# Manipulating Geometry in a STEP DB from Commercial CAD Systems

Junhwan Kim and Soonhung Han

*Concurrent Engineering* 2004 12: 49
DOI: 10.1177/1063293X04041943

CONCURRENT ENGINEERING: Research and Applications

# Manipulating Geometry in a STEP DB from Commercial CAD Systems

**Junhwan Kim*** **and Soonhung Han**

*Department of Mechanical Engineering, Korea Advanced Institute of Science & Technology, 373-1, Gusong-Dong, Yusong-Gu, Daejeon 305-701, Korea*

**Abstract:** It is difficult to access and share design data among heterogeneous CAD systems. Usually different CAD systems exchange design data using a neutral format such as IGES or STEP. A prototype STEP database has been implemented. The prototype system uses the OpenCascade geometric kernel and the commercial object-oriented database ObjectStore. STEP provides the database schema. The STEP database has been accessed and manipulated from commercial CAD systems of SolidWorks and Unigraphics. The data access module of the commercial CAD system has been developed by using the CAD system's native API, ObjectStore API, and ActiveX.

**Key Words:** collaborative design, commercial CAD, data sharing, STEP database.

## 1. Introduction

It is necessary to share design information within a manufacturing company through the Internet, because the company needs to exchange product information with parts suppliers and collaborating partners in addition to sharing information among internal departments. However, it is difficult to share and exchange such information because of different data formats and structures, different hardware platforms, different data storage systems, different operating systems, or different network architecture. Most commercial CAD systems have their native databases that support specific objectives and functions to optimize performance. There are various storage methods as well as various data structures for the storage of CAD data.

To share the geometric information between PDM (product data management) systems and CAD systems, CAD vendors provide interfaces or extra modules for a collaborative design. As the data storage is supported by the file system of an operating system, data is usually exchanged through the neutral formats between heterogeneous CAD systems. If the geometric database of a CAD system can be accessed by using API (application programming interface) of another CAD system, the model exchange would be more efficient than using a neutral format such as STEP (standard for the exchange of product model data) physical file.

As an international standard for product data exchange, STEP is intended to provide information for all engineering applications regardless of hardware, software, or processes [1]. Recently, there are implementations which apply STEP in engineering data integration [2], PDM [3], shipbuilding [4,21], etc.

In this paper, the STEP data structure has been adopted to share data among commercial CAD systems. A STEP database has been constructed to overcome redundancy and heterogeneity among diverse data sets, and a prototype CAD system named DinaCAD has been implemented based on the STEP database. A systematic method of uploading the STEP data into the database and accessing the geometric information from commercial CAD systems is proposed. The following functions have been implemented and tested:

- STEP database: A CAD database has been constructed based on the STEP schema;
- Concurrent access: Different commercial CAD systems can access the database;
- Selective access: Users can query and retrieve a selective portion of the whole data set;
- Filtering: Data can be filtered with specific property.

## 2. Related Research

### 2.1 Literature Review

Various researches have applied DBMS (database management system) to CAD data management since 1980s. In the late 1980s, Meier [5] and Hader [6] applied

*Author to whom correspondence should be addressed.
E-mail: everwind@icad.kaist.ac.kr

1063-293X/04/01 0049–9 $10.00/0 DOI: 10.1177/1063293X04041943

a relational database to solid modeling. Limitations of storing solid models into a relational database are described and methods for efficient retrieval have been presented. Kim [7] shows that the object-oriented database is efficient for CAD environment. But sometimes, relational database is used for performance reason. Häder [6] defined the CSG (constructive solid geometry), B-rep (boundary representation), the control point model respectively, and based on a spanning model that can represent the above three models, the solid model database has been constructed using a RDB (relational database), and he then described how to support the database in terms of structure and efficiency.

The *OMG CAD Service* [8] defined an interface API to effectively deliver the geometric information to downstream applications using IDL (interface definition language) of CORBA (common object request broker architecture). Morris presented an implementation of testing system for STEP and OMG standards [9]. Hardwick suggested the three layers of protocol – distribution, abstraction, and management. In the abstraction protocol, the SDAI–C++ binding is implemented through the Express–IDL–C++ mapping [10].

The NIIIP (national industrial information infrastructure protocol) project provided the protocol for data sharing through the Internet. STEP service is implemented as a part of the NIIIP model. The data saved according to the STEP format can be used throughout the virtual enterprise environment [11]. Kreb implemented the STEP database using DBMS such as INGRES, Informix, Postgres, and ObjectStore, and also implemented the SDAI (standard data access interface) interface that can access the STEP database [12].

