2.1 PRODUCT CYCLE

To appreciate the scope of Product Data Exchange in the operations of a manufacturing firm, it is appropriate to examine the various activities and functions that must be accomplished in the design and manufacture of a product. We will refer to these activities and functions as the product cycle.

A diagram showing the various steps in the product cycle is presented in Figure 1.The cycle is driven by customers and markets, which demand the product. It is realistic to think of these as a large collection of diverse industrial and consumer markets rather than one monolithic market. Depending on the particular customer group, there will be differences in the way the product cycle is activated.

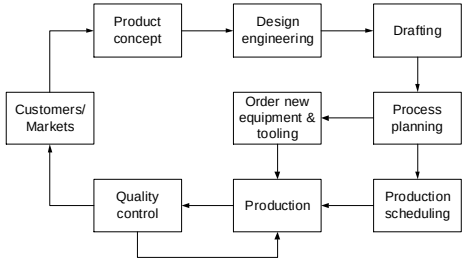

Figure 1 Product cycle (design & manufacturing)

In some cases, the customer performs the design functions and a different firm manufactures the product. In other cases, the same firm accomplishes design and manufacturing. Whatever the case, the product cycle begins with a concept, an idea for a product. This concept is cultivated, refined, analyzed, improved, and translated into a plan for the product through the design engineering process. Drafting a set of engineering drawings showing how the product is made and providing a set of specifications indicating how the product should perform document the plan.

Except for engineering changes typically follow the product throughout its life cycle; this completes the design activities in Figure 1. The next activities involve the manufacture of the product. A process plan is formulated which specifies the sequence of production operations required to make the product. New equipment and tools must sometimes be acquired to produce the new product. Scheduling provides a plan that commits the company to the manufacture of certain quantities of the product by certain dates. Once all of these plans are formulated, the product goes into production, followed by quality testing, and delivery to the customer.

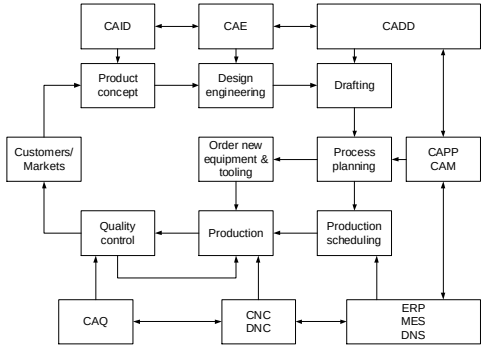

Figure 2 Product cycle revised with CAX overlaid

CAX technologies enable concurrent engineering and enhanced the level of integration.

The impact of CAX is manifest in all of the different activities in the product cycle, as indicated in Figure 2. Computer-aided design and automated drafting are utilized in the conceptualization, design, and documentation of the product. Computers are used in process planning and scheduling to perform these functions more efficiently. Computers are used in production to monitor and control the manufacturing operations. In quality control, computers are used to perform inspections and performance tests on the product and its components.

As illustrated in Figure 2, CAX is overlaid on virtually all of the activities and functions of the product cycle. In the design and production operations of a modern manufacturing firm, the computer has become a pervasive, useful, and indispensable tool.

It is strategically important and competitively imperative that manufacturing firms and the people who are employed by them understand Product Data Exchange.

2.2 PRODUCT DATA PROBLEM

Increased competitiveness in the market place means that engineering companies have to be flexible and responsive to rapidly changing market needs. Maintaining a competitive edge requires the adoption of state-of-the-art techniques and methodologies, sometimes before they have stood the test of time.
Such a requirement has given rise to the concept of the Extended Enterprise, where companies have to work closely with their suppliers, customers, and partners in order to shorten the product development lifecycle and to highlight potential problems much as possible. They were traditionally carried out under the same roof,

may now be carried out by different project teams belonging to different organizations that are geographically distributed throughout the world.

To make this possible, and to enable team members in an Extended Enterprise to work effectively, a new generation of information systems is necessary. Such systems must be capable of enabling simultaneous and controlled access to the same data pool by different team members to carry out their different tasks.

These systems must provide:

Portability: Not only is it able to run a given application on different hardware and operating system environments, but also to move data among applications. That is, the output of one application, say a design tool, may be the input to an analysis tool. Early application software, which was standalone and hardware and operating system bound, is no longer acceptable in today's fast paced engineering computing environments.

Interoperability: For team members to work in parallel requires their applications to be capable of sharing data. Data sharing ensures that every team member is working on the latest set of data.

Longevity: Data should outlive the software and hardware on which it was created. At first glance, this may seem necessary only for large, long-term projects, e.g. process plants, aircraft, buildings, and weapons systems. However, with the fast rate of evolution of technology, system obsolescence has become a significant issue for practically every project. To speed new product development, reusing preexisting designs has become a tactic. When designs can't be accessed, previous work isn't leveraged.

Extensibility: Design and data modeling techniques are in continuous evolution. It is important that both data and application tools can continue to take advantage of new and innovative techniques.

The evolution of CAD/CAM/CAE systems over the past two decades was driven by industry's need for more sophisticated and productive design and analysis tools capable of designing more complex products more quickly.

2.3 DATA EXCHANGE

The current generation of CAD/CAM/CAE systems such as CADDS 5, CATIA Version 4, Euclid 3, and I-DEAS Master Series, each provide very strong solutions for engineering mechanical products. But, each does so with proprietary technical capabilities, and worse, with proprietary data formats. While users can work concurrently with the same set of product data within any single vendor's application sets, they can't work easily with a mixed set of applications selected from more than one vendor.

Thus whilst these systems can provide a practical solution within an enterprise, by "standardizing" on one vendor's system, this is not generally the case in an Extended Enterprise in which it is typical for overlapping systems from several vendors to be in use. Where suppliers, customers and partners are changing all the time, it is impossible to effect such "standardization" throughout the Extended Enterprise.

Therefore, the seamless integration between applications supplied within one vendor's product line solved yesterday's problem where everything was done under the same roof, but has yet to provide a solution for today's problem. In other words, the islands of automation have grown bigger, but they remain islands in a bigger sea.

Over the past decade many industry, national, and international standards for exchanging data among different systems have emerged. Some, like IGES are mature and are widely used through out industry. Whilst these standards provide a mechanism for moving data among different proprietary systems, it is difficult to use them as the basis for an interactive, concurrent engineering environment that utilizes these systems as illustrated in Figure 3.

Data exchange standards serve to provide a mechanism to ferry data between islands of automation.

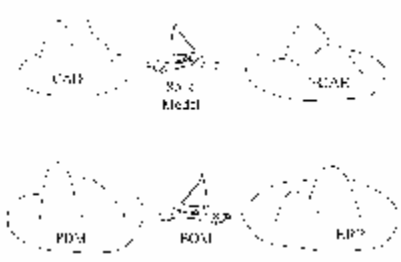

Figure 3 Product Data Exchange as linking tool between islands of automation

Data exchange standards address only the problem of portability, and to a certain extent, longevity of data. They do not address application interoperability or data sharing. There is a problem in working with the same data simultaneously, using two different applications that require different data formats. When one application changes the data, the other can't access the changes until the data is translated. How often the working set of data is translated determines the level of simultaneity that the users perceive.

