An XML-Based Macro Data Representation for a Parametric CAD Model Exchange

Jeongsam Yang[1], Soonhung Han[2], Joonmyun Cho[3], Byungchul Kim[4] and Hyun Yup Lee[5]

[1]PartDB Co. Ltd, jsyang@partdb.com
[2]Korea Advanced Institute of Science & Technology, shhan@kaist.ac.kr
[3]Korea Advanced Institute of Science & Technology, cjmyun@icad.kaist.ac.kr
[4]Korea Advanced Institute of Science & Technology, mir7942@icad.kaist.ac.kr
[5]Chungnam National University, hylee@cnu.ac.kr

ABSTRACT

A macro-parametric approach, which is a history-based method of parametric CAD model exchange, has recently been proposed. CAD models can be exchanged in the form of a macro file that comprises a sequence of modeling commands. As a set of event-driven commands, a standard macro file can transfer the designer intent such as parameters, features and constraints. Moreover, it is suitable for a network environment because standard macro commands are open and explicit, and the data size is small. This paper introduces XML technology to represent the macro-parametric exchange. Using XML to represent macro-parametric commands enables the management of a large amount of dynamic content, Web-enabled distributed applications, and the inherent characteristics of structure and validation.

Keywords: CAD model exchange; design intent; macro-parametric; XML

1. INTRODUCTION

During the last ten years, the manufacturers of the world invested about a trillion dollars to convert or generate their products as digital models [1]. The significant assets of manufacturers now include CAD drafting, three-dimensional solid models, assemblies, bills of material, and engineering models for analysis and simulation. Although the current trend calls for a digital enterprise that integrates C3PE (CAD, CAM, CAE, PDM and ERP), manufacturers have difficulty in reusing existing product models because of the data loss during the analogue to digital conversion. The RTI estimates that the interoperability problem cost the members of the US automotive supply chain at least a billion dollars in 1999 [2].

There are two major approaches for information sharing between different C3PE systems. One approach involves a static interface of data exchange that translates models through a neutral file such as the Standard for Exchange of Product Model Data (STEP) and the Initial Graphics Exchange Specification (IGES); in this approach, a snapshot of the model is exchanged. The other approach is to implement a dynamic interface using a standardizing application program interface. The application interface specification of the Consortium for Advanced Manufacturing International provides a standard programming interface that interfaces user applications with CAD/CAM systems [3].

There are three methods for exchanging static CAD models. The first uses a single CAD system throughout the various disciplines of a company, without the need for model exchange. However, in practice each discipline needs a specialized CAD system, and companies are heavily dependent on a single commercial CAD system. In addition, the introduction of a single CAD system is unsuitable for an extended enterprise, especially in a distributed development environment that requires the participation of various organizations with heterogeneous CAD systems.

The second method uses direct translators between different CAD systems. However, because the geometric modeling kernels within CAD systems differ, data loss and shape distortion occur during data translation. Furthermore, costs increase because an enterprise that uses N different CAD systems needs N(N-1)/2 translators.

The third method uses a neutral format such as STEP (ISO-10303), IGEG (ANSI), VDA-FS, or DXF. A preprocessor generates the neutral file from the native format. A postprocessor receives the neutral file and converts it into the native format of the receiving CAD system. Although this method needs two translations and has twice the data loss, it is the recommended method for most cases. One remaining problem of the neutral format is that it does not retain parametric information such as the designer intent but only the pure boundary representation of the CAD model. The boundary representation model without parametric information presents difficulties for engineering changes in a collaborative design process or in a configuration design process; it also leads to an unexpected distortion of shape.

To solve this problem, a macro-parametric method has recently been proposed [4,5,6,7]. Incorporating CAD model exchange based on design history, the macro-parametric method exchanges the parametric information of a CAD model as a design command sequence between heterogeneous CAD systems. Based on the macro-parametric methodology, this paper proposes an exchange method that uses extensible mark-up language (XML) technology to express a set of standard modeling commands extracted from a CAD model. This set of commands can be shared with other CAD systems on the Internet.

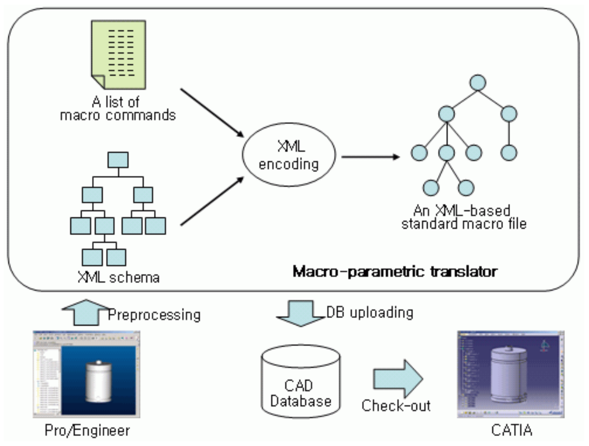


Fig. 1. An overview for XML-based macro-parametric representation

As shown in Fig. 1, a three-dimensional model shape designed with Pro/Engineer was translated into a set of the standard modeling commands in the form of an XML data structure. The translated result was uploaded to a database where it could be translated into other CAD systems such as CATIA, SolidWorks or Unigraphics. Unlike commercial CAD translators that exchange geometry and topology using a CAD kernel, the proposed method extracts aspects of the design intent such as constraints, parameters, design history information, and features from a CAD model. To understand the semantics of the extracted information, the method then uses XML to construct a vocabulary of standard modeling commands.