There are some well-noted researches about STEP standard data access interface. Botting presented the analysis about STEP standard data access interface using formal methods [13], and Liu developed SDAI-based common access interface for object-oriented DBMS [14]. HLDAI (high-level data access interface) [15] does not directly deal with AIM (application interpreted model) of an AP (Application Protocol), but develops APM (application program model) which is an application-oriented model and a higher-level interface than SDAI.

## 2.2 Comparison with Previous Researches

Two differences can be described from previous researches. First is the sharing of the STEP database, and second is the sharing of the CAD database. A STEP-based OODB, which is the native storage of the ship structural CAD system, is used as the CAD database. Previous researches for the STEP file access from a PDM do not focus on the sharing of geometric information. There are researches that simply visualize STEP data by using a tool such as VRML [16]. Table 1 shows the comparison with previous researches.

**Table 1. Comparison with previous research.**

| | No Graphics Library | With Graphic Library or VRML etc. | Geometric Kernel or CAD API |
|---|---|---|---|
| STEP File Access | [1] | [16] | Commercial CAD Commercial translator |
| STEP DB Access | [2,11–13,17,18] | [19] | This research |

We have implemented a STEP database [20]. The database can be accessed from external CAD systems. We can access the geometry and geometric properties including features and domain-related entities because a STEP AP is used as the database schema. When we access a CAD database which is based on STEP, explicit translators are not necessary. The concept of sharing information among heterogeneous systems using a STEP database is not new [15,17–19]. But we have dealt with geometric entities in a STEP-based OODB, AP 218 of CAD systems is used to implement the system and to verify test cases with shape characteristics. ActiveX is used to test accessing the STEP database from the commercial CAD API environment.

Secondly, STEP files are used and accessed as the primary repository in previous researches. The STEP database stores STEP files which are translated from commercial CAD systems. But in this research, the STEP database is the primary storage of the CAD systems. We design the database based on the STEP schemas, and use it as the native storage of the prototype CAD system [20]. We can create CAD models and save the data directly into the database.

Previous researches access geometric information only to visualize the product [19], but we need to access the data by the mode that can be applied for the design process. Whereas the visualization can treat the model as a single entity, we should access and modify the geometric model selectively, entity by entity. The database in a commercial CAD system is optimized for the internal data structure. A database based on the standard schema such as STEP is usually an external database for the data sharing. The database in this research has both roles of the native storage and the external data sharing. We have focused on CAD data sharing, query by attribute, and query by relation. The geometric data in STEP-based OODB can be revised by external system such as SolidWorks client module using its API. Three "access modes" and two "modification modes" are presented in the following sections.

## 3. Database Access

### 3.1 Framework of the System

The types of data sharing can be classified into (1) data sharing among CAD systems of the same brand, (2) data sharing among different commercial CAD systems such as SolidWorks, Unigraphics, and (3) access to the CAD database from programs other than CAD systems. This paper is focused on the second type of data sharing. The system construction process of this research is divided into three phases:

Phase 1, construct the STEP DB as the native storage for the prototype CAD system DinaCAD using a commercial DBMS [20].
Phase 2, access the STEP DB from a commercial CAD system.
Phase 3, make use of the queried data for the design process.

The design information within the database of DinaCAD can be accessed from a commercial CAD. The framework of the database is shown in Figure 1, and the system is composed of the following:

- Base module of DinaCAD which is made of OpenCascade [20].
- CAD database: It is the native storage of DinaCAD and also the shared database [20].
- The *ActiveX Server* module to access the database of DinaCAD.
- The *Visual Basic* client based on SolidWorks API and *ActiveX Interface*.
- The client add-in based on Unigraphics Ver.18 and *UG OpenAPI*.

Table 2 shows the implementation environment. Both of OpenCascade and ObjectStore provide the C++ interface, so that C++ language is used for the implementation. In order to link the *Visual Basic API* of SolidWorks with the database API, ActiveX Server is programmed using the C++ language. Through the COM interface, it is possible that server can be accessed by the *Visual Basic API* of SolidWorks. The *data bulkload* module (left middle of Figure 1) is used to upload massive instance data into the database. To utilize data inside the designer's CAD system after accessing and retrieving from the database, the database API and the CAD API should be harmonized.

