HAL
open science

Interoperability between Cooperative Design Modeller and a CAD System: Software Integration versus Data Exchange

Huijun Song, Lionel Roucoules, Benoit Eynard, Pascal Lafon

To cite this version:

Huijun Song, Lionel Roucoules, Benoit Eynard, Pascal Lafon. Interoperability between Cooperative Design Modeller and a CAD System: Software Integration versus Data Exchange. Journal for Manufacturing Science and Production, 2006, 7 (2), pp.139-149. hal-02291571

HAL Id: hal-02291571

https://hal.science/hal-02291571

Submitted on 19 Sep 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Science Arts & Métiers (SAM)
is an open access repository that collects the work of Arts et Métiers ParisTech researchers and makes it freely available over the web where possible.

This is an author-deposited version published in: https://sam.ensam.eu
Handle ID: .http://hdl.handle.net/null

To cite this version :

Huijun SONG, Lionel ROUCOULES, Benoît EYNARD, Pascal LAFON - Interoperability between Cooperative Design Modeller and a CAD System: Software Integration versus Data Exchange - Interoperability between Cooperative Design Modeller and a CAD System: Software Integration versus Data Exchange - Vol. 7, n°2, p.139-149 - 2006

Any correspondence concerning this service should be sent to the repository Administrator : archiveouverte@ensam.eu

Interoperability between Cooperative Design Modeller and a CAD System: Software Integration versus Data Exchange

Huijun Song[1], Lionel Roucoules[2], Benoît Eynard[2,a], and Pascal Lafon[2]

[1] University of Calgary, Canada

[2] Troyes University of Technology, France

Abstract:

The data exchange between Computer-Aided Design (CAD) systems is a crucial issue in concurrent engineering and collaborative design. The paper presents research works and techniques dealing with the interoperability of a Cooperative Design Modeller (CoDeMo), aiming at the integration of product lifecycle knowledge, and a commercial CAD system (CATIA V5). Two kinds of approaches are implemented in the considered case of CAD interoperability for exchanging geometric data, respectively: one is based on a traditional static interface, in which STEP AP203 standard is used; the other is based on a dynamic interface, in which Application Programming Interfaces (API) of the targeted CAD system is adopted. Both approaches should enhance the communication, exchange and sharing of product data

[a] Corresponding author: Prof. Benoît EYNARD
Troyes University of Technology
Charles Delaunay Institute - LASMIS - FRE CNRS 2848
12 rue Marie Curie - BP 2060 - F.10010 Troyes Cedex - France
T: +33 3 25 71 58 28 - F: +33 3 25 71 56 75
E: benoit.eynard@utt.fr

between CAD systems for improving concurrent engineering. A comparison between these two approaches is made to show their particular advantages and disadvantages. The development of a translator between the both CAD systems based on each approach has been carried out and evaluated on an assembly case.

Keywords: CAD, Interoperability, Data Exchange, STEP, API

1 Introduction

The performance evolution of CAD systems during the last decades has made industries more and more competitive. Traditional CAD systems dealing mainly with the information on geometric modeling and topological analysis /1,2/ have proven to be excellent tools for designers and production engineers /3/. In recent years, some research works have been carried out in various aspects of product development to reinforce and extend the ability of traditional CAD systems, such as some expert systems oriented to early product design phase /4/, CAE and CAM systems, and so on. However, the models of these Computer-Aided x (CAx) systems are heterogeneous because they have been developed independently /5/. Different CAx systems are often found using different formats to display their results, implying a result file developed by a system can hardly be processed by another system /3/. The problem of the format mismatch causes isolation of these systems. As concurrent engineering and collaborative design requires shorter lead-time and faster answer to the market demand, a close interoperability between systems supporting product

development is needed, and an efficient product data exchange between these CAx systems becomes a crucial issue in order to implement this interoperability /6-8/.

Generally, there are two major approaches for information sharing among different CAx systems. One is the static interface based on standard data exchange such as STEP, IGES (Initial Graphics Exchange Specifications) and DXF (Drawing Exchange Format), which translate models through a neutral file. A snap shot of the model is exchanged. The other approach is to implement the dynamic interface by standardized API /9/. Whereas the static interface deals with the content and structure of data, the dynamic interface provides solid modelling and geometric operations /10/.

A Cooperative Design Modeler (CoDeMo) is a research prototype supporting co-operative and integrated design methodology /11,12/. It mainly allows every design actors to share a unique database owing to a formal exchange network. The modeler provides a graphic user interface to add, edit or modify data. Moreover the modeler manages the shared database in order to realize heavy design tasks as data propagation or data consistency management. In this research work, the interoperability of a Cooperative Design Modeler (CoDeMo) and a commercial CAD system (CATIA V5) is discussed. The objective is on the one hand to show the benefit of the interoperability of different CAx systems; on the other hand to evaluate the possible approaches for this aim.

The interoperability is carried out by translating the product geometric definition from CoDeMo toward CATIA V5. The two approaches of data exchange above mentioned have been implemented in this work. In detail, between the two considered

CAx systems, STEP AP203 standard is used as the static interface and CAA-API (Component Application Architecture Application Programming Interface) is adopted to implement a dynamic interface, respectively. A case study is carried out to test, demonstrate and discuss the efficiency of the proposed methods.