In addition, three other factors adversely impact users of current CAD data exchange formats:

- Incomplete or non-existent coverage of current data forms such as parametric models, form features, and product structures

- Loss of data integrity, for instance the degradation of surfaces as they are translated from one mathematical format (say NURBS) to another (say Bezier).

- Loss of numerical precision because most applications assume a particular level of numerical accuracy for their internal data, which may not match that of either the data transfer format or the receiving application.

These all lead to data translation results that are, in the worst case unusable, but often are unacceptable even with significant levels of additional manual labor to put things right. The attendant loss of productivity is unacceptable to many users.

Data exchange standards do not support extensibility either. Existing data exchange standards rely on the definition of a finite number of primitive geometric entities, e.g., line, arc, ellipse. Real life engineering problems require data models that contain complex entities, and these change with evolving needs, e.g. valve, pump, BOM, ECR, etc. The range of such entities is not finite. Furthermore, to ensure the usefulness of data over the lifecycle of a product (longevity of data), not only is it necessary to find a standard with a rich unambiguous data modeling capability, but also one that is capable of capturing the semantics, or meaning of the data.

Clearly, a new type of standard is required. A standard that is possible to build bridges between islands of automation. It needs to cater for the uniformity that is the essence of every standard, as well as the extensibility that leaves the door open for innovation and evolution. This standard was actually conceived in the mid 1980s. It is **ISO 10303**, **ST**andard for **E**xchange of **P**roduct model data, commonly known as STEP.

2.4 STEP

STEP, officially known as ISO 10303 is defined in ISO 10303 Part 1 (the first of the STEP standard documents) as:

"ISO 10303 is an international standard for the computer sensible representation and exchange of product data. The objective is to provide a mechanism capable of describing product data throughout the lifecycle of a product, independent of any particular system. The nature of this description makes it suitable not only for file exchange, but also as a basis for implementing and sharing product databases and archiving."

If we examine this definition clause by clause, we see that it addresses all the requirements outlined in The Problem in previous section.

Extensibility

> ***"ISO 10303 is an international standard for the computer sensible representation and exchange of product data."***

Since computers cannot think, for the representation to be computer sensible, it must be unambiguous; and since it needs to cover product data, the representation must be extensible and capable of representing a variety of data types not just geometry as is the case with current data exchange standards such as IGES, VDA-FS, DXF, etc. While an IGES file representing a CAD drawing is computer readable, it is not computer sensible, as it does not contain semantics. The meaning of the lines, arc, text and other entities that may be contained in the CAD drawing can only be understood by a human who has had engineering drawing training. The evolution of data exchange standard is given in Figure 4.

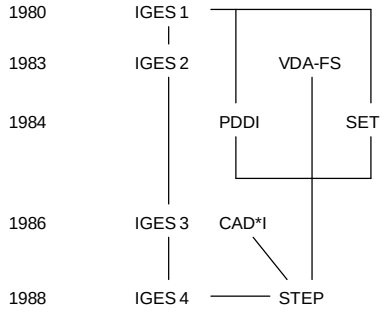

Figure 4 Evolution of CAD Standard

The term product data is relative. It is a function of the processes related to the product throughout its lifecycle, e.g. design, analysis, support, etc. As the product moves through different processes, the data that defines the product (for each process) can be quite different.

Longevity

> ***"The objective is to provide a mechanism capable of describing product data throughout the lifecycle of a product, independent of any particular system."***

Application systems that create, change, and otherwise use product data tend to be short-lived. They may undergo major changes to improve their capabilities and productivity. This evolution of applications often requires that the underlying data formats be updated or completely changed. Product data must transcend changes in applications in order for it to have a long life and not lose its value. Stifling changes to applications is counter-productive in that the result is reduced productivity and failure to take advantage of advances in technology. Computer systems must be allowed to evolve in much the same way that production facilities are re-tooled to maintain a competitive edge through the use of state-of-technology.

Data independence is particularly important for products that are complex and have a long lifecycle, e.g. process plants, aircraft, buildings, and ships. Throughout their

life cycles, which are measured in decades rather than years, such products often undergo design changes in response to safety, legal, and operational factors. Easy access to product data can therefore be both critical and a legal requirement. As application vendors provide new versions at least once a year, tying data to a particular version of a particular system becomes a serious liability.

Portability

> ***"The nature of this description makes it suitable not only for file exchange…"***

The exchange file concept provides portability of product data, complete with its description or semantics, not just a potentially ambiguous subset of product data, e.g. as geometry represented by a 2D drawing. STEP provides a neutral way for describing the product data dictionary. Instead of requiring special translation programs, under STEP, the recipient of the data needs only the same data dictionary to interpret the data. The STEP data definition can be expanded and improved to cover new data types without re-programming translation programs. In contrast, existing data exchange standards only allow pre-defined data definitions without a neutral data definition format. Any product data that cannot be matched with the pre-defined definitions cannot be exchanged.

Interoperability

> ***"…but also as a basis for implementing and sharing product data-bases and archiving."***

Finally, the standard should cater for data sharing and databases to enable concurrent access by different users through different applications. The existence of one logical database is the only way to ensure that all users have access to the most up-to-date version of the data that they need too carry out their intended tasks. The existence of a database also makes it possible to put in place data management and control procedures to ensure that product data in all its versions is properly managed and accessible only by those with proper authority. Such controls are not practicable for exchange files that are transitory formats on different systems.

In short, STEP was conceived to address the key product data problems of Portability, Interoperability, Longevity, and Extensibility.

STEP – A Standard and Architecture

How can STEP be a standard, which by definition needs to be fixed, and at the same time be extensible which implies variability? The answer is that STEP defines as a standard as well as architecture. Though STEP is officially one standard, it is in fact

a family of standards called Parts. Each STEP part is self-contained and each goes through the ISO standardization process as a standard in its own right'. STEP parts can be broadly grouped into two main categories, namely:

- STEP data models
- Tools to create STEP data models, and to enable the communication of STEP data

The data models are the descriptions of product data, or the data dictionary in relational database terminology. The tools include a language called EXPRESS which is used to define the data model, SDAI an Application Programming Interface (API) to enable an application to communicate with a virtual STEP database, and the Exchange File format which enables STEP data to be exported in ASCII text format from one system to another. The tools are fixed and can be used at any time to create STEP data models in response to industry needs.

A STEP data model may range from the description of 2D geometry, i.e. similar to other data exchange standards, to a model capable of describing the whole lifecycle a of product (or products) for a given industry sector and including a wide range of types of data other than geometry. ISO have recently released twelve STEP Parts, collectively called the Initial Release of STEP. Eight of these define STEP data models and the remaining four relate to description and implementation methods.

