Data Management in Distributed CAx Systems

Dipl.-Ing. Thomas Kilb, Dipl.-Inform. Florian Arnold
Research Group for Computer Application in Engineering Design
University of Kaiserslautern, Germany

Abstract

Interoperability between different CAx[1] systems involved in the development process of cars is presently one of the most critical issues in the automotive industry. None of the existing CAx systems meets all requirements of the very complex process network of the lifecycle of a car. With this background, industrial engineers have to use various CAx systems to get an optimal support for their daily work. Today, the communication between different CAx systems is done via data files using special direct converters or neutral system independent standards like IGES, VDAFS, and recently STEP, the international standard for product data description.

To reduce the dependency on individual CAx system vendors, the German automotive industry developed an open CAx system architecture based on STEP as guiding principle for CAx system development.

The central component of this architecture is a common, system-independent access interface to CAx functions[2] and data of all involved CAx systems, which is under development in the project ANICA[3]. Within this project, a CAx object bus has been developed based on a STEP data description using CORBA[4] as an integration platform. This new approach allows a transparent access to data and functions of the integrated CAx systems without file-based data exchange.

The product development process with various CAx systems concerns objects from different CAx systems. Thus, mechanisms are needed to handle the persistent storage of the CAx objects distributed over the CAx object bus to give the developing engineers a consistent view of the data model of their product. The following paper discusses several possibilities to guarantee consistent data management and storage of distributed CAx models. One of the most promising approaches is the enhancement of the CAx object bus by a STEP-based object-oriented data server to realise a central data management.

1 Industrial Background

Today the automotive industry has to react to an enormous pressure of the global market. A continuously rising product complexity in combination with a drastically increasing number of product variants has to be handled. The answer to these rising demands are the classical four criteria Q, T, C, I: **Q**uality, **T**ime to Market, **C**ost control and **I**nnovation in products. As a consequence, the automotive

---

[1] CAx: Generic term for various CA techniques, e.g. CAD, CAQ, CAE, CAM.

[2] The term *function* is used here as an equivalent for 'something with a certain functionality'. A function can therefore be a (simple) function, a procedure or (in object-oriented programming) a method of a class.

[3] ANICA: **AN**alysis of access **I**nterfaces of various **CA**x systems

[4] CORBA: **C**ommon **O**bject **R**equest **B**roker **A**rchitecture

industry developed the simultaneous/concurrent engineering concepts with various enterprises participating in the resulting process network using different CAx systems.

Besides direct converters, neutral interface formats for data exchange become inevitable in this heterogeneous scenery of systems. Problems due to the data exchange have been reduced by different guidelines and standards (e.g. in Germany developed by the VDA[5]), but data exchange remains affected by loss of information and the need of rework. The urgent need to optimise the CAx communication requires an open and modular world of CAx systems. This does not only concern the CAx system with mainly geometric information but also information about product structure, process data and administration.

Being aware of these problems and trends, the working group "CAD/CAM strategies of the German automotive industry" developed a proposal of an open CAx architecture. In this architecture, the conventional concept of users working with just one system, respectively the sequential use of different systems with file based data exchange between them, was overcome. The one and only CAD/CAM system no longer exists in this new architecture [Dankwort et al. 1994]. The user works with a tool box and uses tools (CAx components) from various suppliers. The simple co-operation between these tools has to be realised by existing standards, respectively common interfaces (industrial standards). From the application point of view, there is a need to work interactively with "living" data, a data concept that goes far beyond data exchange with files. All CAx components from product development to manufacturing can be integrated in this structure.

The CAx system user has several demands on a CAx system architecture [Dankwort&Janocha1996]. The most important are:

- portability
- openness
- performance
- interoperability
- security

Two important standards can help to fulfil these demands: STEP and CORBA.

2 Fundamentals

2.1 STEP

In 1984, the International Standardisation Organisation (ISO) started the development of STEP with the goal of:

An unambiguous representation of computer interpretable product information throughout the life of a product [ISO-10303-1-93].