This paper is organized as follows. Section 2 is an introduction to CoDeMo and its integration with CATIA V5. Section 3 briefly describes the STEP standard and Application Protocol 203, then the development of a STEP-based translator is detailed. An outline of CAA-API is presented and the translator based on it is described in Section 4. The two proposed approaches for data exchange are compared in Section 5. Finally, the conclusions and the future works are presented.

2 Interoperability between CoDeMo and CAD System

2.1 Introduction of CoDeMo

Product lifecycle knowledge integration is the kernel and core factor leading to success of concurrent engineering. Its objective is to increase the speed of product development process and to improve the quality of designed product by taking into account a maximum of knowledge and expertise during the engineering design phase /12/. This knowledge comes from all the experts involved in the product life cycle. Moreover, this design methodology allows the simultaneous progress of design tasks gathering all the experts to exchange and communicate on the design project /13/. Multiple-view modelling is used to organize different aspects of product lifecycle knowledge in concurrent engineering /14-16/. CoDeMo is a research prototype aiming

at multiple-view product modeling /17/. It supports designers to carry out a top-down product development process in different views (e.g. technology view, geometric view, manufacturing view, etc.) and build the relationships between those views. In CoDeMo, product data are formalized as features described by their characteristics and behaviors. Features define a knowledge model and each one is specific to a design point of view. The used modeling language in the multiple-view modeling is based on: Components, Links and Relations objects. These three objects structure the product data in order to manage the data consistency. Thus, a good structuring of the product data is obtained. The modeling language structures relationship between product lifecycle knowledge and the product data are formalized with features. Based on this approach, product lifecycle knowledge can be summarized in formalizing information with the modeling language. Indeed, product data are then well known as feature, and well structured as multiple-view breakdown. Figures 1 and 2 respectively show the product model and an example product breakdown.

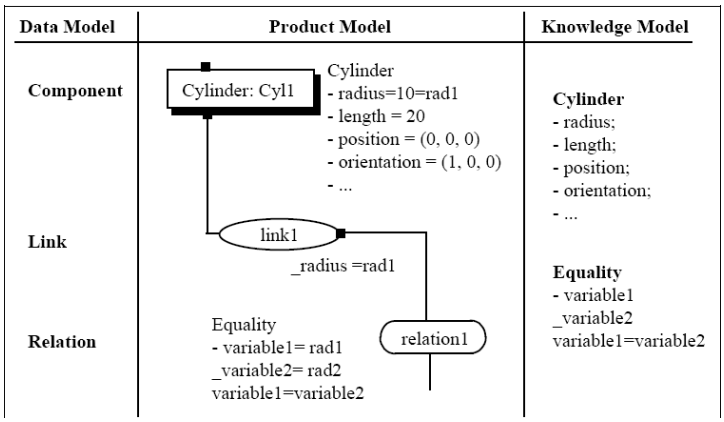

Fig. 1: Relationship between Data Model and Knowledge Model for Specifying Product Model

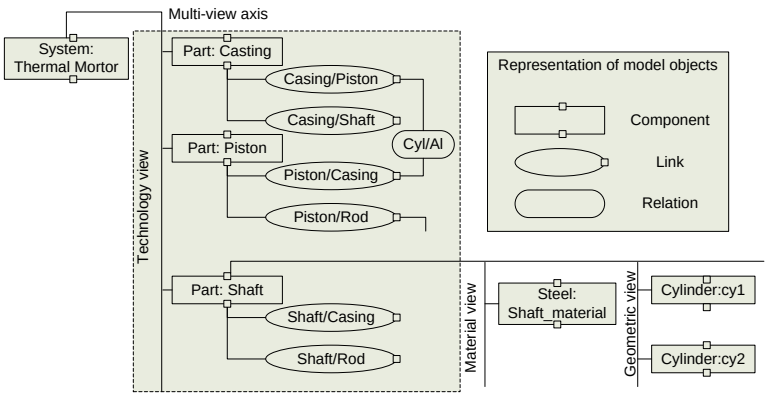

Fig. 2: Example of a Multiple-view Product Breakdown

2.2 Interoperability Framework

In this work, the interoperability between CoDeMo and a commercial CAD system - CATIA V5 (in this work has been chosen regarding the research team know-how and some industrial requirements) is considered, and the data exchange about geometric definition between these two systems will be our concern in this paper. Figure 3 shows the interoperability framework of CoDeMo and CATIA V5 /18/. The proposed framework aims at improving and simplifying the data exchange during the product development process requiring nowadays numerous CAD systems. Those improvements deal with the aid to designers for being more efficient by reducing the translation time, by improving the quality and re-usability of the translated data, and by minimizing the direct data processing by the designers.