In other words, STEP not only enables the creation of data models that extend beyond the description of product geometry', but also provides tools and a methodology to describe data for potentially any conceivable product or service in any conceivable industry. These same STEP tools can be used by ISO to create new ISO standard STEP data models; or by national standards bodies, consortia or even individual companies, to create national, industry, and company standards. In other words, unlike previous data exchange standards, the scope of STEP is *not fixed*. The open approach is given in Figure 5.

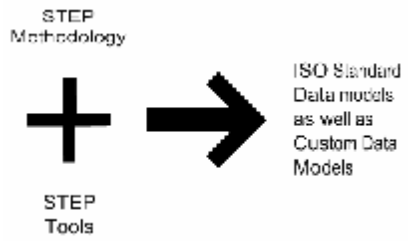

Figure 5 STEP Methodology and implementation Methods can be used to create any Product Data Model

The flexibility of STEP overcomes two of the major problems normally associated with standards, namely:

- Standards take too long to develop and often become out of date by the time they are released. For instance, IGES has never provided full support for solid models and constraint modeling.

- Standards tend to cater for the lowest common denominator, which means that they don't provide the most productive solution.

STEP - More than a Data Exchange Standard

In addition to providing the means to define complex and extensible data models, STEP also defines a mechanism for data sharing and databases. In fact STEP was originally defined to have four conformance levels.

The reference to four levels of STEP implementations was widely used in the early stages of the development of STEP. Then reference to levels of STEP has all but disappeared. Instead, the common practice today is to refer to data exchange (Level 1) and data sharing/databases (Level 3). Definitions of all four levels are included here for completeness.

Conformance levels are important for users. The Level associated with a particular implementation of STEP indicates how rigorous that particular implementation is - the higher the Level number, the closer the implementation is to fulfilling all of STEP ideals.

Level 1-Exchange File
STEP provides a standard format for data exchange files is based on EXPRESS, the STEP Data Definition Language. This is similar to the capability provided by previous data exchange standard, except that STEP offers extensible and much richer data models. With Level 1 conformance however, there is no standard query language, no standard navigational capabilities, no standard mechanisms for accessing the data and no standard methods for tracking changes made from one read-write operation to the next. Figure 6 depicts two independent applications that can exchange data via a STEP Exchange File, providing data portability.

Figure 6 Data Exchange Via a STEP Exchange File Provides Portability of Data

Level 2-Working Form
Level 2 enables more than one application to share non-persistent data (i.e., data held in computer memory rather than in permanent storage such as a database on disk). Level 2 provides standard mechanisms for accessing and navigating through data; these are done through SDAI (***Standard Data Access Interface***) and the STEP API (Application Programming Interface), Figure 7 depicts two applications sharing the same data in shared computer memory thus providing application interoperability.

Figure 7 Working form Provides Sharing of Non-Persistent Data Between Applications, Supporting *Interoperability*

Level 3-Shared Database

In Level 3, product data is stored in a database management system and is therefore persistent data. It also provides the potential for concurrent access by multiple applications without the need for an intermediate exchange file. STEP does not specify a standard way for the physical implementation of the data in the STEP database, Figure 8. Therefore, any database paradigm, e.g. hierarchical, network, relational, object-oriented, as well as flat file could be used to store the data. Access to the data however, is standardized and is affected through SDAI. Level 3 therefore provides a mechanism for sharing data (interoperability) that is independent of any application (longevity).

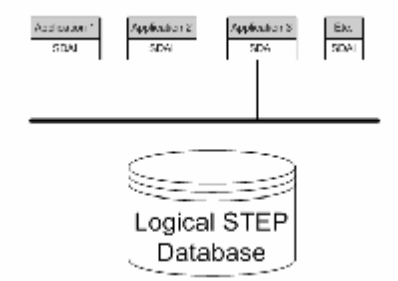

Figure 8 STEP Database (Data are independent of all applications, providing Data Longevity

Level 4-Knowledge Base

Level 4 is currently more a concept than a reality. It is intended to provide data sharing within a knowledge-based system where the data may include constraint specifications found in complex data models.

In contrast to Level 3, where data constraints are enforced by the data typing implemented in the database and user applications, Level 4 provides explicitly defined data constraints and a solver. In this implementation, the system can effectively satisfy or relax the constraints to produce a completely or partially consistent data model, Figure 9.

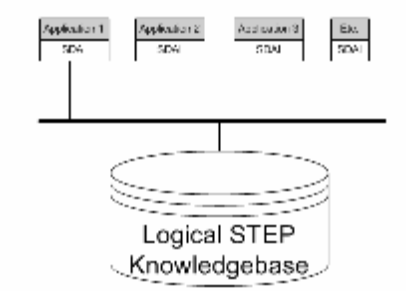

Figure 9 STEP Knowledgebase. Any number of applications can share the same persistent knowledge base. The data model can extend to include rules and constraints

Data Modeling Issues

The official definition of STEP, see page 8, states:

> ***"ISO 10303 is an international standard for the computer sensible representation and exchange of product data. The objective is to provide a mechanism capable of describing product data through the life cycle of a product... "***

- What is product data?
- What constitutes a description of it?
- Is there a unique, unambiguous product description?
- If not, can all product descriptions conform to the STEP standard?

This section answers these and other related questions

Product Data and Product Data Models

Product data is a function of the applications in which the data is to be used, while the description of the data, or the product data model, is the semantics that go with the data to facilitate its correct interpretation by its recipients. In traditional database modeling terminology, the product data model defines the schema (or tables in relational database technology), and the product data is the actual information that populates the table.

For a product's description to be computer-sensible, it should be unambiguous; and to describe product data throughout the lifecycle of a product requires the ability to handle not only geometry, but potentially any type of data that may be needed now and in the future, such as cost, material, color, supplier, or test results. The requirements that product description be unambiguous and not predefine, are two of the major differentiators of STEP from previous standards.

In the early days of computer aided engineering techniques the focus was on automating the production and editing of engineering drawings. In those days,

product data meant the lines, arcs, splines, etc., that were used to generate a 2D drawing. 3D wireframe, surface, and solid models evolved from this, each requiring new types of entities in their data models. There was a parallel evolution of finite element, boundary element, Computational Fluid Dynamics, and other analysis models. More recently, there has been the need to model not only product attributes, but also product related configurations and processes. In short product data and product descriptions are live entities. What constitutes product description today is different from what it was a few years ago and will be different still in a few years time. STEP was designed to cope with this evolutionary process. The following list illustrates the wide range of types of product related information covered by STEP; these STEP Parts have already been developed or are under development (on the left are the official ISO STEP Part numbers).

10303-41	Fundamentals of Product Description and Support
10303-42	Geometric and Topological Representation
10303-43	Representation Structures
10303-44	Product Structure Configuration
10303-45	Materials
10303-46	Visual Presentation
10303-47	Shape Tolerances
10303-48	Form Features
10303-49	Process Structure and Properties
10303-101	Drafting Resources
10303-103	Electrical/Electronics Connectivity
10303-104	Finite Element Analysis
10303-105	Kinematics

It can be seen that STEP has broad application to many types of products and supports much more than geometric model definition and exchange.