### 3.2 STEP Database as the Native Storage

The DinaCAD system which has the STEP DB as the native storage has been implemented to support the collaborative design of ship structure. A file-based CAD system cannot support concurrent accesses to the model data. The term, *Dina* means "database interface as the native storage". Whereas most of the commercial CAD systems have the file-based storage structures, DinaCAD uses the ObectStore database as the storage system and STEP is used as database schema. In addition to functionalities without database, the concurrent access to geometric data is possible. STEP file is just

***Table 2. Implementation environment.***

| Component | Environment |
|---|---|
| Operating System | Windows 2000 |
| Geometric Kernel | OpenCascade 3.1 [23] |
| Database | ObjectStore 6.0 [24] |
| Programming Language | C++, Visual Basic, ActiveX, |
| External CAD | SolidWorks 2000 VB API [25] |
| | UG 18 Open C++ API [26] |
| STEP Library | ST-Developer 8.0 Win. [27] |

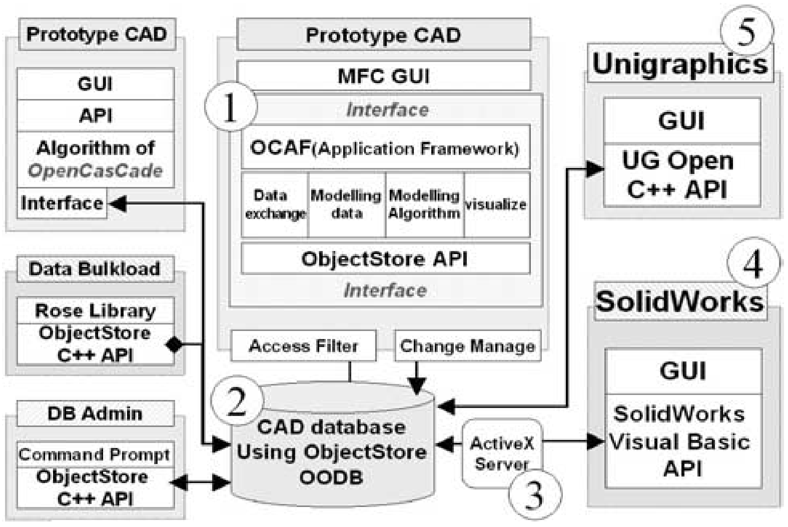

**Figure 1.** System framework.

a dump of the current view that can be taken by the result of query from database and cannot be shared concurrently.

A STEP database can be implemented using a commercial STEP-DB adapter such as ST-Oracle or ST-ObjectStore. However, in this paper, a STEP OODB has been implemented referencing the AP218 schema, the ISO standard for ship structure, and using the *Rose* library of STEP-Tools co. The linkage between the database and the geometric modeling kernel is synchronized by the encapsulated interfaces [20].

With the *Express to C++ compiler* included in ST-Developer, the source code of C++ classes is generated from the schema which is written in EXPRESS. Each header file corresponds to one entity of the schema. Using those header files, the ASCII schema file is created and converted to a binary schema file using the *ossg* OjectStore schema compiler. The binary schema file can link the other object code of C++ logic. Its file extension is generally*.adb*. The ASCII schema file includes the header file such as the *ostore/coll.hh*, and the macro *OS_MARK_SCHEMA_TYPE* (*db_circle*). It represents the whole database schema.

Figure 2 shows the client GUI of DinaCAD. DinaCAD supports basic functions such as geometric primitive modeling, undo, and redo. The *deck plate* and the *girder plane* of the *Midship* section model are displayed on the upper-left window. They are retrieved from the STEP DB through the *filter by property*. It is possible to retrieve only the necessary portion out of the whole database by a filter. Table 3 shows the filter types which are supported by DinaCAD. There are several filter modules such as geometric properties, region, and object type. After a search operation, results of various forms can be given to users. It can be either displayed by the client CAD system through the main memory, saved as a STEP file, or saved as the native CAD file of commercial vendors.

Three access modes to *DinaCAD database* from commercial CAD systems are listed in Table 4. Previous works [16,19] correspond to the *view mode* because they display STEP geometry using a graphic library. In the *view mode*, it simplifies the search process of geometry before it is retrieved. For the *Midship* section model of Figure 2 which is composed of hundreds of parts, it is possible to retrieve the parts without nongeometric information in the view mode. Using a CAD API, they can be merged into one shell, and it makes the viewing operation faster. In the *real time operation mode*, users can edit and change the model. Only one designer can perform the editing operation at the same time and others are allowed for the *read only* operation during the editing.