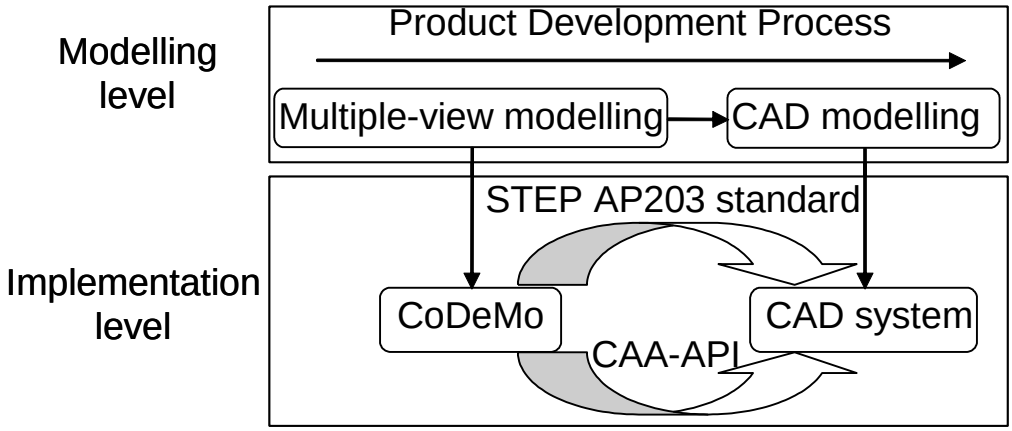

Fig. 3: Interoperability Framework of CoDeMo and CAD system

There are two levels in this framework. The chosen modelling techniques are described on the modelling level: multiple-view modelling is used to embed product lifecycle knowledge and expertise (design, analysis, process planning, manufacturing, assembly, etc.) into the description of form features; traditional CAD modelling is adopted to describe the geometric and topological information and to enable the analysis. Correspondingly on the implementation level, the used modelers include CoDeMo and a CAD system (CATIA V5 in the case study).

Two kinds of approaches are used for exchanging product data between these two systems (modeller). STEP AP203 standard, as a static interface, offers an efficient mechanism of product data exchange between heterogeneous systems /19/. CAA-API of CATIA V5, as a dynamic interface, provides a powerful and flexible way for the integration between Dassault Systèmes software and other systems. In the first way, the STEP neutral file format is used to communicate geometric data. In the second way, a file of CATIA V5 format is directly produced by CAA-API according to the corresponding CoDeMo file.

3 STEP-based data exchange between CoDeMo and CAD systems

3.1 The STEP standard

ISO 10303 is an international standard for the computer-interpretable representation and exchange of product data /20/. The objective is to provide a neutral mechanism able to describe product data through the life cycle of a product, independent of any particular system /21,22/. The nature of this description makes it suitable for neutral file exchange, but also as a basis for implementing and sharing product database and archiving.

AP 203 of ISO 10303 - STEP specifies an application protocol (AP) for the use of product data within a defined context which satisfies an industrial need to exchange configuration- controlled 3D product design data of mechanical parts and assemblies /19/. According to the STEP standard, AP203 data may be exchanged as physical files which are a textual encoding of the product data (ASCII). The syntax and organisation of data in STEP physical files is described in /23/.

3.2 STEP-based translator

Figure 4 shows the STEP-based integration between CoDeMo and the CAD system. The concerned information generated by CoDeMo is translated into STEP file format by a translator, and then, this STEP file can directly be opened by any CAD system implementing STEP interface (in the case study CATIA V5).

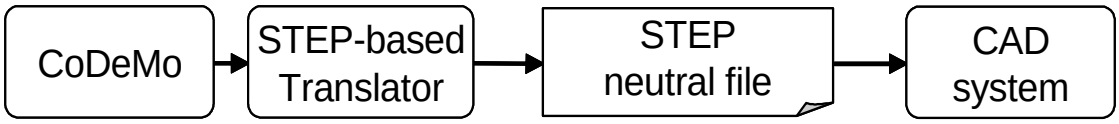

Fig. 4: STEP-based Exchange between CoDeMo and CAD system

In this data exchange, the kernel of the process is the development of the STEP-based translator. This translator has already been developed using MS Visual C++ 6.0. Considering the reusability and extensibility of the module, the operation for generating the STEP codes of a geometric object is encapsulated into a class which provides a compact interface for accessing. The structures, the attributes of the class, are based on the EXPRESS definitions of the entities described in the standard parts. As the translator is developed with the objective of demonstrating feasibility, so it does not aimed to be exhaustive. Currently, its development focused upon the translation of some basic geometric objects and their assemblies.

Here, the benefit of encapsulating the syntax of STEP file into classes is that it is not necessary to care about the details of the STEP codes. The needed data are the specification some required parameters (e.g. diameter of circle, length of cylinder, etc.), the definition the object (e.g. circle, cylinder, etc.) and then the program calls the right member function. In addition, this class hierarchy is easy to extend. A new class, which may be about a part or an assembly, can be derived from some existing class so that it will have the unified accessing interface.

Figure 5 shows the architecture of the STEP-based translator (the second translator CAAAPI-based is described in Section 4). The STEP based translator includes the following three modules:

– **Data reading module for CoDeMo file:** All the data in an output file of CoDeMo are read line by line into the memory in this module. Details of the CoDeMo file can be found in /17/. In fact, only some specific data need to be

translated in this process. So, the data in the memory are parsed in order to extract the required attribute for the translation, which includes some parameters used to call the member function for generating STEP codes in a C++ class. Also, those required data are stored in the memory.