Level of Richness of a Data Model

As mentioned earlier, CAD/CAM/CAE systems have evolved over the past two decades to provide high levels of functional sophistication in response to industry needs. For as long as there is innovation, this trend will continue. To deliver increasing functionality, system vendors have had to improve and refine the data models used in their systems to increasingly higher levels of sophistication (or data richness), always attempting to approach some ultimate product definition, Figure 10.

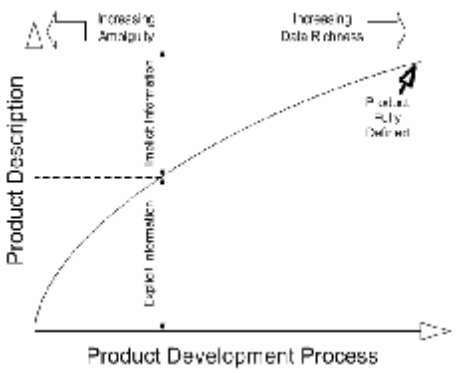

Figure 10 Relationship between product description, product definition, level of richness, and ambiguity

The term product definition is used here to include everything about a product that can be described in a computer sensible form. The gap between product definition and product description (the explicitly defined information about the product) is the implicit information that is not described and remains for the recipient of the product data to interpret correctly. The amount of implicit information is a measure of the ambiguity of the product's description at any point in time.

Level of Richness of a Product Data Model

The level of richness of a data model depends on the level of detail required to define a particular product. If the objective is to automate the process of producing engineering drawings, it is only necessary to define a data model that includes entities like line, arc, dimension, etc. Here the drawing has no built in "intelligence," e.g., a circle may define a hole or a boss, and a line may represent the intersection of two planes or the intersection of a plane and a fillet. This information is implicit, and its correct interpretation depends on the recipient's knowledge of engineering drawings. On the other hand, it is more appropriate to use a surface model for the design of a car body or an aircraft wing, and a solid model to calculate mass properties and motion dynamics. Surface and solid models differ from drawings, and from each other in their level of richness.

Moving from 2D drafting to solid models, through wireframe and surface models increases the level of richness and decreases the ambiguity of the product's definition.

As the level of richness is increased, the amount of implicit information and ambiguity are reduced. Conversely, as illustrated in Figure 11, moving from a high level of richness to a lower level increases the quantity of implicit information as well as the level of ambiguity since some of the product's descriptive information must be lost. For instance for a line in 3D space, the location of each end point maybe defined by X-Y-Z coordinates. For the data to be translated into a 2D image of the

line requires only X-Y coordinates of the end points. Therefore, unless the Z coordinates of the end points are retained in some way, it is not possible to regenerate the original 3D line. While it is possible to pass engineering drawing information from a CAD system that supports 3D models to a 2D drafting system, it is not always possible to automatically generate a 3D model from data transferred from a 2D drawing. This usually requires additional information that may have to be entered manually.

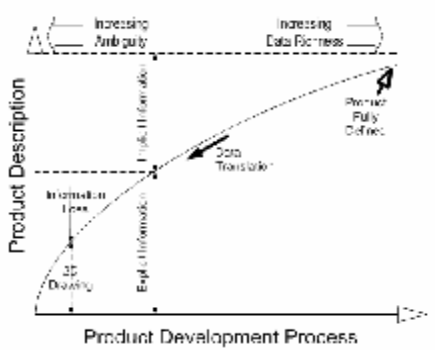

Figure 11 Data loss occurs when moving to a lower level of data richness

The concept of level of richness is not limited to geometry. A schematic representing a hydraulic or electronic circuit may be derived from a data model where circuit components have operational attributes as well as intelligent, associative relationships with each other. While the same schematic layout can be produced in a 2D drafting package, in this case it is simply a drawing or collection of symbols that does not contain knowledge about the functioning of the system. The user, rather than the system, has to ensure that the drawing does not represent an invalid schematic.

As a general rule, therefore, it is easier to move from a higher to a lower level of data richness. Indeed this change can usually be effected automatically, e.g., it richness is one where a 2D circle is defined by three points in one model, by a point and a radius in another, and by two end points of a diameter in a third model.

STEP not only supports models with higher levels of richness than past data exchange standards, see Figure 12, but can also support multiple data models or representations of the same product referred to as Conformance Classes, see page 8.

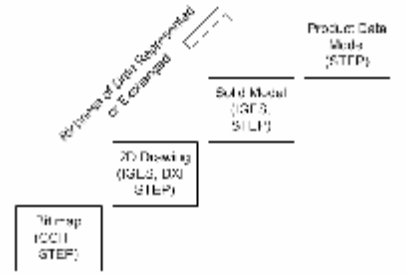

Figure 12 Increased ability to represent and exchange product data with STEP

This feature is an important differentiator of STEP It means that different data models for a product can be defined for use in different applications, e.g., CAD, CAM, CAE, PIM, MRP, etc.; or one product data model can be defined, from which different representations or views can be derived, e.g., to support the needs of different users within an extended enterprise. Therefore, because of the richness of data it can represent and the unlimited types of data it can define, STEP can support application areas beyond those covered by past standards. In summary, data exchange and data sharing between different systems is only possible if the exchange and communication of data takes place between systems supporting:

- The same data model.
- Data models with the same levels of richness, e.g., one solid model to or from another solid model.
- From one with high level of richness to one with lower level of richness, e.g., from a solid model to a surface or wireframe model.

Exchanging or sharing incompatible data models requires special process that may include manual input.

STEP and Product Data Management

While data formatting and exchange standards for geometric data have long been available, no standard format exists for managing all product information. A goal of STEP is to provide methods for storing and exchanging data management entities such as configurations, BOMB, engineering changes, organizational and personnel information, and additional attributes that pertain to Product Data Management (PIM).

An important and difficult area in PIM is the interaction between the PIM system and the applications that create and change product data.

STEP Conformance Levels 3 and 4 offered a data management layer between user applications and the STEP repository, simplifying their integration with Product Data Management systems that provide data security and configuration management.

The architecture of the current generation of PIM systems, Figure 13, has the following drawbacks:

- Data generated by one application has to be translated and accessed by other applications.
- No interoperability between applications, though an application that is integrated with the PIM system may access the PIM system's metadata through the PIM API.
- Coarse granularity of data with a few exceptions, PIM systems generally manage file-based data.
- Customized data integration is required between the PIM system and every application it supports.
- Depending on the specific external application and the PIM system, the interface link may not be robust enough to ensure data integrity at all times.