STEP is primarily a data model description of a product. The STEP standard contains a set of rules including not only mere product model data but also description methods, implementation methods and conformance testing methods.

The description language for STEP information models is EXPRESS (ISO 10303-11). EXPRESS allows to describe the data model with entities and entity structures including attributes and relationships between entities.

The demand on the scope of product data modelling highly depends on the industrial usage of specific applications. Because of this, product models for several application domains, the so called Application Protocols (AP) have to be defined.

---

[5] VDA: German Automotive Industry Association (**V**erband **d**er **A**utomobilindustrie e.V.)

To implement the STEP-philosophy in automotive industry, the Application Protocol "Core Data for Automotive Mechanical Design" (AP214) was defined. AP214 includes geometrical and non-geometrical data of parts as well as product data for mechanical parts and assemblies in the automotive development [HOLLA95].

2.2 Common Object Request Broker Architecture (CORBA)

Another international standard used for the realisation of the proposed CAx architecture is CORBA. It is an industrial consensus standard for distributed computing from the Object Management Group (OMG) [OMG 1997] to define interfaces for interoperable software systems using object-oriented technology.

The central component of the specification is the Object Request Broker (ORB). The ORB is the middleware that enables and establishes the client-server relationships and interactions between objects, possibly from different applications. Using an ORB, a client can transparently invoke a method on a server object which can be in the same software on the same workstation or even somewhere within a network. The client does not have to be aware of where the object is located, its programming language, its operating system, or any other implementation aspects that are not part of an object's interface. In doing so, the ORB provides interoperability between applications in heterogeneous distributed environments and seamlessly interconnects multiple object systems.

The interfaces of objects therefore are defined using the Interface Definition Language (IDL) which is a part of the CORBA standard and has become the universal notation for software interfaces. There are standard language bindings for C, C++, Java and several other programming languages.

ORBs also allow the integration of existing systems. In an ORB-based solution, developers simply "wrap" the legacy systems using IDL. A "wrapper" code translates between the standardised interfaces and the legacy interfaces, i.e. the Application Programming Interfaces (APIs) of the legacy systems.

CORBA has gained widespread acceptance and is supported by several commercial products. There are various implementations of the abstract CORBA standard from several vendors[6]. Version 2.0 of CORBA, adopted in December 1994, defines interoperability by specifying how ORBs from different vendors can interoperate.

While CORBA does not directly support object persistence, its open architecture includes an object adapter which may provide object persistence [Reverbel 1997]. An object adapter is an intermediary between an object implementation (the server) and the ORB core and can provide object persistence.

3 The CAx object bus

To verify the feasibility of the proposed CAx architecture (comp. chapter 1), the project ANICA has been started. Figure 1 shows the system architecture of ANICA.

The aim of the project is to analyse the interfaces of conventional CAx systems and CAx system kernels in order to develop a concept for a common harmonised access interface.

This common interface is the basis for the concept of distributed object-oriented CAx systems consisting of components from various system suppliers and a common interface to these components. In this architecture, the components provide the necessary CAx functionality and the interface interlinks the components, enabling the user to call functions across system and platform borders (Figure 1).

[6] e.g. Orbix from IONA, Visi-Broker from Visigenic or CORBAplus from ExperSoft.