- **Translation module:** This module generates the STEP codes according to data contained in the CoDeMo file. This translation is carried out with the help of an algorithm which establishes the mapping between CoDeMo data and STEP data. These mapping have been established previously based on an analysis of the CoDeMo documentation and the STEP standard parts. The STEP codes are stored in the memory.
- **Writing module:** This module writes the generated STEP codes as an ASCII physical file. This file can directly be read and edited by any CAD system implementing STEP standard interface. Currently, the exchange of parameterized STEP models is under processed /24/. These new developments of the STEP standard will provide much more reusable neutral CAD files.

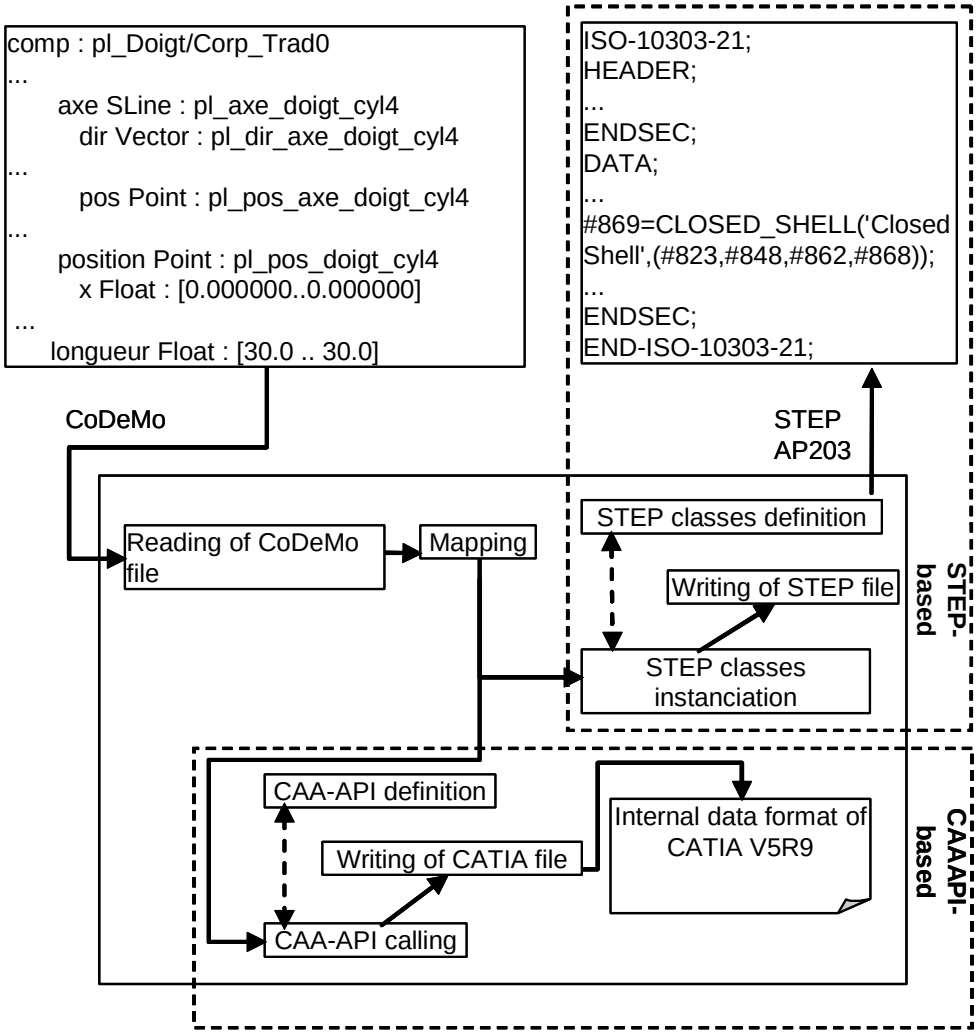

Fig. 5: Architecture of translators (STEP based and CAA-API based)

4 CAA-API based data exchange between CoDeMo and CATIA system

4.1 CAA and CAA-API

Component Application Architecture (CAA) V5 provided by Dassault Systèmes (DS) is a powerful tool for system extension and customization. The integration of DS products (CATIA V5, ENOVIA, DELMIA, etc.) for the product lifecycle management relies on the opened, extensible and modular CAA. CAA products inlcude the two following parts:

- **CAA API (Application Programming Interface):** CAA API formes the basis of CAA development, into which all the interfaces for operating the CAA objects are encapsulated. CAA API consists of API sets for CATIA V5, and others DS products.
- **CAA RADE (Rapid Application Development Environment):** RADE is a visual IDE (Integarted Development Enviroment), which provides a whole group of programming tools. It can not be run independently. In fact, it should be integrated into Microsoft Visual Studio VC++, as a result, some CAA related development tools are added into this enviroment.

In this work, the CAA-API has been used for developing a closer interoperability of CoDeMo and CATIA in a CAD/CAM framework.

4.2 CAA-API based translator

The communication between CoDeMo and CATIA based on CAA-API is a kind of direct connection. No neutral file is created in this translation process. The required internal data format of CATIA V5R9 (e.g. CATPart file, .CATProduct file, etc.) is directly generated according to some necessary parameters included in the CoDeMo file.