2. RELATED WORKS

2.1 Exchange of Parametric Information

To define CAD models, Part 42 of the international standard STEP ISO 10303 defines shape entities. However, because it does not define expressions for parameters, constraints and features, it cannot translate the design parameters of a CAD model. Although Part 42 narrowly defines entities for constructive solid geometry, the dimensions of such entities cannot be parameterized [8]. Moreover, it contains no definitions for modeling functions based on design history, such as shelling, chamfering and drafting, though such definitions are usually available in commercial CAD systems.

A project called Enabling Next Generation Mechanical Design (ENGEN) proposed an ENGEN data model, which is a product data model with parameters, features, design history, and constraints based on Part 42 of STEP [9]. The purpose of the ENGEN project is to verify the exchange capability of design intent, which is represented by design parameters, constraints and features. The exchange experiments use the CAD systems of Pro/Engineer (PTC), I-DEAS (SDRC), and CADDS5 (CV). Because the main focus of the ENGEN project is to enable the exchange of models with constraints, the ENGEN data model has insufficient entities for the design history, and it cannot be used to represent a history-based parametric model.

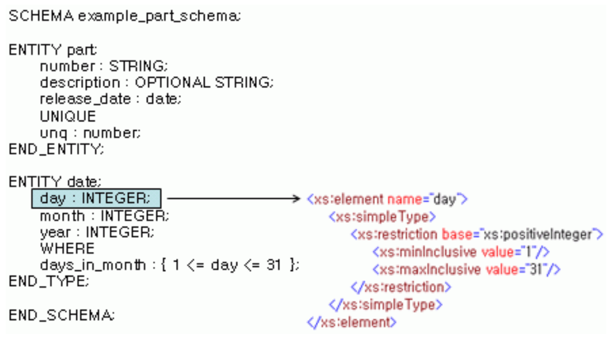
Varra and Anderson have suggested a solid model construction history schema for exchanging parameters, constraints, features and design history information [10]. The schema includes an implicit entity and structure for the design process, which can exchange geometrically restricted solids and the history-based model, along with parameterized features.

2.2 XML Initiatives for Product Data

With the increasing popularity of XML on the distributed environment, mapping CAD model data into XML seems a logical way to make CAD models more accessible through the Web. Examples of previous XML initiatives include the following: materials property data markup language (MatML), metal XML, product data markup language (PDML), product definition exchange (PDX), and universal commerce language and protocol (UCLP) [11].

The use of XML for exchanging CAD models has been studied by Rezayat, who converted a CAD model into an XML-based knowledge base called knowledge-based product development [12]. He proposed a DTD schema