## 4. Implementation and Experiments

### 4.1 DB Access from SolidWorks Using an ActiveX Server

The DinaCAD database can be accessed from SolidWorks through the *Visual Basic* language interface of the API. The ActiveX Server which has the access

**Table 3. Filter types.**

| Filter Type | Usage Examples |
|---|---|
| Geometric property | Retrieve the deck, girder, etc. |
| Region | Retrieve the region that is $X < 5\,\mathrm{m}$, $Y > 10\,\mathrm{m}$, $3\,\mathrm{m} < Z < 5\,\mathrm{m}$ |
| Object type | Retrieve the *Panel_system* entity |

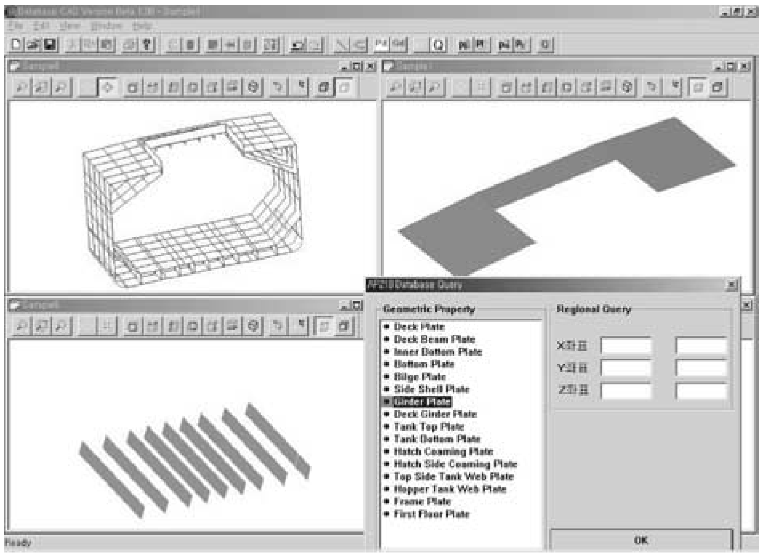

**Figure 2.** Graphical user interface of the DinaCAD client.

***Table 4. Access modes of DinaCAD database from an external CAD.***

| | Real Time Operation Mode | View Mode | Export Mode |
|---|---|---|---|
| 1 | Program start, Database initialization, Database connection, Create os_typespec | | |
| 2 | ObjectStore transaction start | | |
| 3 | Search the root, or create the root in case of the new model creation | Search the root | Search the root |
| 4 | Create a new document of the target CAD system | | |
| 5 | Deliver the necessary portion of model to the designer's workspace through filter | | |
| 6 | Model creation or visualization using CAD API | Model creation using CAD API | Model creation using CAD API |
| 7 | Arrange the entity set with rounding loop, Instance creation and modification, Save the modified result | | |
| 8 | ObjectStore transaction end | | |
| 9 | Repeat steps 2–8 by starting another transaction | Model simplification using CAD API | Database close |
| 10 | Database close | Visualization in CAD; Database close | Save as a file of the commercial CAD system; Open the file to utilize |

logic to the database has been implemented using the Visual C++ language. The client which works together with the ActiveX Server has been implemented using the SolidWorks API and the Visual Basic language.

In case of the file-based data structure, the designer should open the whole file for any operation. But in case of database, the designer can connect to the database, and start a transaction, and retrieve the necessary portion, and perform modifications, and close the transaction. The ActiveX Server allows us to use all the API functions of the database with the Visual Basic language interface since the constructors and the methods of the relevant scheme are created as a COM object.

The version of the database schema of ObjectStore is being updated regularly. If we do not use the same version of schema, we cannot access the database constructed previously. In order to use the same version of schema, we have used the binary schema file (*.adb* extension), which was created in Section 3.1. Using *Object Designer* which is provided by vendor [25] for the data modeling, the binary schema file of DinaCAD is imported. The current version of the schema file for the ActiveX control is generated from the binary schema, and C++ methods for ActiveX Server are created. The *Object Designer* provided by ObjectStore basically creates the *Get* method for getting the relevant objects and the *Set* method for setting the relevant objects. The ActiveX Server physically assumes the form of DLL (dynamic link library) of Windows 2000.

In the *view mode*, a trim line of the model has been searched from the database, and rectangles are created and visualized using the CAD API. Planes are searched in the *real-time operation mode*. Figure 3 shows the result and Table 5 shows the overall operation sequence.

- When a user calls the Visual Basic method, ActiveX Server interface *ObjectStore.OpenDatabase* is called in order to access the database.
- The transaction of ObjectStore starts.
- Using the *.Value* method with the root name, the root is found. The root is the starting point for searching collections.
- To bring the result of accessing the work space into the CAD system, a new document of SolidWorks is created.
- Suitable objects are selected while collection objects are being searched along the loop.
- The selected object is regenerated using the adequate CAD API. At this time, the modification operation is available.
- After regeneration, the *rebuild* method is called in SolidWorks.
- The transaction of ObjectStore stops.