The architecture of the CAA-API based translator is also shown in Figure 5 (CAAAPI-based part of the picture). The data reading module for this translator is the same than that of the counterpart of the STEP-based translator. In the translation module, the required parameters extracted from the source CoDeMo file are used to call the CAA-APIs to generate the corresponding CATIA file. In the writing module,

a file with the internal data format of CATIA V5R9 is generated, which can only be read by the CATIA V5 system.

It should be underlined that all the CAA related developments in this work have been carried out without the support of CAA RADE. The environment of Windows 2000 Professional and Microsoft Visual Studio VC++ 6.0 are set manually according to the requirement of CAA development.

5 Case study

A design case study is detailed in this section in order to demonstrate the applicability of the both translators and to discuss their strength and weakness. During this case study, an assembly is considered which consists of two parts (Figure 5). The "shaft" part is defined in the file format of CoDeMo as shown on Figure 6.

```
{ axe SLine : pl_axe_doigt_cyl4
        dir Vector : pl_dir_axe_doigt_cyl4
                z Float : [0.000000..0.000000]
                y Float : [1.000000..1.000000]
                x Float : [0.000000..0.000000]
        diameter Float : [12.0 .. 12.0]
        position Point : pl_pos_doigt_cyl4
                z Float : [0.000000..0.000000]
                y Float : [-20.000000..-20.000000]
                x Float : [0.000000..0.000000]
        length Float : [30.0 .. 30.0]}
```

Fig. 6: Example of CoDeMo File Format

The values of each attribute is represented with an interval, which allows giving tolerance information in design activities, according to the constraints programming approach included in CoDeMo.

At the beginning of the translation process, these CoDeMo codes are read line by line by the data reading module, and then, the required parameters are extracted during the parsing process. In this case, three parameters are obtained: center1(0,-20,0), center2(0,10,0), and radius 12.

5.1 STEP-based translation process

A cylinder object is defined with the extracted parameters and its unified interface is accessed in order to generate the required STEP codes:

CCylindrical_surface_segment Css(center1,center2, radius),

Css.GetSTEPCode(…)

CCylindrical_surface_segment is the class name, which encapsulates the logics for generating a cylinder object. Css is an user defined instance of the class, and the member funtion GetSTEPCode() is the unified interface of the class. A part of the generated physical STEP file is shown on Figure 7.

```
#71=CARTESIAN_POINT('Axis2P3D Location',(- 5.,0.,0.));
#72=DIRECTION('Axis2P3D Direction',(0.,1.,0.));
…
#84=CYLINDRICAL_SURFACE('generated cylinder',#54,6.);
…
```

Fig. 7: Example of the Generated Physical STEP File

These STEP codes are written into the objective file (.stp) by the writing module. The objective file can be read by most of the commercial CAD software (CATIA, Pro/ENGINEER, SolidWorks, etc.).

5.2 CAA-API based translation process

There are two steps in this CAA-API based translation process. The first step is to generate the section 2D circle, and the second step is to create the extrusion from the circle as shown on Figure 8 in which CreateCircle and CreatePad are two CAAAPIs.

```
CatiaCircles[1]= sketchFactory2D->CreateCircle(centre1, radius),
CATISpecObject_var spSpecObj = spPrtFactOnPrtCont->CreatePad(…),
spPadOnSpecObj->ModifyStartOffset(…),
```

Fig. 8: Example of the CAAAPIs Implementation

The result of the CAAAPI-based translator is a file with the internal file format of CATIA. This file can be viewed and edited only throught the CATIA system.

The translation process of the second part in the assembly is similar to that of the first one, and it is not detailed here. Figure 9 shows the translation process of the assembly case in STEP-based and CAAAPI-based approach, respectively. The arrows between the modules of each translator detail the used treatments to read, extract or write data embedded in the various used file formats (CoDeMo, STEP, .CATProduct).