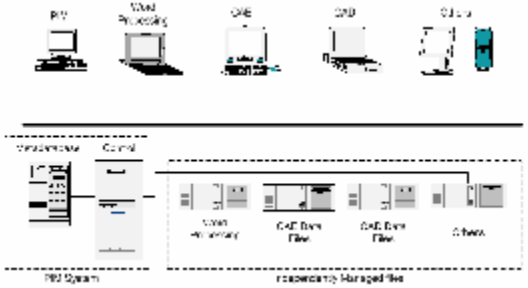

Figure 13 Architecture of current generation of PIM systems

In contrast, the PIM architecture, Figure 14, based on STEP offers:

- Data that is independent of the application that generates it.
- Data that can be shared among applications.
- Fine granularity of data - down to the entity level if needed. This also means that data storage can be reduced, since only new versions of entities need to be stored - not new versions of whole files.
- Any application that uses the SDAI can communicate with any other that uses SDAI, including a PIM system.

- Applications can only access data through the PIM system, thus it is easier to ensure data model integrity.

Though STEP is officially one standard, it is in effect a family of standards, that have been conceived to support data for a number of practical engineering problems. STEP allows not only the exchange of geometry data, but also the exchange and sharing of all types of product data throughout a product's lifecycle. STEP extensibility offers a unique opportunity for industry to benefit from standardization without foregoing the continuing innovation that is essential for industrial competitiveness.

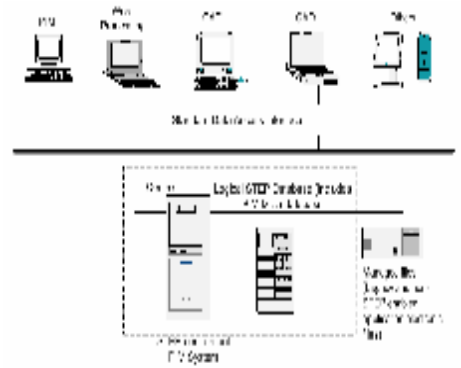

Figure 14 PIM application integration with STEP

2.5 STEP REPRESENTATION

The STEP representation of a cube with length 50 units and centered at x=0, y=0 and z=0 is given in the appendix and the geometry is shown in figure 15.

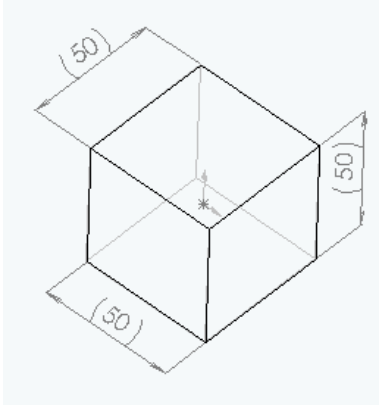

Figure 15 A solid cube

If we take a quick look on the STEP file, out of around 300 lines half of the content are dedicated for geometry description.

A simplified tree for describing the “part” is illustrated below, Figure 16.

Figure 16 STEP Geometry

This chapter provides the basic idea of Product Data Exchange. For more detailed information, you can visit the following URL:

http://www.iso.org/iso/en/ISOOnline.openerpage

http://www.cimdata.com

http://www.steptools.com/

Appendix

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION (( 'STEP AP203' ), '1' );
FILE_NAME ('cube.STEP', '2001-09-22T07:00:34',( 'tyiem26' ),( 'iem/ive(ty)' ), 'SwSTEP 2.0','SolidWorks 2001039','' );
FILE_SCHEMA (( 'CONFIG_CONTROL_DESIGN' ));
ENDSEC;
DATA;
#1 = CARTESIAN_POINT ( 'NONE',  ( 0.0000000000000000000, 0.0000000000000000000, 0.0000000000000000000 ) ) ;
#2 = DIRECTION ( 'NONE',  ( 0.0000000000000000000, 0.0000000000000000000, 1.0000000000000000000 ) ) ;
#3 = DIRECTION ( 'NONE',  ( 1.0000000000000000000, 0.0000000000000000000, 0.0000000000000000000 ) ) ;
#4 = APPLICATION_CONTEXT ( 'configuration controlled 3d designs of mechanical parts and assemblies' ) ;
#5 = APPLICATION_PROTOCOL_DEFINITION ( 'international standard', 'config_control_design', 1994, #4 ) ;
#6 = MECHANICAL_CONTEXT ( 'NONE', #4, 'mechanical' ) ;
#7 = PRODUCT ( '2', 'cube', '', ( #6 ) ) ;
#8 = PERSON ( 'UNSPECIFIED', 'UNSPECIFIED', 'UNSPECIFIED', ('UNSPECIFIED'), ('UNSPECIFIED'), ('UNSPECIFIED') ) ;
#9 = ORGANIZATION ( 'UNSPECIFIED', 'UNSPECIFIED', '' ) ;
#10 = PERSON_AND_ORGANIZATION ( #8, #9 ) ;
#11 = PERSON_AND_ORGANIZATION_ROLE ( 'design_owner' ) ;
#12 = CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT ( #10, #11, ( #7 ) ) ;
#13 = APPLICATION_CONTEXT ( 'configuration controlled 3d designs of mechanical parts and assemblies' ) ;
#14 = APPLICATION_PROTOCOL_DEFINITION ( 'international standard', 'config_control_design', 1994, #13 ) ;
#15 = DESIGN_CONTEXT ( 'detailed design', #13, 'design' ) ;
#16 = PRODUCT_DEFINITION ( 'UNKNOWN', '', #48, #15 ) ;
#17 = PERSON_AND_ORGANIZATION ( #8, #9 ) ;
#18 = APPROVAL_STATUS ( 'not_yet_approved' ) ;
#19 = APPROVAL ( #18, 'UNSPECIFIED' ) ;
#20 = APPROVAL_ROLE ( '' ) ;
#21 = APPROVAL_PERSON_ORGANIZATION ( #17, #19, #20 ) ;
#22 = COORDINATED_UNIVERSAL_TIME_OFFSET ( 8, 0, .AHEAD. ) ;
#23 = LOCAL_TIME ( 15, 0, 34.00000000000000000, #22 ) ;
#24 = CALENDAR_DATE ( 2001, 22, 9 ) ;
#25 = DATE_AND_TIME ( #24, #23 ) ;
#26 = APPROVAL_DATE_TIME ( #25, #19 ) ;
#27 = CC_DESIGN_APPROVAL ( #19, ( #16 ) ) ;
#28 = PERSON_AND_ORGANIZATION ( #8, #9 ) ;
#29 = PERSON_AND_ORGANIZATION_ROLE ( 'creator' ) ;
#30 = CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT ( #28, #29, ( #16 ) ) ;
#31 = COORDINATED_UNIVERSAL_TIME_OFFSET ( 8, 0, .AHEAD. ) ;
#32 = LOCAL_TIME ( 15, 0, 34.00000000000000000, #31 ) ;
#33 = EDGE_LOOP ( 'NONE', ( #236, #89, #220, #229 ) ) ;
#34 = CALENDAR_DATE ( 2001, 22, 9 ) ;
#35 = DATE_AND_TIME ( #34, #32 ) ;
#36 = DATE_TIME_ROLE ( 'creation_date' ) ;
#37 = CC_DESIGN_DATE_AND_TIME_ASSIGNMENT ( #35, #36, ( #16 ) ) ;
```