called CAD markup language that can express CAD models in terms of limited boundary representation.
To make use of XML, the working group ISO/TC184/SC4/WG 11 recently developed implementation methods for STEP ISO 10303-28 titled Implementation methods: XML representation of EXPRESS schemas and data. These methods enable XML to be used as a representation mechanism for STEP-conformant schema and instance data [13,14].
Although XML schema is not as powerful as EXPRESS, XML has potential for economy of scale because the tools for XML technology will flourish on account of the large number of users. Implementing all the details of the specification of EXPRESS to ensure robust information systems is difficult. One example is the EXPRESS schema's rule-based constraint. As shown in Fig. 2, the EXPRESS schema can only use the command WHERE to express constraints. The XML schema, however, offers user-defined types of data and 44 keywords such as Element options, Attribute values and Facet to describe various types of constraints.



```
SCHEMA example_part_schema;

ENTITY part;
    number : STRING;
    description : OPTIONAL STRING;
    release_date : date;
    UNIQUE
    unq : number;
END_ENTITY;

ENTITY date;
    day : INTEGER;
    month : INTEGER;
    year : INTEGER;
    WHERE
    days_in_month : { 1 <= day <= 31 };
END_TYPE;

END_SCHEMA;
```

```
<xs:element name="day">
    <xs:simpleType>
        <xs:restriction base="xs:positiveInteger">
            <xs:minInclusive value="1"/>
            <xs:maxInclusive value="31"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Fig. 2. Comparison of EXPRESS schema with XML schema

3. MACRO-PARAMETRIC METHODOLOGY

3.1 The Macro-parametric Approach

The macro-parametric approach is one of the history-based parametric methods that learn from a database recovery when a transaction log file is used for a crash recovery. A transaction log file contains consecutive SQL commands. To transfer a parametric CAD model that includes the design history, a set of standard modeling commands is defined and used as a neutral format. A macro file records the modeling command sequence issued during a session. It implicitly contains the design intent, which is represented by constraints, parameters, design history and features. The history of user commands is recorded in a macro file, and the macro file is used for a static model exchange. If these commands are translated into the commands used in other CAD system, the macro file regenerates the parametric model inside the receiving CAD system.
Figure 3 shows the data exchange model used in the macro-parametric approach. Mapping in the macro-parametric approach comprises two levels: the schema mapping between the user command set of a CAD system and a standard command set; and the actual data translation between the macro file of a commercial CAD system and a standard macro file. To translate data models between CAD systems, the macro file generated by a commercial CAD system is translated into a standard macro file, which is again translated into a macro file of the receiving CAD system.

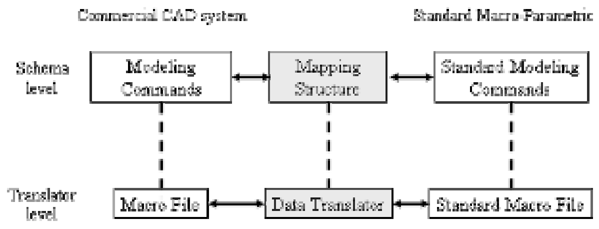


Fig. 3. Concept of the macro-parametric mapping

3.2 A Set of Standard Modeling Commands

To classify the modeling commands of commercial CAD systems, we analyzed the common denominators of user commands used in the following five commercial CAD systems: CATIA, Unigraphics, SolidWorks, Pro/Engineer and IDEAS. Using XML schema, we also defined a set of 144 standard modeling commands: 57 sketch commands, 40 solid modeling commands, 23 surface modeling commands and 24 constraint commands (see Appendix A and B). A group of sketch commands was submitted as a new work item to ISO/TC/184/SC4/WG12, and progress has been made in international standardization.

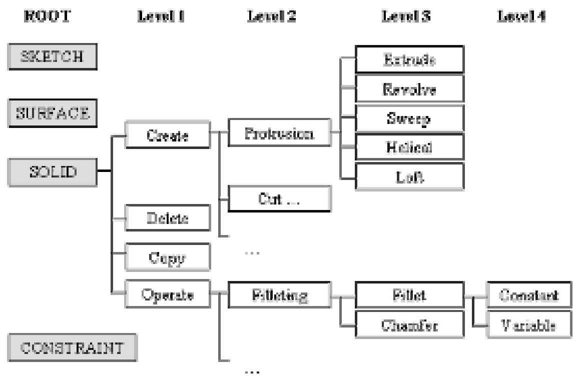
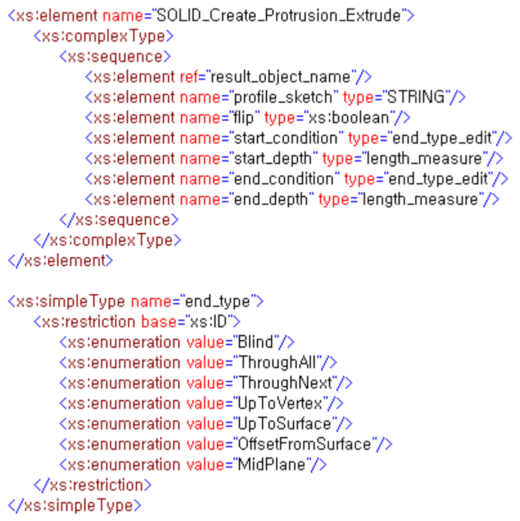


Fig. 4. Classification of standard modeling commands

As shown in Fig. 4, the set of standard modeling commands comprises four groups at the root level—namely *SKETCH*, *SOLID*, *SURFACE* and *CONSTRAINT*. The modeling commands are further classified into four levels of concrete action.

A standard modeling command is made of consecutive classification names; for example, the linear protrusion command is *Solid_Create_Protrusion_Extrude*. Its XML schema representation is shown in Fig. 5. In the schema, *result_object_name* is the name of a protrusion feature created by the command operation, profile-sketch is the 2D profile of protrusion, and *flip* indicates the protrusion direction. The terms *start_condition* and *end_condition* represent the beginning condition and the end condition, while *start_depth* and *end_depth* represent the length of the protrusion. The shape generated by the command *Solid_Create_Protrusion_Extrude* is shown in Fig. 6.



```
<xs:element name="SOLID_Create_Protrusion_Extrude">
   <xs:complexType>
      <xs:sequence>
         <xs:element ref="result_object_name"/>
         <xs:element name="profile_sketch" type="STRING"/>
         <xs:element name="flip" type="xs:boolean"/>
         <xs:element name="start_condition" type="end_type_edit"/>
         <xs:element name="start_depth" type="length_measure"/>
         <xs:element name="end_condition" type="end_type_edit"/>
         <xs:element name="end_depth" type="length_measure"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>