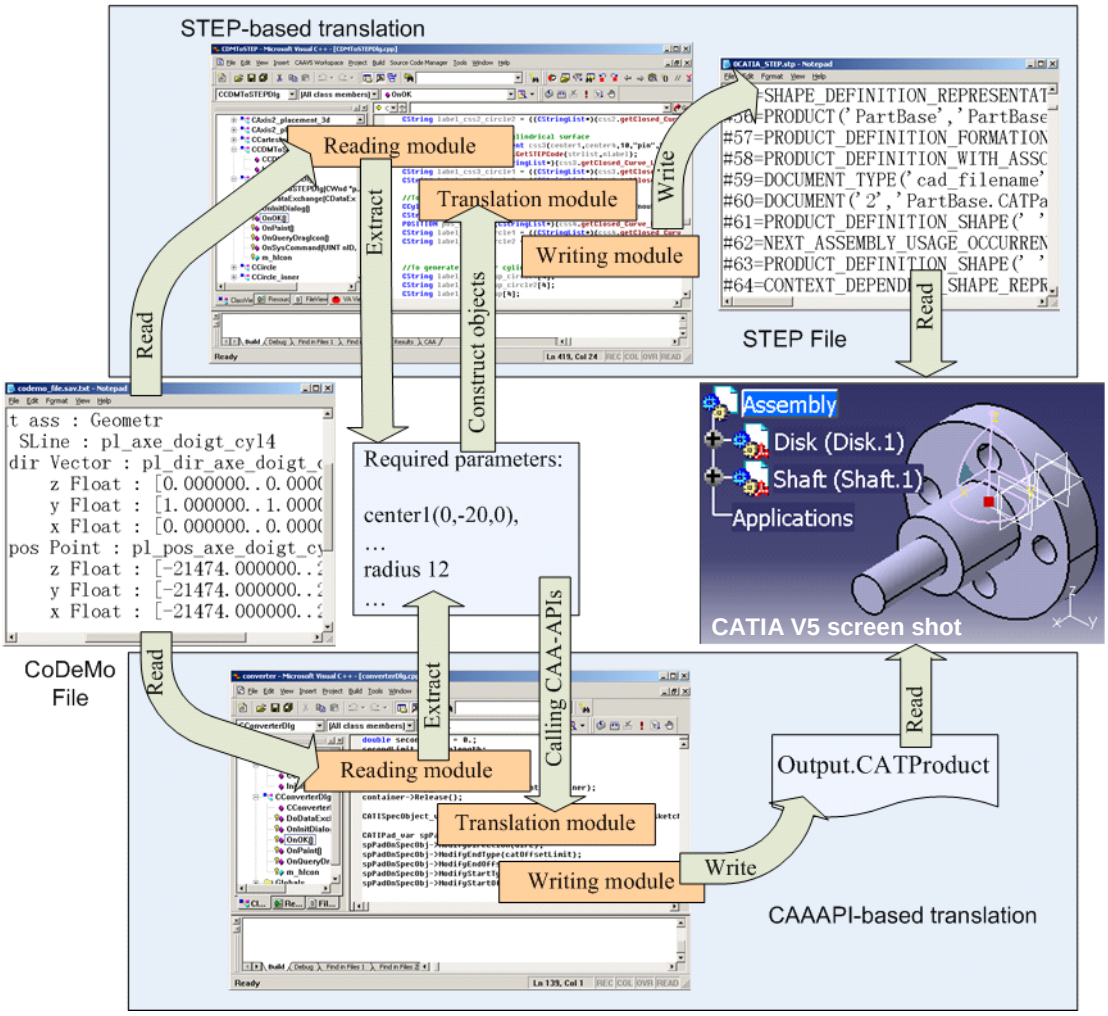

Fig. 9: Translation Processes between CoDeMo and CATIA V5

6 Discussion of the translators development

This section discusses the static and dynamic approaches developed for the interoperability of CAD systems in the following three aspects: generality and functionality, and development skills.

6.1 Generality and functionality

In a static interface approach (e.g. STEP based), the data exchange between different CAD systems is implemented via a neutral file (e.g. STEP file) as a transition. The neutral file is generated according to a data exchange standard, and the

neutral file format is not dependent on any CAD system. Nowadays, most of the commercial CAD systems can process the file compliance with a neutral format like STEP or IGES. So, the translator based on a static interface approach (data exchange standard) provides a robust "write-once, read everywhere" solution for the integration of different CAD systems. Current developments of STEP standard will allow the reuse of exchanged CAD including parameterization. During the data exchange process with a dynamic interface approach (e.g. CAAAPI based), the source file (e.g. CoDeMo file) is translated into the target file (e.g. CATIA file) directly by standardized APIs which are provided with the CAD system. As we know, different CAD systems are often found using different file formats, so the translator based on a dynamic interface approach varies with the related CAD systems. Then, considering the CAAAPI-based translator, it is oriented to the specific integration task, which is, of course, Dassault Systèmes-related. Theoretically, C_n^2 translators are needed for the integration of *n* heterogeneous CAD systems.

Currently, a data exchange standard addresses an extraction of a large number of specific applications. It formalizes the common properties of all the relevant applications. This characteristic of a standard results in the ignorance of the details specific to the applications. This is the prerequisite for the "write-once, read everywhere" solution. Generally, the process for generating a CAD object strongly depends on the used CAD system. In order that the design issues of a CAD system can be shared with other CAD systems, the modeling process has to be ignored in the exchanged file. For example, the STEP file is neutral and it describes only the final

result of the CAD modeling. All the modeling process in the design history (e.g. intermediate issues) can currently not be kept in a STEP file. Obviously, this limitation causes that some parts of the design issues can not be edited. It should be mentioned that the current developments of STEP aims at solving those matters including parameters, constraints or model history /25/.

In the dynamic interface approach, all the operations provided by a CAD system are encapsulated into APIs. All the procedures performed in the environment of a CAD system can recur by programming on APIs. This characteristic makes it possible that the design operations performed in the source CAD system can be translated (via API-based translator) into the counterpart in the target CAD system. So, the information on the design process can be kept between CAD systems. For example, the cylinder object is modeled with two steps in the case study: generate the sketch 2D circle, and create the extrusion of the circle. These two operations are reconstructed in the target CATIA file by CAA-API and the design history can be modified. Figure 10 shows the STEP file and CAA-API based target file opened in CATIA.