```
#38 = PERSON_AND_ORGANIZATION ( #8, #9 ) ;
#39 = APPROVAL_STATUS ( 'not_yet_approved' ) ;
#40 = APPROVAL ( #39, 'UNSPECIFIED' ) ;
#41 = APPROVAL_ROLE ( '' ) ;
#42 = APPROVAL_PERSON_ORGANIZATION ( #38, #40, #41 ) ;
#43 = COORDINATED_UNIVERSAL_TIME_OFFSET ( 8, 0, .AHEAD. ) ;
#44 = LOCAL_TIME ( 15, 0, 34.00000000000000000, #43 ) ;
#45 = CALENDAR_DATE ( 2001, 22, 9 ) ;
#46 = DATE_AND_TIME ( #45, #44 ) ;
#47 = APPROVAL_DATE_TIME ( #46, #40 ) ;
#48 = PRODUCT_DEFINITION_FORMATION_WITH_SPECIFIED_SOURCE ( 'ANY', '', #7, .NOT_KNOWN. ) ;
#49 = CC_DESIGN_APPROVAL ( #40, ( #48 ) ) ;
#50 = VERTEX_POINT ( 'NONE', #112 ) ;
#51 = PERSON_AND_ORGANIZATION ( #8, #9 ) ;
#52 = PERSON_AND_ORGANIZATION_ROLE ( 'creator' ) ;
#53 = CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT ( #51, #52, ( #48 ) ) ;
#54 = PERSON_AND_ORGANIZATION ( #8, #9 ) ;
#55 = PERSON_AND_ORGANIZATION_ROLE ( 'design_supplier' ) ;
#56 = CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT ( #54, #55, ( #48 ) ) ;
#57 = SECURITY_CLASSIFICATION_LEVEL ( 'unclassified' ) ;
#58 = SECURITY_CLASSIFICATION ( '', '', #57 ) ;
#59 = PERSON_AND_ORGANIZATION ( #8, #9 ) ;
#60 = PERSON_AND_ORGANIZATION_ROLE ( 'classification_officer' ) ;
#61 = CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT ( #59, #60, ( #58 ) ) ;
#62 = COORDINATED_UNIVERSAL_TIME_OFFSET ( 8, 0, .AHEAD. ) ;
#63 = LOCAL_TIME ( 15, 0, 34.00000000000000000, #62 ) ;
#64 = CALENDAR_DATE ( 2001, 22, 9 ) ;
#65 = DATE_AND_TIME ( #64, #63 ) ;
#66 = DATE_TIME_ROLE ( 'classification_date' ) ;
#67 = ADVANCED_FACE ( 'NONE', ( #113 ), #114, .F. ) ;
#68 = VERTEX_POINT ( 'NONE', #121 ) ;
#69 = CC_DESIGN_DATE_AND_TIME_ASSIGNMENT ( #65, #66, ( #58 ) ) ;
#70 = PERSON_AND_ORGANIZATION ( #8, #9 ) ;
#71 = APPROVAL_STATUS ( 'not_yet_approved' ) ;
#72 = APPROVAL ( #71, 'UNSPECIFIED' ) ;
#73 = APPROVAL_ROLE ( '' ) ;
#74 = APPROVAL_PERSON_ORGANIZATION ( #70, #72, #73 ) ;
#75 = COORDINATED_UNIVERSAL_TIME_OFFSET ( 8, 0, .AHEAD. ) ;
#76 = LOCAL_TIME ( 15, 0, 34.00000000000000000, #75 ) ;
#77 = CALENDAR_DATE ( 2001, 22, 9 ) ;
#78 = DATE_AND_TIME ( #77, #76 ) ;
#79 = APPROVAL_DATE_TIME ( #78, #72 ) ;
#80 = CC_DESIGN_APPROVAL ( #72, ( #58 ) ) ;
#81 = CC_DESIGN_SECURITY_CLASSIFICATION ( #58, ( #48 ) ) ;
#82 = PRODUCT_RELATED_PRODUCT_CATEGORY ( 'detail', '', ( #7 ) ) ;
#83 = PRODUCT_DEFINITION_SHAPE ( 'NONE', 'NONE',  #16 ) ;
#84 = SHAPE_DEFINITION_REPRESENTATION ( #83, #218 ) ;
```

```
#85 = MANIFOLD_SOLID_BREP ( 'NONE', #90 ) ;
#86 = ORIENTED_EDGE ( 'NONE', *, *, #224, .T. ) ;
#87 = VERTEX_POINT ( 'NONE', #131 ) ;
#88 = ORIENTED_EDGE ( 'NONE', *, *, #226, .F. ) ;
#89 = ORIENTED_EDGE ( 'NONE', *, *, #156, .F. ) ;
#90 = CLOSED_SHELL ( 'NONE', ( #111, #102, #94, #67, #110, #119 ) ) ;
#91 = EDGE_CURVE ( 'NONE', #223, #87, #132, .T. ) ;
#92 = ORIENTED_EDGE ( 'NONE', *, *, #120, .F. ) ;
#93 = ORIENTED_EDGE ( 'NONE', *, *, #226, .T. ) ;
#94 = ADVANCED_FACE ( 'NONE', ( #143 ), #144, .T. ) ;
#95 = EDGE_CURVE ( 'NONE', #137, #139, #150, .T. ) ;
#96 = ORIENTED_EDGE ( 'NONE', *, *, #100, .F. ) ;
#97 = EDGE_LOOP ( 'NONE', ( #227, #88, #92, #228 ) ) ;
#98 = ORIENTED_EDGE ( 'NONE', *, *, #95, .F. ) ;
#99 = EDGE_CURVE ( 'NONE', #237, #87, #161, .T. ) ;
#100 = EDGE_CURVE ( 'NONE', #139, #237, #165, .T. ) ;
#101 = ORIENTED_EDGE ( 'NONE', *, *, #109, .F. ) ;
#102 = ADVANCED_FACE ( 'NONE', ( #169 ), #170, .F. ) ;
#103 = ORIENTED_EDGE ( 'NONE', *, *, #224, .F. ) ;
#104 = EDGE_LOOP ( 'NONE', ( #222, #108, #103, #232 ) ) ;
#105 = ORIENTED_EDGE ( 'NONE', *, *, #157, .T. ) ;
#106 = ORIENTED_EDGE ( 'NONE', *, *, #109, .T. ) ;
#107 = EDGE_LOOP ( 'NONE', ( #235, #101, #230, #93 ) ) ;
#108 = ORIENTED_EDGE ( 'NONE', *, *, #91, .F. ) ;
#109 = EDGE_CURVE ( 'NONE', #137, #231, #176, .T. ) ;
#110 = ADVANCED_FACE ( 'NONE', ( #180 ), #181, .F. ) ;
#111 = ADVANCED_FACE ( 'NONE', ( #186 ), #187, .F. ) ;
#112 = CARTESIAN_POINT ( 'NONE', ( 24.99999999999999600, 24.99999999999999600, 25.00000000000000000 ) ) ;
#113 = FACE_OUTER_BOUND ( 'NONE', #33, .T. ) ;
#114 = PLANE ( 'NONE', #115 ) ;
#115 = AXIS2_PLACEMENT_3D ( 'NONE', #116, #117, #118 ) ;
#116 = CARTESIAN_POINT ( 'NONE', ( -24.99999999999999600, -25.00000000000000400, -25.00000000000000000 ) ) ;
#117 = DIRECTION ( 'NONE', ( 0.000000000000000000, 0.000000000000000000, 1.000000000000000000 ) ) ;
#118 = DIRECTION ( 'NONE', ( 1.000000000000000000, 0.000000000000000000, 0.000000000000000000 ) ) ;
#119 = ADVANCED_FACE ( 'NONE', ( #192 ), #193, .F. ) ;
#120 = EDGE_CURVE ( 'NONE', #223, #50, #198, .T. ) ;
#121 = CARTESIAN_POINT ( 'NONE', ( 24.99999999999999600, 24.99999999999999600, -25.00000000000000000 ) ) ;
#122 = LINE ( 'NONE', #123, #124 ) ;
#123 = CARTESIAN_POINT ( 'NONE', ( -25.00000000000000700, 24.99999999999999600, -25.00000000000000000 ) ) ;
#124 = VECTOR ( 'NONE', #125, 1000.000000000000000 ) ;
#125 = DIRECTION ( 'NONE', ( 2.081668171172168500E-016, -1.000000000000000000, 0.000000000000000000 ) ) ;
#126 = CARTESIAN_POINT ( 'NONE', ( 25.00000000000000400, -25.00000000000000400, 25.00000000000000000 ) ) ;
#127 = LINE ( 'NONE', #128, #129 ) ;
#128 = CARTESIAN_POINT ( 'NONE', ( -24.99999999999999600, -25.00000000000000400, 25.00000000000000000 ) ) ;
#129 = VECTOR ( 'NONE', #130, 1000.000000000000000 ) ;
#130 = DIRECTION ( 'NONE', ( 1.000000000000000000, 0.000000000000000000, 0.000000000000000000 ) ) ;
#131 = CARTESIAN_POINT ( 'NONE', ( 25.00000000000000400, -25.00000000000000400, -25.00000000000000000 ) ) ;
```