The mapping table between the methods of DinaCAD, which have the DB access methods as the substructure, and the methods of the commercial CAD API should be predefined. Table 6 shows a part of the mapping table. For example, when you want to get a new model from the database and regenerate the model in each CAD system, you can use the *New_database* method of DinaCAD, *UF_PART_new* OpenAPI method of Unigraphics, or *CreateObject* Visual Basic SolidWorks API of SolidWorks. When you want to create a line, you can use *UF_CURVE_create_line*

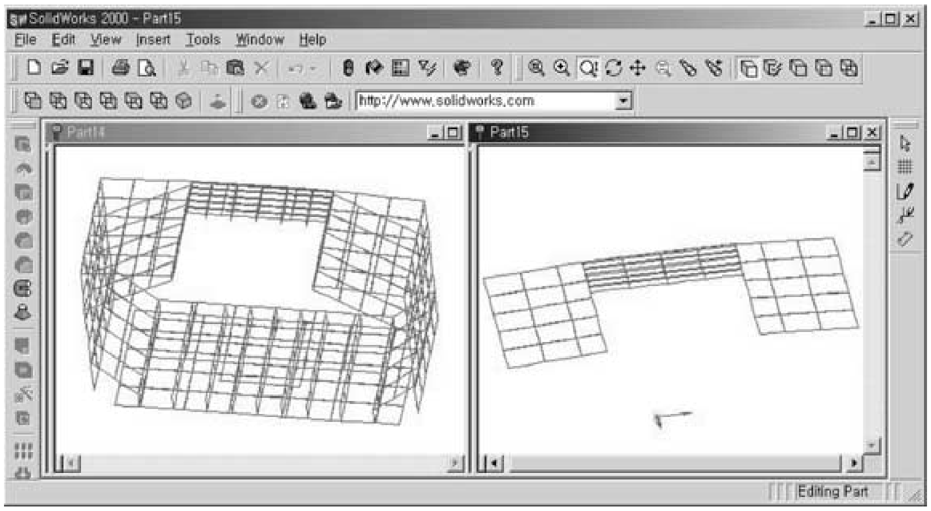
SolidWorks 2000 - Part15
File Edit View Insert Tools Window Help
http://www.solidworks.com
Part14
Part15
Editing Part

**Figure 3.** Database access from SolidWorks 2000.

***Table 5. Database access from SolidWorks.***

| | | | DB API | CAD API |
|---|---|---|---|---|
| 1 | Call ActiveX server | ObjectStore.OpenDatabase | O | |
| 2 | Transaction starts | ObjectStore.BeginTrans | O | |
| 3 | Search root | .Value("root_0", Getdb_trimmed_planeClass) | O | |
| 4 | Create new SolidWorks | CreateObject ("SldWorks.Application") | O | |
| 5 | Retrieving using cursor | For each item in PlineRoot.children ~ Next item | | |
| 6 | SolidWorks entity creation function | Part.SketchRectangleAtAnyAngle | | O |
| 7 | Visualization of memory data | Part.EditRebuild | | O |
| 8 | Transaction end | ObjectStore.CommitTrans | O | |
| 9 | Manipulate in CAD | | | O |

***Table 6. Mapping table between DinaCAD functions and commercial CAD functions.***

| | | DinaCAD | Unigraphics | SolidWorks |
|---|---|---|---|---|
| 1 | New model | New database | UF_PART_new | CreateObject ("SldWorks.Application") |
| 2 | Open | Open database | | OpenObject (".prt") |
| 3 | Primitive block | ODIS_Create_Block | | |
| 4 | Line | ODIS_Create_line | UF_CURVE_create_line | Part.SketchLine |
| 5 | PolyLine | ODIS_Create_line | UF_CURVE_line_t | Part.SketchRectangleAtAnyAngle |
| | | | UF_CURVE_create_line | Part.SketchLine |
| 6 | Plane | ODIS_Create_Plane | UF_Surface_create_plane | |

method of Unigraphics, or *Part.SketchLine* method of SolidWorks.

## 4.2 Access from Unigraphics for Data Export

There are differences between the two cases – Figures 3 and 4. In SolidWorks, the information from database is directly visualized by GUI using API. In Unigraphics, that information can be presented accessed through the *.prt* file which is stored after creating entities using API. This fact shows that the result of accessing entities through the filter can be made in various forms. In SolidWorks, input arguments of a function is the attributes of database, and in Unigraphics, however, input, and arguments are the related structures which are provided by the *UG-OpenAPI* and need to be predefined by the API program.