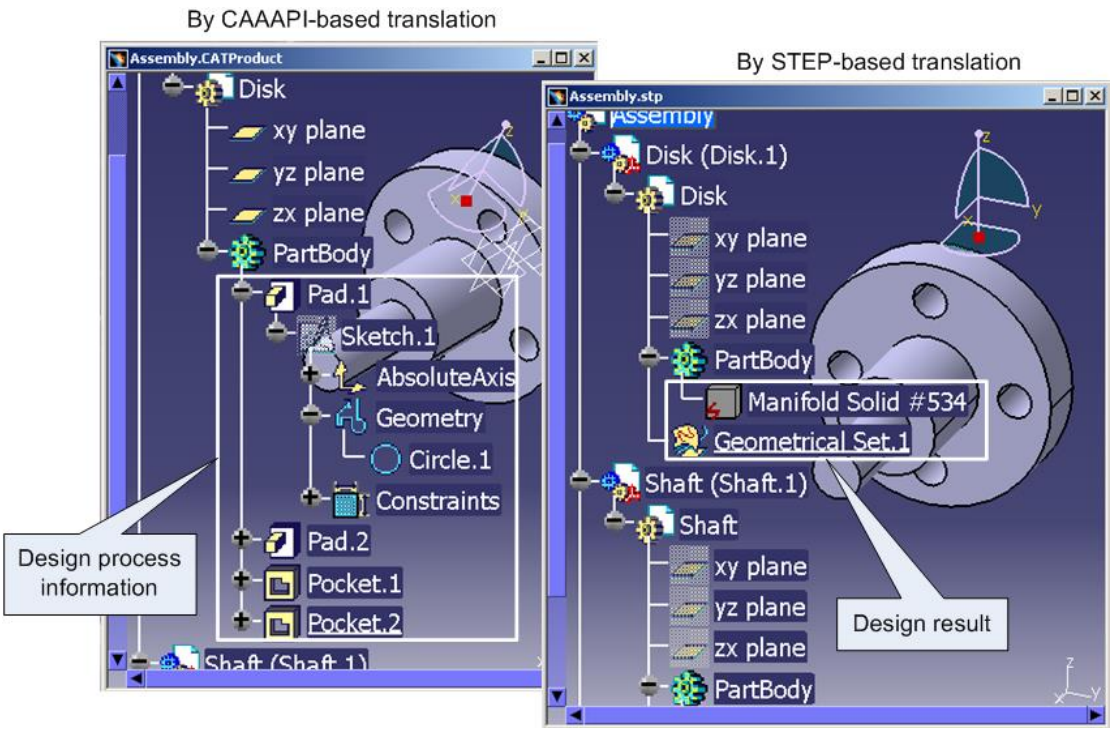

Fig. 10: Target file in CATIA with STEP and CAAAPI Approaches

6.2 Developing Skills

In the specification and programming of a static interface, the developers works are mainly focused onto the redefinitions of the data structures specified in the considered data exchange standard (e.g. EXPRESS definitions of the entities described in the relevant STEP standard parts). So, being familiar with the data exchange standard (STEP, IGES, DXF, etc.) and a programming environment for developers are necessary for this kind of interoperability implementation. Currently, many commercial programming environments are fulfilling this task, such as Visual C++, Visual Basic, and Borland Delphi. Then, the developer has a large freedom to choose a developing tool.

In the development of a dynamic interface, most of the works are concentrated on the API calling process to generate required CAD objects in a development

environment. Peoples in charge of software developments should be accomplished in some API architecture, and a programming environment. Generally, APIs provided by commercial CAD systems are difficult to read and understand, and the corresponding developing tools are specific (e.g. RADE). So the software developers will spend much time on the study of APIs and specific developing tool (e.g. setting the developing environment manually is also a difficult process like that in our work).

Debugging is the most time consuming activity in a program developing process. When developing a static interface, the program debugging can be carried out completely in a developing tool (e.g. considering CATIA V5 case study VC++ and MSDN). When developing a dynamic interface, the program debugging is related to not only the developing tool but also the specific APIs, and the documents (e.g. VC++, MSDN, CAA APIs and CAA encyclopedia).

To summarize, both approaches have their advantages and disadvantages. The main advantage of STEP-based translator is that it provides a real open and generic format to any CAD system while the CAAAPI-based one only addresses the CATIA V5. Regarding the CAAAPI-based translator, it offers valuable possibilities to create many kind of 3D data directly in CATIA V5, and to access to whole the available modules in this system. Then, the approach choice depends on the end-user needs of the application and is specific to the nature of the targeted case study also. Of course, if the system is seen as a black box interfacing with any kind of CAD system, the static interface will be the only relevant choice to be implemented for application interoperability.

7 Conclusions

This paper has detailed the development of two kinds of translators based on a static interface and a dynamic interface for the data exchange for CAD systems interoperability. STEP AP203 standard is used as the static interface, and CAA-API is adopted to implement a dynamic interface between a multiple-view product modeler system - CoDeMo and a commercial CAD system (CATIA V5).