```
#132 = LINE ( 'NONE', #133, #134 ) ;
#133 = CARTESIAN_POINT ( 'NONE', ( 25.00000000000000400, -25.00000000000000400, 25.00000000000000000 ) ) ;
#134 = VECTOR ( 'NONE', #135, 1000.000000000000000 ) ;
#135 = DIRECTION ( 'NONE', ( 0.000000000000000000, 0.000000000000000000, -1.000000000000000000 ) ) ;
#136 = LINE ( 'NONE', #140, #141 ) ;
#137 = VERTEX_POINT ( 'NONE', #202 ) ;
#138 = ORIENTED_EDGE ( 'NONE', *, *, #221, .T. ) ;
#139 = VERTEX_POINT ( 'NONE', #203 ) ;
#140 = CARTESIAN_POINT ( 'NONE', ( 24.99999999999999600, 24.99999999999999600, 25.00000000000000000 ) ) ;
#141 = VECTOR ( 'NONE', #142, 1000.000000000000000 ) ;
#142 = DIRECTION ( 'NONE', ( 0.000000000000000000, 0.000000000000000000, -1.000000000000000000 ) ) ;
#143 = FACE_OUTER_BOUND ( 'NONE', #225, .T. ) ;
#144 = PLANE ( 'NONE', #145 ) ;
#145 = AXIS2_PLACEMENT_3D ( 'NONE', #146, #147, #148 ) ;
#146 = CARTESIAN_POINT ( 'NONE', ( -24.99999999999999600, -25.00000000000000400, 25.00000000000000000 ) ) ;
#147 = DIRECTION ( 'NONE', ( 0.000000000000000000, 0.000000000000000000, 1.000000000000000000 ) ) ;
#148 = DIRECTION ( 'NONE', ( 1.000000000000000000, 0.000000000000000000, 0.000000000000000000 ) ) ;
#149 = CARTESIAN_POINT ( 'NONE', ( -25.00000000000000700, 24.99999999999999600, -25.00000000000000000 ) ) ;
#150 = LINE ( 'NONE', #151, #152 ) ;
#151 = CARTESIAN_POINT ( 'NONE', ( -25.00000000000000700, 24.99999999999999600, 25.00000000000000000 ) ) ;
#152 = VECTOR ( 'NONE', #153, 1000.000000000000000 ) ;
#153 = DIRECTION ( 'NONE', ( 2.081668171172168500E-016, -1.000000000000000000, 0.000000000000000000 ) ) ;
#154 = LINE ( 'NONE', #155, #158 ) ;
#155 = CARTESIAN_POINT ( 'NONE', ( 24.99999999999999600, 24.99999999999999600, -25.00000000000000000 ) ) ;
#156 = EDGE_CURVE ( 'NONE', #68, #231, #204, .T. ) ;
#157 = EDGE_CURVE ( 'NONE', #50, #137, #209, .T. ) ;
#158 = VECTOR ( 'NONE', #159, 1000.000000000000000 ) ;
#159 = DIRECTION ( 'NONE', ( -1.387778780781445700E-016, 1.000000000000000000, 0.000000000000000000 ) ) ;
#160 = CARTESIAN_POINT ( 'NONE', ( -24.99999999999999600, -25.00000000000000400, -25.00000000000000000 ) ) ;
#161 = LINE ( 'NONE', #162, #163 ) ;
#162 = CARTESIAN_POINT ( 'NONE', ( -24.99999999999999600, -25.00000000000000400, -25.00000000000000000 ) ) ;
#163 = VECTOR ( 'NONE', #164, 1000.000000000000000 ) ;
#164 = DIRECTION ( 'NONE', ( 1.000000000000000000, 0.000000000000000000, 0.000000000000000000 ) ) ;
#165 = LINE ( 'NONE', #166, #167 ) ;
#166 = CARTESIAN_POINT ( 'NONE', ( -24.99999999999999600, -25.00000000000000400, 25.00000000000000000 ) ) ;
#167 = VECTOR ( 'NONE', #168, 1000.000000000000000 ) ;
#168 = DIRECTION ( 'NONE', ( 0.000000000000000000, 0.000000000000000000, -1.000000000000000000 ) ) ;
#169 = FACE_OUTER_BOUND ( 'NONE', #207, .T. ) ;
#170 = PLANE ( 'NONE', #171 ) ;
#171 = AXIS2_PLACEMENT_3D ( 'NONE', #172, #173, #175 ) ;
#172 = CARTESIAN_POINT ( 'NONE', ( -25.00000000000000700, 24.99999999999999600, 25.00000000000000000 ) ) ;
#173 = DIRECTION ( 'NONE', ( 1.000000000000000000, 2.081668171172168500E-016, 0.000000000000000000 ) ) ;
#174 = ORIENTED_EDGE ( 'NONE', *, *, #120, .T. ) ;
#175 = DIRECTION ( 'NONE', ( -2.081668171172168500E-016, 1.000000000000000000, 0.000000000000000000 ) ) ;
#176 = LINE ( 'NONE', #177, #178 ) ;
#177 = CARTESIAN_POINT ( 'NONE', ( -25.00000000000000700, 24.99999999999999600, 25.00000000000000000 ) ) ;
#178 = VECTOR ( 'NONE', #179, 1000.000000000000000 ) ;
```

#179 = DIRECTION ('NONE', (0.0000000000000000000, 0.0000000000000000000, -1.000000000000000000)) ;
#180 = FACE_OUTER_BOUND ('NONE', #97, .T.) ;
#181 = PLANE ('NONE', #182) ;
#182 = AXIS2_PLACEMENT_3D ('NONE', #183, #184, #185) ;
#183 = CARTESIAN_POINT ('NONE', (24.99999999999999600, 24.99999999999999600, 25.00000000000000000)) ;