<xs:simpleType name="end_type">
   <xs:restriction base="xs:ID">
      <xs:enumeration value="Blind"/>
      <xs:enumeration value="ThroughAll"/>
      <xs:enumeration value="ThroughNext"/>
      <xs:enumeration value="UpToVertex"/>
      <xs:enumeration value="UpToSurface"/>
      <xs:enumeration value="OffsetFromSurface"/>
      <xs:enumeration value="MidPlane"/>
   </xs:restriction>
</xs:simpleType>
```

Fig. 5. XML schema representation of *Solid_Create_Protrusion_Extrude*

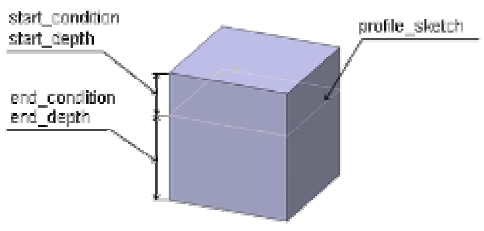


Fig. 6. The shape generated by the command *Solid_Create_Protrusion_Extrude*

4. XML-BASED DEFINITION OF MODELING COMMANDS AND EXPERIMENTS

4.1 XML Representation of Standard Modeling Commands

XML separates semantics for context from syntax for data. The structure of standard modeling commands, which is described using XML schema definition language, offers a dynamic and extensible information skeleton to design the history and syntax.

Using XML for standard modeling commands has two strong points. Firstly, XML is a widely accepted format for data exchange and is suitable for the open data architecture of application software. Closed architecture is inconvenient for users. CAD models, for instance, last for decades, whereas most CAD systems seldom last more than a few years. Because of the potential risk, users demand the deployment of open standards. Moreover, XML can synchronize the speed of translator development and the standardization of modeling commands using some type of conformance testing such as validation checking.

Secondly, the Web-based CAD data exchange (DEX) service system can be easily implemented. The DEX service can exchange CAD model data as an XML message stream between heterogeneous CAD systems of a distributed environment (Fig. 7). If two different DEX systems adopt XML, loosely coupled, the DEX service is not subordinate to the specific language or the component model, and it can be applied to any operating system. To develop an application for a distributed environment without XML, the tight coupling required by the two systems imposes a significant amount of customization overhead. To enable communication, an in-depth understanding of both systems is required. In addition, communication between tightly coupled systems requires a system-level interface and a package oriented to a commercial system such as RPC (remote procedure call), RMI (remote method innovation), CORBA (common object request broker architecture), or DCOM (Microsoft distributed component object model).

Figure 7 shows a conceptual Web-based DEX service that adopts the macro-parametric method and XML technology. The macro-parametric organization that standardizes modeling commands deploys a macro-parametric description and schema in DEX service sites; it also uses macro-parametric translator developers through the HTTP protocol. A third-party macro-parametric translator that conforms an up-to-date macro-parametric description has been released. Each DEX system provides a CAD model exchange service to C3PE systems and local site users through the HTTP protocol.

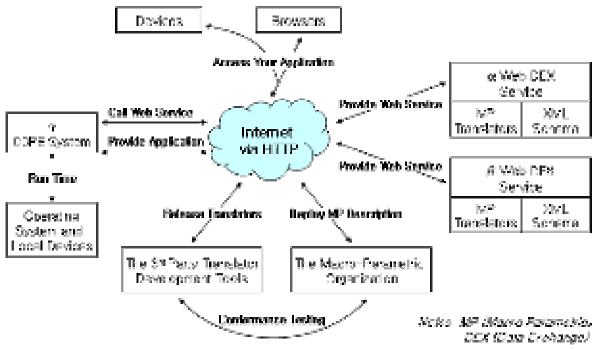


Fig. 7. Overview of the Web-based DEX service

4.2 XML Schema Design

As shown in Table 1, standard modeling commands are classified as core commands (CC) and non-core commands (NCC) based on the frequency of modeling use (see section 3.2 and Appendices).

	No. of CC	*No. of NCC*	*Sum*
Sketch	30	27	57
Surface	17	6	23
Solid	22	18	40
Constraint	21	3	24
	90	54	144

Tab. 1. Classification of standard modeling commands

XML schema were designed for 90 of the standard core commands (see Appendix A). The schema comprise three layers. The first layer, *MACRO_PARAMETRICS*, is the root element that shows the document conforms to the macro-parametric methodology. As shown in Fig. 8, the root element comprises four groups: *SKETCH_COMMANDS*, *SOLID_COMMANDS*, *SURFACE_COMMANDS* and *CONTRAINT_COMMANDS*.

Each of the four groups or the root element is used as a base element for subsidiary elements of standard modeling commands. The second layer is made up of command elements that are actually performed for modeling. The command elements comprise 30 sketch commands, 22 solid commands, 17 curve commands and 21 constraints. These commands can be recorded selectively within the root element *MACRO_PARAMETRICS*. The third layer comprises elements that define the parameters of each command. These elements define the data type according to the command parameters, thereby enabling XML validation. The XML schema of Fig. 8 can represent the design history by connecting the commands subordinately.

The *SOLID_Create_Protrusion_Revolve* command, which is enclosed in the dashed box of Fig. 8, revolves around a sketch and creates a solid as shown in Fig. 9. Figure 10 defines the XML schema structure of *Solid_Create_Protrusion_Revolve*. The structure is composed of the following five elements:

- *profile_sketch* represents the sectional shape of the revolution and the sketch plane, including the revolving axis.
- *flip* represents the revolution direction of the Boolean type.
- *start_angle* and *end_angle* indicate the beginning angle and the last angle of the revolution.
- *result_object_name* of the STRING type represents the name of the object generated by the revolution.

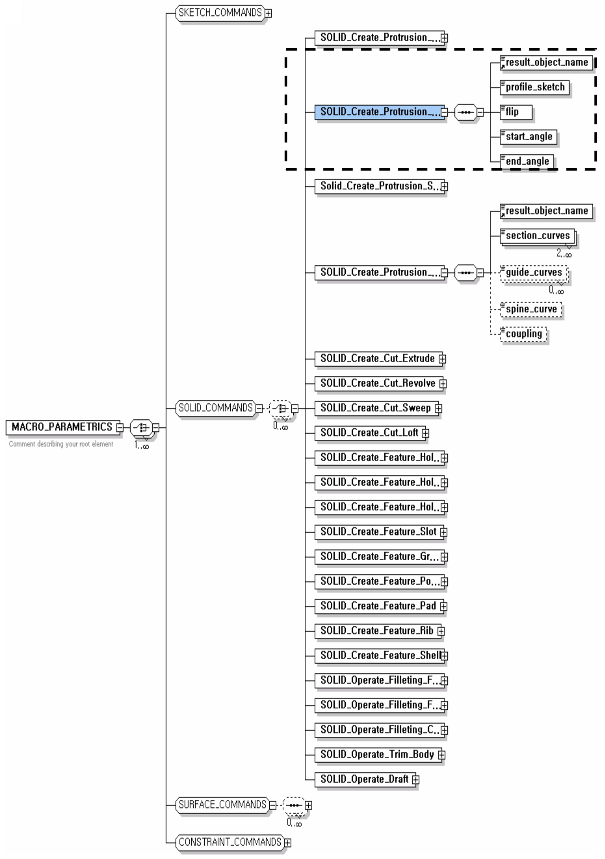

SKETCH_COMMANDS
SOLID_Create_Protrusion_...
SOLID_Create_Protrusion_...
result_object_name
profile_sketch
flip
start_angle
end_angle
Solid_Create_Protrusion_S...
SOLID_Create_Protrusion_...
result_object_name
section_curves
2..∞
guide_curves
0..∞
spine_curve
coupling
SOLID_Create_Cut_Extrude
SOLID_Create_Cut_Revolve
SOLID_Create_Cut_Sweep
SOLID_Create_Cut_Loft
SOLID_Create_Feature_Hol...
SOLID_Create_Feature_Hol...
SOLID_Create_Feature_Hol...
SOLID_Create_Feature_Slot
SOLID_Create_Feature_Gr...
SOLID_Create_Feature_Po...
SOLID_Create_Feature_Pad
SOLID_Create_Feature_Rib
SOLID_Create_Feature_Shell
SOLID_Operate_Filleting_F...
SOLID_Operate_Filleting_F...
SOLID_Operate_Filleting_C...
SOLID_Operate_Trim_Body
SOLID_Operate_Draft
MACRO_PARAMETRICS
Comment describing your root element
1..∞
SOLID_COMMANDS
0..∞
SURFACE_COMMANDS
0..∞
CONSTRAINT_COMMANDS

Fig. 8. The XML schema structure designed for the standard modeling commands

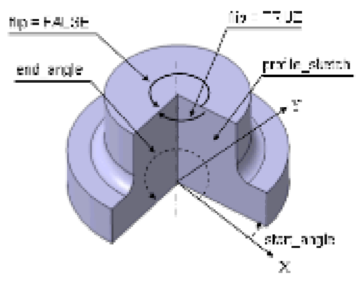
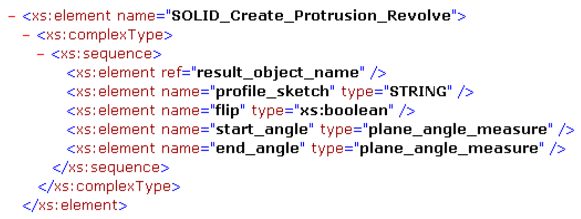
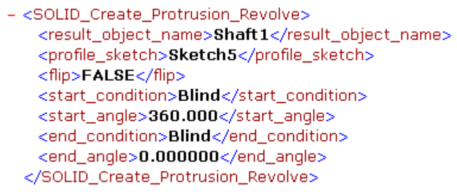


Fig. 9. Parameters of *SOLID_Create_Protrusion_Revolve*



