an implementation of PLMXQuery Adapter for Teamcenter PLM system (edited by Siemens PLM Software). In the following we define the mapping rules for this implementation and query extension functionalities.

Mappings. Mapping rules defined earlier are standard and common to all PLM systems. If needed, these rules could be completed by additional mapping rules specific for every PLM system. Table 3 shows additional rules for Teamcenter PLM system needed to map its content with XML.

Table 3. Teamcenter mapping rules.

PLM object	XML node
Revisions (relation)	children nodes + direction attribute
Bill-Of-Material (relation)	children nodes
constraints, conditions, BOM variants	child node's attributes

Extension functionalities. Standard functions provided by XPath/XQuery could be limited to answer specify some queries. The selected XML platform, Saxon[4], allows to define new extension functions that can be reused inside the query allowing more flexibility in query expression. The developed extensions are (with examples):

1. Direct access functions to Teamcenter objects:

```
(: determine where the extensions are defined :)
declare namespace tc= "ja-
va:com.cadesis.plmxquery.teamcenter.TeamCenterXQueryExten
sion";
(: get TC object to use as a context node :)
declare variable $item := tc:getObject("item00001");
```

2. Filtering functions (types, relations, attributes filters):

```
<root typeFilter="{tc:setTypeFilter("Item")}"
      relFilter="{tc:setRelationFilter("Revision,view")}"
      attFilter="{tc:setAttributeFilter("current_id,
      current_name,object_type,constraint, ")}" >
{.}
</root>
```

3. Downloading files function

```
(: Get set of documents associated to the item :)
declare variable $docs := $item/IMAN_reference/Text;
(: Beginning of the resulting XML :)
<root> {
(: Examine all documents :)
for $doc in $docs
   (: We try to get the file using the extension :)
```

[4] http://www.saxonica.com

```
   if (tc:getFile($c, "Text")) then <a>File downloaded</a>
   else <error>Cannot get file from source</error>
} </root>
```

The first and the second extensions are developed especially to enhance the adapter performance by limiting the number of visited nodes. But the third function is just developed to download files from PLM into a temporary folder.

All of the adapter implementation for Teamcenter, including extension functions, is based on a common package that defines all of common abstract classes and functions. This package makes easy to implement another PLM adapter.

5 Conclusion

In this paper we presented the PLMXQuery approach, a new way to extract data from PLM in XML format. The proposed approach considers the PLM content as an XML resource than queries this content using the XML standard query language XQuery. Throughout the implementation of this approach (Teamcenter XQuery adapter) we faced two challenges: 1) defining mapping rules between PLM and XML and 2) restricting number of PLM browsed objects. The developed adapter was based on an XML platform named Saxon, equipped with an XQuery engine. This platform was extensible enough to overcome the raised challenges and to define additional functions used to refine more elaborated queries.

The PLMXQuery approach is very flexible since it doesn't define a limited set of concepts to avoid exchange standards drawbacks and it doesn't recourse to full, complex and slow integration of applications. Using this approach, when we need to have an interface with a PLM system we reuse the standard package to develop its specific adapter. But, for a specific PLM system, only one adapter is developed for several purposes (data export, data exchange, integration, migration, etc) and, for each different application, we just need to write the queries and to configure the query engine (without any modification of the source code).

Teamcenter XQuery adapter is currently under test in a real industrial context and until now we don't have enough results to discuss in this paper.

We believe that we proposed an interesting standard way to access and query PLM content. Using an XML-based approach in a PLM context is very suitable and more advisable since XML is widely used especially in data transfer and communication among heterogeneous applications.

References

1. Goldstein, B., Kemmerer, S., Parks, C.: A Brief History of Early Product Data Exchange Standards, National Institute of Standards and Technology, NISTIR 6221, Gaithersburg, MD, USA (1998)
2. Saaksvuori, A. and Immoen, A.: Product Lifecycle Management, Berlin, Springer-Verlag (2004)

3. Eynard, B.: Gestion du cycle de vie des produits et dynamique des connaissances industrielles en conception intégrée. Report of Habilitation to manage researches. Troyes University of Technology, France (2005)
4. Fenves, S.J., Sriram, R.D., Subrahmanian, E., Rachuri, S.: Product Information Exchange: Practices and Standards. ASME Journal of Computing and Information Science in Engineering 5(3): 238-246 (2005)
5. Rachuri, S., Subrahmanian, E., Bouras, A., Fenves, S.J., Foufou, S., & Sriram., D.R.: The Role of Standards in Product Lifecycle Management Support. In proceedings of the International Conference on Product Lifecycle Management 2006: 122-136 (2006)
6. Srinivasan, V.: Open Standards for PLM. PLM Emerging solutions and challenges, Bouras A. Gurumorthy B., Sudarsan R.. Interscience Enterprise Ltd, pp. 475-484. (2005)
7. Szykman, S., Fenves, S., Keirouz, W., & Shooter, S.: A Foundation for Interoperability in Next-Generation Product Development Systems. Computer-Aided Design 33 (7): 545-559 (2001)
8. Lee, C., Lau, H., Yu, K.M., & Ip, W.H.: A generic model to support rapid product development: an XML schema approach. International Journal of Product Development 1(3): 323-340 (2005)
9. Siemens PLM Software: Open product lifecycle data sharing using XML. White Paper (2011)
10. Lin, H.K., & Harding, J.A.: A Manufacturing System Engineering Web Ontology Model on the Semantic Web for Inter-enterprise Collaboration. Computers in Industry 58(5): 428-437 (2007)
11. Pratt, M.J.: Introduction to ISO 10303-the STEP Standard for Product Data Exchange. Journal of Computing and Information Science in Engineering 1(1): 102-103 (2001)
12. Burkett, W.C.: Product data markup language: a new paradigm for product data exchange and integration. Computer-Aided Design 33 (7): 489-500 (2001)
13. Peak, R., Lubell, J., Srinivasan, V. & Waterburry, S.: STEP, XML, and UML: Complementary Technologies, ASME Journal of Computing and Information Science in Engineering 4(4): 379-390 (2004)
14. Bellatreche, L., Xuan, D.N., Pierra, G. & Dehainsala, H.: Contribution of Ontology-based Data Modeling to Automatic Integration of Electronic Catalogues within Engineering Databases, Computers in Industry Journal 57 (8-9): 711-724 (2006)
15. Hefke, M., Szulman, P. & Trifu, A.: A Methodological Approach for Constructing Ontology-Based Reference Models in Digital Production Engineering. Proceedings of the 6th International Conference on Knowledge Management, Graz, Austria (2005)
16. Sriti, M-F, Eynard, B., Boutinaud, P., Matta, N., Zacklad, M.: Towards a semantic-based platform to improve knowledge management in collaborative product development. International Product Development Management Conference, Milan, Italy: 1381-1394 (2006)
17. OMG: Product Data Management Enablers. Object Management Group standard specification (2000)
18. OMG: Product Lifecycle Management Services. Object Management Group standard specification (2005)