#184 = DIRECTION ('NONE', (-1.000000000000000000, -1.38777878078144570E-016, 0.0000000000000000000)) ;
#185 = DIRECTION ('NONE', (1.38777878078144570E-016, -1.000000000000000000, 0.0000000000000000000)) ;
#186 = FACE_OUTER_BOUND ('NONE', #107, .T.) ;
#187 = PLANE ('NONE', #188) ;
#188 = AXIS2_PLACEMENT_3D ('NONE', #189, #190, #191) ;
#189 = CARTESIAN_POINT ('NONE', (-25.00000000000000700, 24.99999999999999600, 25.00000000000000000)) ;
#190 = DIRECTION ('NONE', (0.0000000000000000000, -1.000000000000000000, 0.0000000000000000000)) ;
#191 = DIRECTION ('NONE', (0.0000000000000000000, 0.0000000000000000000, -1.000000000000000000)) ;
#192 = FACE_OUTER_BOUND ('NONE', #104, .T.) ;
#193 = PLANE ('NONE', #194) ;
#194 = AXIS2_PLACEMENT_3D ('NONE', #195, #196, #197) ;
#195 = CARTESIAN_POINT ('NONE', (-24.99999999999999600, -25.00000000000000400, 25.00000000000000000)) ;
#196 = DIRECTION ('NONE', (0.0000000000000000000, 1.000000000000000000, 0.0000000000000000000)) ;
#197 = DIRECTION ('NONE', (0.0000000000000000000, 0.0000000000000000000, 1.000000000000000000)) ;
#198 = LINE ('NONE', #199, #200) ;
#199 = CARTESIAN_POINT ('NONE', (24.99999999999999600, 24.99999999999999600, 25.00000000000000000)) ;
#200 = VECTOR ('NONE', #201, 1000.000000000000000) ;
#201 = DIRECTION ('NONE', (-1.38777878078144570E-016, 1.000000000000000000, 0.0000000000000000000)) ;
#202 = CARTESIAN_POINT ('NONE', (-25.00000000000000700, 24.99999999999999600, 25.00000000000000000)) ;
#203 = CARTESIAN_POINT ('NONE', (-24.99999999999999600, -25.00000000000000400, 25.00000000000000000)) ;
#204 = LINE ('NONE', #205, #206) ;
#205 = CARTESIAN_POINT ('NONE', (-25.00000000000000700, 24.99999999999999600, -25.00000000000000000)) ;
#206 = VECTOR ('NONE', #208, 1000.000000000000000) ;
#207 = EDGE_LOOP ('NONE', (#138, #96, #98, #106)) ;
#208 = DIRECTION ('NONE', (-1.000000000000000000, 0.0000000000000000000, 0.0000000000000000000)) ;
#209 = LINE ('NONE', #210, #211) ;
#210 = CARTESIAN_POINT ('NONE', (-25.00000000000000700, 24.99999999999999600, 25.00000000000000000)) ;
#211 = VECTOR ('NONE', #212, 1000.000000000000000) ;
#212 = DIRECTION ('NONE', (-1.000000000000000000, 0.0000000000000000000, 0.0000000000000000000)) ;
#213 =(LENGTH_UNIT ( ) NAMED_UNIT (*) SI_UNIT (.MILLI., .METRE.));
#214 =(NAMED_UNIT (*) PLANE_ANGLE_UNIT ( ) SI_UNIT ($, .RADIAN.));
#215 =(NAMED_UNIT (*) SI_UNIT ($, .STERADIAN.) SOLID_ANGLE_UNIT ( ));
#216 = UNCERTAINTY_MEASURE_WITH_UNIT (LENGTH_MEASURE(1.000000000000000100E-005), #213, 'NONE', 'NONE');
#217 =(GEOMETRIC_REPRESENTATION_CONTEXT (3) GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT ((#216)) GLOBAL_UNIT_ASSIGNED_CONTEXT ((#213, #214, #215)) REPRESENTATION_CONTEXT ('NONE', 'WORKASPACE'));
#218 = ADVANCED_BREP_SHAPE_REPRESENTATION ('cube', (#85, #219), #217) ;
#219 = AXIS2_PLACEMENT_3D ('NONE', #1, #2, #3) ;
#220 = ORIENTED_EDGE ('NONE', *, *, #234, .F.) ;
#221 = EDGE_CURVE ('NONE', #231, #237, #122, .T.) ;
#222 = ORIENTED_EDGE ('NONE', *, *, #99, .T.) ;
#223 = VERTEX_POINT ('NONE', #126) ;
#224 = EDGE_CURVE ('NONE', #139, #223, #127, .T.) ;

#225 = EDGE_LOOP ('NONE', (#233, #86, #174, #105)) ;
#226 = EDGE_CURVE ('NONE', #50, #68, #136, .T.) ;
#227 = ORIENTED_EDGE ('NONE', *, *, #234, .T.) ;
#228 = ORIENTED_EDGE ('NONE', *, *, #91, .T.) ;
#229 = ORIENTED_EDGE ('NONE', *, *, #99, .F.) ;
#230 = ORIENTED_EDGE ('NONE', *, *, #157, .F.) ;
#231 = VERTEX_POINT ('NONE', #149) ;
#232 = ORIENTED_EDGE ('NONE', *, *, #100, .T.) ;
#233 = ORIENTED_EDGE ('NONE', *, *, #95, .T.) ;
#234 = EDGE_CURVE ('NONE', #87, #68, #154, .T.) ;
#235 = ORIENTED_EDGE ('NONE', *, *, #156, .T.) ;
#236 = ORIENTED_EDGE ('NONE', *, *, #221, .F.) ;
#237 = VERTEX_POINT ('NONE', #160) ;
ENDSEC;
END-ISO-10303-21;