```
- <xs:element name="SOLID_Create_Protrusion_Revolve">
  - <xs:complexType>
    - <xs:sequence>
        <xs:element ref="result_object_name" />
        <xs:element name="profile_sketch" type="STRING" />
        <xs:element name="flip" type="xs:boolean" />
        <xs:element name="start_angle" type="plane_angle_measure" />
        <xs:element name="end_angle" type="plane_angle_measure" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

(a) Schema document



```
- <SOLID_Create_Protrusion_Revolve>
    <result_object_name>Shaft1</result_object_name>
    <profile_sketch>Sketch5</profile_sketch>
    <flip>FALSE</flip>
    <start_condition>Blind</start_condition>
    <start_angle>360.000</start_angle>
    <end_condition>Blind</end_condition>
    <end_angle>0.000000</end_angle>
  </SOLID_Create_Protrusion_Revolve>
```

(b) Instance document

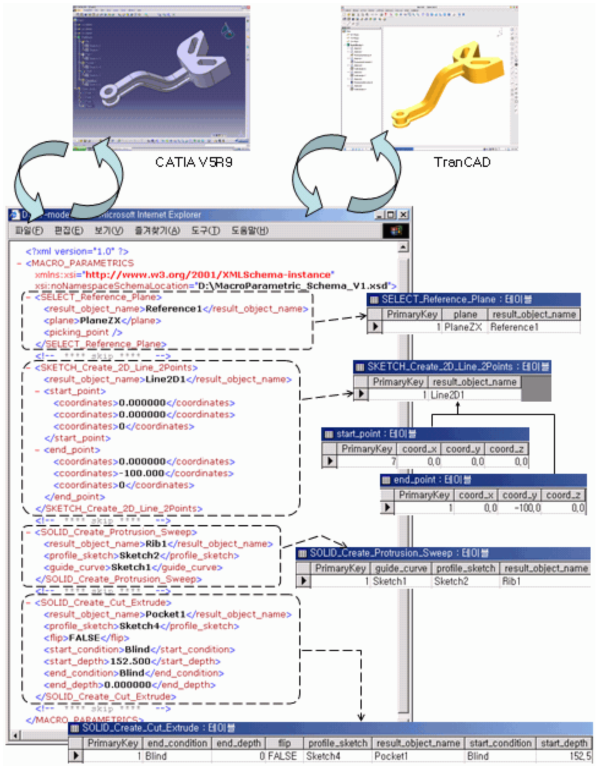
Fig. 10. XML schema and instance of *SOLID_Create_Protrusion_Revolve*