Because Unigraphics provides the C++ API, the extra server is not necessary. Data is obtained by accessing the database with the ObjectStore API. PolyLines can be generated by the *UG OpenAPI* function, which supports to create *Line* as shown in Figure 4. The left side of Figure 4 shows the whole *Midship* section stored in the database, and the right side shows the result

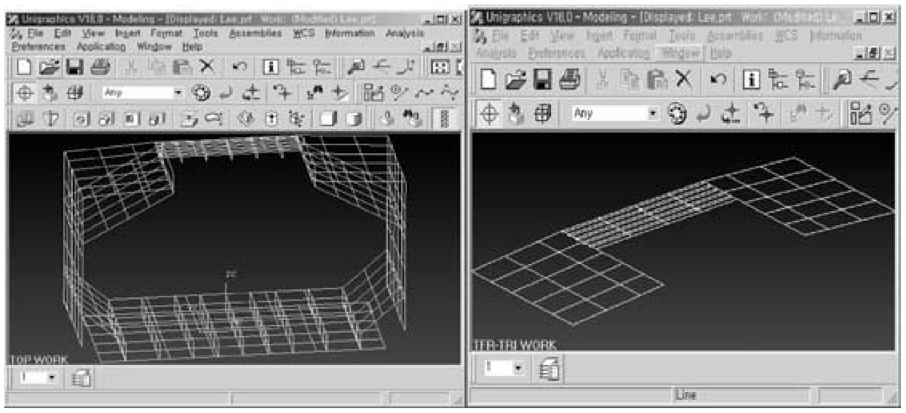
Unigraphics V18.0 - Modeling
File Edit View Insert Format Tools Assemblies WCS Information Analysis
Preferences Application Window Help
TOP WORK
TFR-TRI WORK
Line

**Figure 4.** Database access from Unigraphics Ver.18.

***Table 7. Database access from Unigraphics.***

| | Process | Function Name | DB Func. | CAD Func. |
|---|---|---|---|---|
| 1 | New part | UF_PART_new | | O |
| 2 | Database open | os_database::open | O | |
| 3 | Transaction begin | OS_BEGIN_TXN | O | |
| 4 | Find root | db->find_root | O | |
| 5 | Define cursor | os_Cursor | O | |
| 6 | Structure for UG *Line* | UF_CURVE_line_t | | O |
| 7 | Create line in cursor | UF_CURVE_create_line | | O |
| 8 | Transaction end | OS_END_TXN | O | |
| 9 | Database closed | db->close() | O | |
| 10 | Save result as UG *Part* | UF_PART_save() | | O |

of taking only *Deck Plates* from the *Midship* section. The detail access process is made of the database API and the Unigraphics API, which is shown as Table 7.

1. A new part is generated using the *UF_PART_new* command of the UG API.
2. The database is opened with the ObjectStore API.
3. The transaction of ObjectStore starts.
4. Find the predefined root preserved in the database, which is the starting point to find the stored model.
5. Find objects which should be brought into the workspace, by declaring the cursor and circulating the collection objects.
6. The structure of UG, *UF_CURVE_line_t*, is generated from the entities achieved from the database.
7. The line is created using UF_CURVE_create_line.
8. The transaction is closed.
9. When the whole process is finished, the database is closed, and the part is stored by *UF_PART_save*().

The process generates the *.prt* file after storing the part.

### 4.3 Modifying Database Entities from Outside

It is easy to change nongeometric properties, but in case of geometric entities, only the limited attributes of primitive solid, polyline, and B-spline surface can be modified. There are two modes where the *modify* operation is possible. The first mode is that designer's workspace is the independent workspace allocated only for the design change. Except the changes, that workspace does not affect other design data. The independent workspace is the top node of the data structure. In this prototype implementation, *Plate_Design_definition* or *non_manifold_surface_shape_representation* is the top node where the data is represented as the STEP data or the reference data of that top node.

For the second mode, because the workspace is not an independent workspace, only the entity that has the predefined relationship can be modified. For the change management after modifications, the parameter that affects entities outside of the workspace should be monitored. Figure 5 shows an example of the 1:1 relationship of the CAD data based on the ObjectStore relationship module.

The current implementation focuses on the geometric entities. The relationship is given that two plates share one edge through the GUI. The relationship can be modified during the modeling process using the GUI. The ObjectStore relationship macro *os_relationship_1_1* has been used for the implementation. Two adjacent rectangular faces share one edge. When coordinates of the shared edge are changed, both planes have been changed, and the adjacency relationship will be kept.