After the translator's development and application, their comparison has led to the following main conclusions:

- Considering a static interface, only the final result of the 3D part modelling can be translated through a neutral file between CAD systems. Then, this kind of approach has strong generality and its implementation would greatly decrease the number of translators in the interoperability environment. The development of the translator is relatively easy, because many commercial programming environments are fulfilling with the encapsulation of data structures of a data exchange standard. It remains complicated to have a full understanding of STEP due to the volume of information and spread of concepts inherent of the standard.
- Considering a dynamic interface, not only the design result but also the 3D part modelling can be shifted between CAD systems. This kind of translator is always for a specific application, because different CAD systems often use different file formats. The development of the translator is relatively difficult,

because the development tool related to a dynamic interface (APIs) is fully specific and APIs architecture is often difficult to read and understand.

Future research works may focus on the combination of static interface and dynamic interface. It includes two aspects: to implement a future STEP schema for recording parameterizations, constraints, and history information; to develop preprocessor and post-processor using API.

References

1. Tomiyama T, Umeda Y, Yoshikawa H. A CAD Functional Design. Annals of the CIRP; 1993; 42(1); 143-146.

2. Umeda Y, Ishii M, Yoshioka M, Tomiyama T. Supporting Conceptual Design Based on the Function-Behavior-State Modeler. Artificial Intelligence for Engineering Design, Analysis and Manufacturing; 1996; 10(4); 275-288.

3. Chao PY, Wang YC. A Data Exchange Framework for Networked CAD/CAM. Computers in Industry; 2001; 44(2); 131-140.

4. Zhang WY, Tor SB, Britton GA, Deng YM. EFDEX: A Knowledge-based Expert System for Functional Design of Engineering Systems. Engineering with Computers; 2001; 17(4); 339-353.

5. Oh Y, Han S, Suh H. Mapping Product Structures Between CAD and PDM Systems Using UML. Computer-Aided Design; 2001; 33(7); 521-529.

6. Zhou S, Chin KS, Xie Y, Yarlagadda PKDV. Internet-based Distributive Knowledge Integrated System for Product Design. Computers in Industry; 2003; 50(2); 195-205.

7. Lee RS, Tsai JP, Kao YC, Lin GCI, Fan KC. STEP-based Product Modeling System for Remote Collaborative Reverse Engineering. Robotics and Computer-Integrated Manufacturing; 2003; 19(6); 543-553.

8. Hwang HJ, Han S, Kim YD. Mapping 2D Midship Drawings into A 3D Ship Hull Model Based on STEP AP218. Computer-Aided Design; 2004; 36(6); 537-547.

9. Choi GH, Mun D, Han S. Exchange of CAD Part Models Based on the Macro-Parametric Approach. International Journal of CAD/CAM; 2002; 2(1); 13-21.

10. Magleby SP, Jackson DB. A Standardized Application Interface for Geometric Modelers. Product Modeling for Computer-Aided Design and Manufacturing. Amsterdam: North-Holland IFIP series, 1991.

11. Tichkiewitch S. De la CFAO à la Conception Intégrée. International Journal of CAD/CAM and Computers Graphics; 2004; 9(5); 609-621.

12. Tichkiewitch S. Specification on integrated design methodology using a multi-view product model. Proc. ASME Engineering System Design and Analysis Conference, Montpellier, France, July, 1996.

13. Ishii K. Modelling of concurrent engineering design. Concurrent Engineering: Automation, Tools and Techniques. New York: Wiley, 1993.

14. Bidarra R, Kranendonk N, Noort A, and Bronsvoort WF. A collaborative framework for integrated part and assembly modeling. Journal of Computing and Information Science in Engineering; 2002; 2(4); 256-264.

15. Bronsvoort WF, Noort A. Multiple-view Feature Modelling for Integral Product Development. Computer-Aided Design; 2004; 36(10); 929-946.

16. Roucoules L, Salomon O, Paris H. Process Planning as An Integration of Knowledge in the Detailed Design Phase. International Journal of Computer Integrated Manufacturing; 2003; 16(1); 25-37.

17. Roucoules L, Tichkiewitch S. CoDE: A Co-operative Design Environment. A New Generation of CAD Systems. Concurrent Engineering Research and Application; 2000; 8(4); 263-280.

18. Song H, Eynard B, Lafon P, Roucoules L. Towards Integration of CAx Systems and a Multiple-View Product Modeller in Mechanical Design. Acta Polytechnica; 2005; 45(3); 26-31.

19. ISO 10303-203. STEP – Part 203: Application protocol: Configuration controlled design of mechanical parts and assemblies. Geneva: International Standard Organization, 1994.

20. ISO 10303-1. STEP – Part 1: Overview and fundamental principles. Geneva: International Standard Organization, 1994.

21. Bloor MS, Owen J. CAD/CAM Product-data Exchange: the Next Step. Computer-Aided Design; 1991; 23(4); 237-243.

22. Gu P, Chan K. Product Modelling Using STEP. Computer-Aided Design; 1995; 27(3); 163-179.

23. ISO 10303-21. STEP – Part 21: Implementation methods: Clear text encoding of the exchange structure. Geneva: International Standard Organization, 1994.

24. Pratt MJ. A new ISO 10303 (STEP) resource for Modeling Parameterizations and Constraints. Journal of Computing and Information Science in Engineering; 2004; 4(4); 339-351.

25. Pratt MJ, Anderson BD, Ranger T. Towards the standardized exchange of parameterized feature-based CAD models. Computer-Aided Design; 2005; 37(12); 1251-1265.