4.3 A Data Exchange Experiment

A data exchange experiment based on XML schema of standard modeling commands has been accomplished. Comprising 54 standard modeling commands, the Y-model was exchanged between our own TransCAD and the four commercial CAD systems Pro/Engineer, CATIA V5, SolidWorks and Unigraphics.

Figure 11 shows the following selection of commands from among the 54 commands of the Y-model:

- *SELECT_Reference_Plane*
- *SKETCH_Create_2D_Line_2Points*
- *SOLID_Create_Protrusion_sweep*
- *SOLID_Create_Cut_Extrude.*



```
<?xml version="1.0" ?>
- <MACRO_PARAMETRICS
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="D:\MacroParametric_Schema_V1.xsd">
  - <SELECT_Reference_Plane>
      <result_object_name>Reference1</result_object_name>
      <plane>PlaneZX</plane>
      <picking_point />
    </SELECT_Reference_Plane>
    <!-- **** skip ****   -->
  - <SKETCH_Create_2D_Line_2Points>
      <result_object_name>Line2D1</result_object_name>
    - <start_point>
        <coordinates>0.000000</coordinates>
        <coordinates>0.000000</coordinates>
        <coordinates>0</coordinates>
      </start_point>
    - <end_point>
        <coordinates>0.000000</coordinates>
        <coordinates>-100.000</coordinates>
        <coordinates>0</coordinates>
      </end_point>
    </SKETCH_Create_2D_Line_2Points>
    <!-- **** skip ****   -->
  - <SOLID_Create_Protrusion_Sweep>
      <result_object_name>Rib1</result_object_name>
      <profile_sketch>Sketch2</profile_sketch>
      <guide_curve>Sketch1</guide_curve>
    </SOLID_Create_Protrusion_Sweep>
    <!-- **** skip ****   -->
  - <SOLID_Create_Cut_Extrude>
      <result_object_name>Pocket1</result_object_name>
      <profile_sketch>Sketch4</profile_sketch>
      <flip>FALSE</flip>
      <start_condition>Blind</start_condition>
      <start_depth>152.500</start_depth>
      <end_condition>Blind</end_condition>
      <end_depth>0.000000</end_depth>
    </SOLID_Create_Cut_Extrude>
    <!-- **** skip ****   -->
  </MACRO_PARAMETRICS>
```

SELECT_Reference_Plane : 테이블

PrimaryKey	plane	result_object_name
1	PlaneZX	Reference1

SKETCH_Create_2D_Line_2Points : 테이블

PrimaryKey	result_object_name
1	Line2D1

start_point : 테이블

PrimaryKey	coord_x	coord_y	coord_z
7	0,0	0,0	0,0

end_point : 테이블

PrimaryKey	coord_x	coord_y	coord_z
1	0,0	-100,0	0,0

SOLID_Create_Protrusion_Sweep : 테이블

PrimaryKey	guide_curve	profile_sketch	result_object_name
1	Sketch1	Sketch2	Rib1

SOLID_Create_Cut_Extrude : 테이블

PrimaryKey	end_condition	end_depth	flip	profile_sketch	result_object_name	start_condition	start_depth
1	Blind	0	FALSE	Sketch4	Pocket1	Blind	152,5

Fig. 11. CAD model exchange experiment using an XML based macro fil

The Y-model's macro file, which was translated from CATIA V9, was created according to the XML schema. It contains the following commands:

- *SELECT_Reference_Plane* generates a two-dimensional sketch plane.
- *SKETCH_Create_2D_Line_2Points* defines a straight line between two points in the sketch plane.
- *SOLID_Create_Protrusion_Sweep* protrudes the sketch according to a given path.
- *SOLID_Create_Cut_Extrude* cuts the section along a flat plane.

The translated commands and parameters have been stored in a database designed on the basis of the XML schema. The commands are shown in database tables of the lower right corner of Fig. 11. Accessing the modeling history in a CAD model is difficult in conventional CAD systems with file-based storage structures unless a user activates the modeling history in a viewing window. However, a database that uses standard modeling commands is constructed as a user command unit of the modeling history. Such a database can provide concurrent access, data management at the logical level, external reference, or change propagation [15].

Based on macro-parametric methodology, TransCAD is a stand alone CAD modeler that allows validation of XML schema and data interface with four commercial CAD systems by means of an XML macro file.

5. CONCLUSION

An important business objective for the effective application of product data is the mantra "create once, reuse many times." However, CAD model exchange between different C3PE systems causes shape distortion, and the loss of parametric information complicates engineering change. Consequently, in many companies the reuse of CAD models remains a serious problem.

The macro-parametric approach, which is an offshoot of the history-based parametric method, enables parametric information to be transferred as a sequence of standard modeling commands. To represent a set of standard modeling commands in a macro file, we have used XML technology. XML representation of standard modeling commands, which emphasizes the information representation and meaning, provides open data architecture for translator developers and standardization communities. It is suitable for constructing a Web-based DEX service that communicates with two loosely coupled systems through the HTTP standard protocol.