## 5. Conclusion

An object-oriented STEP database has been constructed using the STEP schema. A method of accessing geometric data from commercial CAD systems has been proposed and implemented. Two commercial CAD systems of Unigraphics and SolidWorks have accessed the database and manipulated the product model. Designers can bring the desired workspace of a product model into their CAD systems. After manipulation,

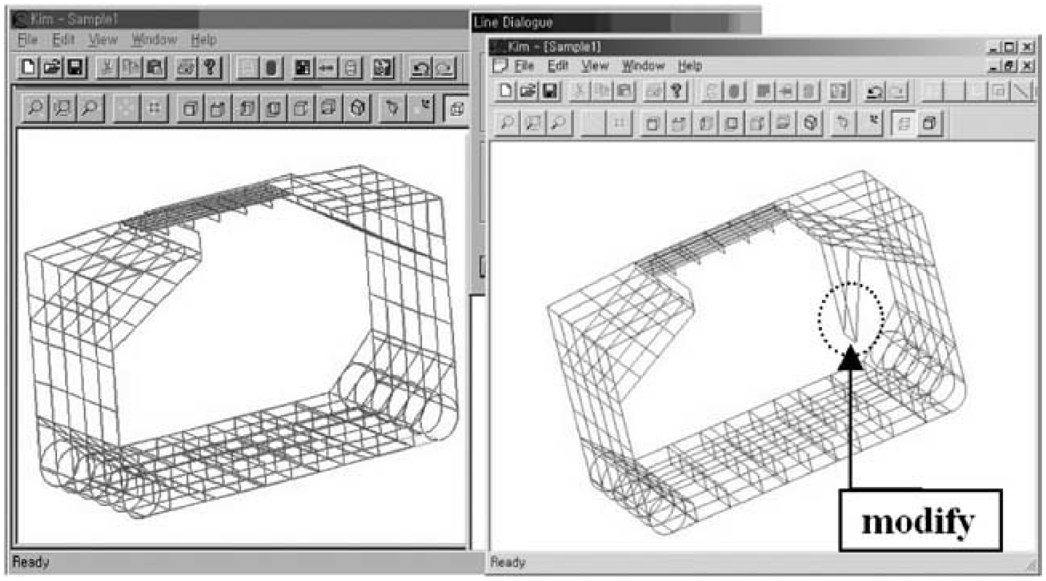

**Figure 5.** An example of relationship module.

designers can store it back into the database. Sharing of product data among different organizations, different design stages, and different CAD systems can be achieved through the STEP database.

In this implementation, the database access module is made up of the minimal set of methods. In order to apply it to complex models, more mapping tables must be added. This paper addresses the domain of ship design. If other industrial domain should be addressed, the relevant AP (application protocol such as AP203 or AP214 for automotive industry) should be used. The results of this paper can be used as the foundation of a distributed CAD or an ASP (application service provider) design environment which should be based on standards such as STEP, OMG CAD Service, or OMG PDM-enabler. Also the geometric data access performance of a STEP-based CAD database should be studied before applying STEP database facility to the industry application [22].

## References

1. An, D., Leep, H.R., Parsaei, H.R. and Nyaluke, A.P. (1995). A Product Data Exchange Integration Structure Using PDES/STEP for Automated Manufacturing Application, *Computers Ind. Engng*., **29**(1–4): 711–715.
2. Yeh, Shen-Chou and You, Chun-Fong (March 2000). Implementation of STEP-Based Product Data Exchange and Sharing, *Concurrent Engineering: Research and Applications*, **8**(1): 50–60.
3. Zhang, Yanping, Zhang, Chun and Wang, H.P. (June 1998). Interoperation of STEP Application Protocols for Product Data Management, *Concurrent Engineering: Research and Applications*, **6**(2): 161–168.
4. Shin, Yongjae and Han, Soon-Hung (1998). Data Enhancement for Sharing of Ship Design Models, *Computer Aided Design*, **30**(12): 931–941.
5. Meier, Andreas (July 1986). Applying Relational Database Techniques to Solid Modeling, *Computer-Aided Design*, **18**(6): 319–326.
6. Häder, Th., Hubel, Ch. and Mitschang, B. (1988). Information Structures and Database Support for Solid Modelling, In: *Proc. of Int. Workshop on Theory and Practice of Geometric Modelling*, Blaubeuren, Germany.
7. Kim, W. (October 1990). Object-Oriented Database Support for CAD, *CAD*, **22**(8): 469–479.
8. OMG (2001). CAD Services Specification, http://www.omg.org.
9. Morris, K.C. and Flater, David (2000). Design of a Flexible, Integrated Testing System for STEP and OMG Standards, *Computer Standards & Interfaces*, **22**, 297–305.
10. Hardwick, M., Spooner, D., Rando, T. and Morris, K. (January/February 1997). Data Protocols for the Industrial Virtual Enterprise, *IEEE Internet Computing*, **1**(1): 20–29.
11. Hardwick, M. and Spooner, D. (Sept. 1998). STEP Services for Sharing Product Models in a Virtual Enterprise, In: *Procceedings of ASME DETC98/CIE-5518*, Atlanta, GA.
12. Krebs, T. and Luehrsen, H. (1995). STEP Databases as Integration Platform for Concurrent Engineering, In: *Proc. 2nd International Conf. on Concurrent Engineering (McLean, Virginia, August 23–25)*, Johnstown, PA: Concurrent Technologies Co., pp. 131–142.
13. Liu, Thu-Hua, Trappey, Amy J.C. and Lin, Chii-Shi (September 2000). Development of SDAI-Based Common Access Interface for Object-Oriented DBMS, *Concurrent Engineering: Research and Applications*, **8**(3): 230–241.
14. Botting, Richard M. and Godwin, Anthony N. (1995). Analysis of the STEP Standard Data Access Interface Using Formal Methods, *Computer Standards & Interfaces*, **17**: 437–455.
15. ISO TC184/SC4/WG11 N039 (1997). The EXPRESS-HLDAI Mapping Language.