We have used an XML macro file to exchange a CAD model between our own TransCAD and the four commercial CAD systems Pro/Engineer, CATIA V5, SolidWorks and Unigraphics. In addition, we have proposed a pilot database system that uses MS-ACCESS to store the standard modeling command sequence.

Future research is needed on the implementation of non-core commands, the extension of standard modeling commands for complex models and harmonization with the existing STEP Application Protocols of the ISO.

6. REFERENCES

[1] Crabb, H. C., *The virtual engineer: 21st century product development*, SME/ASME Press, New York, NY, 1998.

[2] Gregory, T., *Interoperability cost analysis of the U.S. automotive supply chain – Final report: RTI Project Number 7007-03*, Research Triangle Institute, 1999.

[3] Application Interface Specification (AIS) Version 2.1: Technical Report R-94-PM-01, *Consortium for Advanced Manufacturing International Inc.*, Bedford, TX, 1994.

[4] Choi, K., Mun, D.-H and Han, S., Exchange of CAD part models based on the macro-parametric approach, *International Journal of CAD/CAM (www.ijcc.org)*, Vol. 2, 2002, pp 23-31.

[5] Mun, D.-H, Kim, B. and Han, S., A hybrid parametric translator using the feature tree and the macro file (in Korean), *Transactions of the Society of CAD/CAM Engineers*, Vol. 7, 2002, pp 240-247.

[6] Kim, B.-C., Verification of the standard modeling commands set by developing a geometric modeler (in Korean), Master's thesis, Korea Advanced Institute of Sci. and Tech., 2002.

[7] Mun, D.-H, Han, S. and Oh, Y.-C., A set of standard modeling commands for the history-based parametric approach, *Computer Aided Design*, Vol. 35, 2003, pp 1171-1179.

[8] Pratt, M. J, Extension of the Standard ISO10303 (STEP) for the exchange of parametric and variational CAD models, in *Proceedings of the Tenth International IFIP WG 5.2/5.3 Conference PROLAMAT98*, 1998.

[9] Anderson, B., ENGEN Data Model: a neutral model to capture design intent, in *Proceedings of the Tenth International IFIP WG 5.2/5.3 Conference PROLAMAT98*, 1998.

[10] Barra, R. and Anderson, B., Draft implementor's guide solid model construction history: DRAFT Minutes of WG12 Parametrics Meeting, October 2000.

[11] Kerer, C., Kirda, E. and Kruegel, C., XGuide - A practical guide to XML-based Web engineering, *Lecture Notes in Computer Science*, No. 2376, 2002, pp 104-117.

[12] Rezayat, M., Knowledge-based product development using XML and KCs, *Computer Aided Design*, Vol. 32, 2002, pp 299-309.

[13] Product data representation and exchange: Implementation methods: XML schema governed representation of EXPRESS schema governed data

- ISO/WD 10303-28e2, ISO TC184/SC4/WG11 N202 (http://www.tc184-sc4.org/SC4_Open/SC4_and_Working_Groups/WG11/), 2002.

[14] STEP & XML White Paper: Technologies for Digital Product Data Integration, European Marine STEP Association (EMSA) (http://emsa.germanlloyd.org), 2001.

[15] Kim, J. and Han, S., Encapsulation of geometric functions for ship structural CAD using a STEP database as native storage, *Computer Aided Design*, Vol. 35, 2003, pp 1161-1170.

[16] Han, S. et al., Standard Modeling Commands Set: 2003-09, iCAD Lab. KAIST, 2003.

[17] Yang, J., Han, S. and Park, S., A Method for Verification of CAD Model Quality, *Journal of Engineering Design*, Vol. 15, No. 4, 2004.

[18] Yang, J., Goltz, M. and Han, S., Parameter-Based Engineering Changes for a Distributed Engineering Environment, *Concurrent Engineering: Research and Applications*, Vol. 12, No. 2, 2004.

Appendix A: Core Commands List

	Commands list	
Sketch commands	SKETCH_Create_2D_Line_2Points	SKETCH_Create_2D_Ellipse_CenterPoint
	SKETCH_Create_2D_Polyline	SKETCH_Create_2D_Spline
	SKETCH_Create_2D_Centerline	SKETCH_Create_2D_Conic
	SKETCH_Create_2D_Rectangle	SKETCH_Create_2D_Text
	SKETCH_Create_2D_Point	SKETCH_Create_2D_AxisPoint
	SKETCH_Create_2D_Arc_Concentric	SKETCH_Create_SketchName
	SKETCH_Create_2D_Arc_3Tangents	SKETCH_Operate_Transform_Move
	SKETCH_ Create_2D_Arc_CenterEnds	SKETCH_Operate_Transform_Rotate
	SKETCH_Create_2D_Arc_3Points	SKETCH_Operate_Transform_Mirror
	SKETCH_Create_2D_Circle_CenterPoint	SKETCH_Operate_Transform_Scale
	SKETCH_Create_2D_Circle_Concentric	SKETCH_Operate_Offset
	SKETCH_Create_2D_Circle_3Tangents	SKETCH_Operate_Fillet
	SKETCH_Create_2D_Circle_3Points	SKETCH_Operate_Chamfer
	SKETCH_Create_2D_Arc_Angles	SKETCH_Open
	SKETCH_Create_2D_Ellipse_3Points	SKETCH_Close
Surface commands	SURFACE_Create_ThroughPointFromPoles	SURFACE_Operate_Split
	SURFACE_Create_FromPointCloud	SURFACE_Operate_Merge
	SURFACE_Create_Sweep	SURFACE_Operate_MidSurface
	SURFACE_Create_Section	SURFACE_Operate_Trim
	SURFACE_Create_Offset	SURFACE_Operate_Extension
	SURFACE_Create_Offset_DraftOffset	SURFACE_Operate_Filleting_Fillet_Constant
	SURFACE_Create_Extrude	SURFACE_Operate_Filleting_Fillet_Variable
	SURFACE_Create_Revolve	SURFACE_Operate_Filleting_Chamfer
	SURFACE_Create_Plane_3Points	
Solid commands	SOLID_Create_Protrusion_Extrude	SOLID_Create_Feature_Slot
	SOLID_Create_Protrusion_Revolve	SOLID_Create_Feature_Groove
	SOLID_Create_Protrusion_Sweep	SOLID_Create_Feature_Pocket
	SOLID_Create_Protrusion_Loft	SOLID_Create_Feature_Pad
	SOLID_Create_Cut_Extrude	SOLID_Create_Feature_Rib
	SOLID_Create_Cut_Revolve	SOLID_Create_Feature_Shell
	SOLID_Create_Cut_Sweep	SOLID_Operate_Filleting_Fillet_Constant
	SOLID_Create_Cut_Loft	SOLID_Operate_Filleting_Fillet_Variable
	SOLID_Create_Feature_Hole_Linear	SOLID_Operate_Filleting_Chamfer
	SOLID_Create_Feature_Hole_Counterbore	SOLID_Operate_Trim_Body
	SOLID_Create_Feature_Hole_Countersunk	SOLID_Operate_Draft
Constraint commands	CONSTRAINTS_Create_Dimension_Horizontal	CONSTRAINTS_Create_Constraint_Concentric
	CONSTRAINTS_Create_Dimension_Vertical	CONSTRAINTS_Create_Constraint_Horizontal
	CONSTRAINTS_Create_Dimension_Arbitrary	CONSTRAINTS_Create_Constraint_Vertical
	CONSTRAINTS_Create_Dimension_Circular	CONSTRAINTS_Create_Constraint_Symmetric
	CONSTRAINTS_Create_3DReference_CoordSys	CONSTRAINTS_Create_3DReference_Plane
	CONSTRAINTS_Create_Constraint_Perpendicular	CONSTRAINTS_Create_3DReference_Axis
	CONSTRAINTS_Create_Constraint_Tangent	CONSTRAINTS_Create_3DReference_Curve
	CONSTRAINTS_Create_Constraint_Coincident_SamePoints	CONSTRAINTS_Create_Constraint_Parallel
		CONSTRAINTS_Create_3DReference_Point
	CONSTRAINTS_Create_Constraint_Coincident_Collinear	SELECT_Reference_Plane
		SELECT_Object
	CONSTRAINTS_Create_3DReference_OffsetPlanes	

Appendix B: Non-Core Commands List

	Commands list	
Sketch	SKETCH_Create_3D_Line SKETCH_Create_3D_Point SKETCH_Create_3D_Arc_3Points SKETCH_Create_3D_Circle SKETCH_Create_3D_Ellipse SKETCH_Create_3D_Curve_IntersectionCurve SKETCH_Create_3D_Curve_ProjectionCurve SKETCH_Create_3D_Curve_Spline SKETCH_Create_3D_Spline SKETCH_Create_3D_PointSet SKETCH_Create_3D_Helix SKETCH_Operate_Pattern_Rectangular SKETCH_Operate_Pattern_Circular SKETCH_Operate_Pattern_UserDefined	SKETCH_Operate_Intersect SKETCH_Operate_Divide SKETCH_Operate_Merge SKETCH_Delete SKETCH_Copy SKETCH_Reference_Plane_ReferencePlane SKETCH_Reference_Axis SKETCH_Reference_CoordSys SKETCH_Reference_Curve SKETCH_Reference_Line SKETCH_Reference_Point SKETCH_Reference_Section_Attach SKETCH_Reference_Section_CrossSection
Surface	SURFACE_Operate_Transform_Move SURFACE_Operate_Transform_Rotate SURFACE_Operate_Transform_Mirror	SURFACE_Operate_Transform_Scale SURFACE_Delete SURFACE_Copy
Solid	SOLID_Create_Protrusion_Helical SOLID_Create_Cut_Helical SOLID_Create_Feature_Pipe SOLID_Create_Feature_Thread SOLID_Create_Feature_UserDefined SOLID_Operate_Boolean_Union SOLID_Operate_Boolean_Intersect SOLID_Operate_Boolean_Difference SOLID_Operate_Pattern_Rectangular	SOLID_Operate_Pattern_Circular SOLID_Operate_Transform_Move SOLID_Operate_Transform_Rotate SOLID_Operate_Transform_Mirror SOLID_Operate_Transform_Scale SOLID_Delete_Object SOLID_Delete_Pattern SOLID_Delete_Parameter SOLID_Copy
Constraint	CONSTRAINTS_Create_Equation CONSTRAINTS_Delete	CONSTRAINTS_Copy