16. Kim, Namkug, Kim, Yeonho and Kang, Sukho (1997). Subdivision Method of Converting STEP into VRML on Web, *Computers Ind. Engng*, **33**(3–4): 497–500.
17. Loffredo, David (1998). Efficient Database Implementation of EXPRESS Information Models, PhD Thesis, RPI.
18. Sun, Jianhong and Hardwick, Martin (Sept. 1999). Building an Integrated Large Scale STEP Database for Virtual Enterprises, In: *Proc. of ASME DETC99/DFM*, Las Vegas, USA.
19. Kim, Cheol-Young, Kim, Namkug, Kim, Yeongho, Kang, Suk-Ho and O'Grady, Peter (March 1998). Distributed Concurrent Engineering: Internet-Based Interactive 3-D Dynamic Browsing and Markup of STEP Data, *Concurrent Engineering: Research and Applications*, **6**(1): 53–70.
20. Kim, Junhwan and Han, Soonhung (Nov. 2003). Encapsulation of Geometric Functions for Ship Structural CAD Using a STEP Database as Native Storage, *Computer Aided Design*, **35**(13): 1161–1170.
21. Shin, Yongjae, Bae, Doo-hwan and Han, Soon-Hung (June 2000). Integration of Heterogeneous CAD Databases Using STEP and the Internet, *Decision Support System*, **28**(4): 365–379.
22. Kim, Junhwan, Han, Soonhung and Kim, Youngdae (Feb. 2003). An Evaluation of Access Performance of STEP-Based CAD Database, *Conference Proceeding of Society of Naval Architects of Korea*, Hanjin Heavy Industry (in Korean).
23. http://www.opencascade.org
24. http://www.objectstore.net/
25. http://www.solidworks.com/
26. http://www.eds.com/products/plm/unigraphics_nx/
27. http://www.steptools.com

## Junhwan Kim

Junhwan Kim is a Guest Researcher at the DPG (Design and Process Group), MSID (Manufacturing System Integration Division), Manufacturing Engineering Laboratory, NIST (National Institute of Standards and Technology), Gaithersburg, MD, USA. He received a BS (1995), MS (1998), and PhD (2003) in Mechanical Engineering from the KAIST (Korea Advanced Institute of Science and Technology). His research interests include CAD database, STEP, Parametric data exchange, collaborative design and Internet-based CAD.

## Soonhung Han

Soonhung Han is an Associate Professor in the Department of Mechanical Engineering at the KAIST. He is leading the Intelligent CAD laboratory (icad.kaist.ac.kr) at the KAIST, and the STEP community of Korea (www.kstep.or.kr). His research interests include the STEP, VR for engineering design, and knowledge-based design system. His domain of interests include automotive and shipbuilding industries. He has a BS and a MS from the Seoul National University of Korea, another MS from the University of Newcastle upon Tyne of UK, and a PhD from the University of Michigan of USA. He is involved in the professional societies of CAD/CAM (www.cadcam.or.kr) and e-Business (www.calsec.or.kr). He is the editor-in-chief of the new web-based journal, International Journal of CAD/CAM (www.ijcc.org).