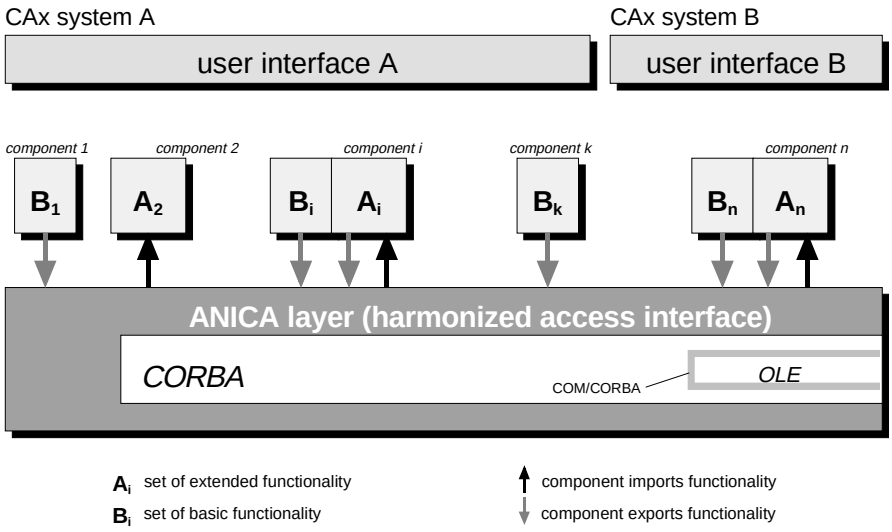

Figure 1: System architecture of ANICA

Based upon the comparison of different peculiarities of semantically equivalent or similar functions, in the system-specific APIs, a normalisation process leads to common harmonised interfaces. Hereby, CORBA is not only understood as a middleware but also as an architectural principle for CAx objects co-operating via a standardised common interface. Therefore the common interface environment can also be called a 'CAx object bus'.

There is an important difference between this approach and the conventional data exchange: By using the CAx object bus, there is no file transfer or file-based data exchange at all. Neither files in system specific data formats nor in neutral formats are transferred or even created. The whole co-operation is carried out online and without multiple persistent storage of data.

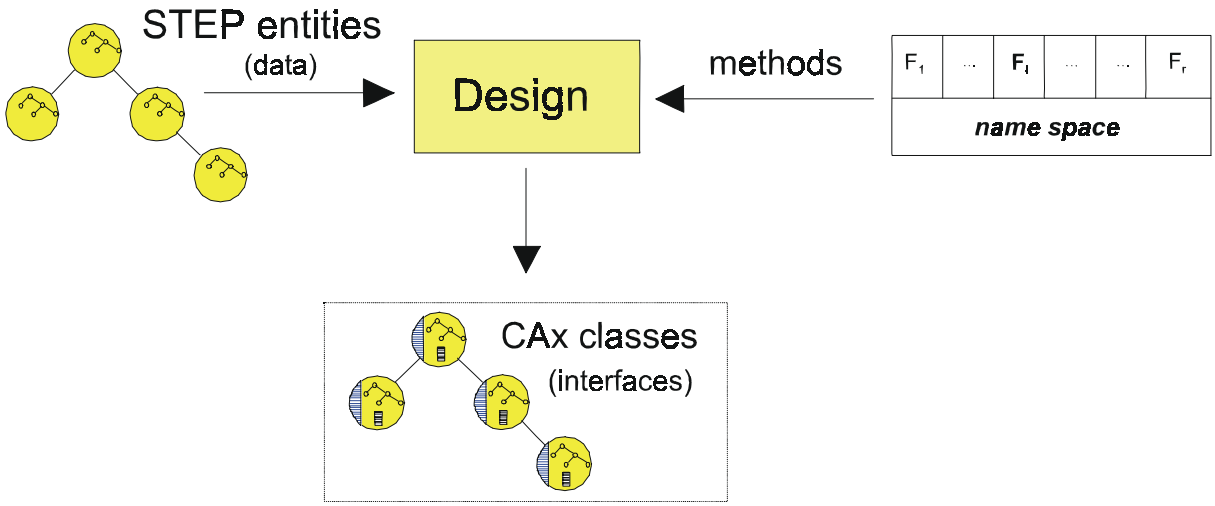

Figure 2: Design process of the CAx classes

The common interface consists of a collection of CAx classes which are defined in IDL. In general, the definition of classes and relationships between them is a decisive point in the design of an object-oriented system. Applied to the CAx environment of the automotive industry, the STEP entities of subset CC1 (conformance class 1) of the Application Protocol AP 214 have been chosen as a starting

point. But since STEP entities are mainly data descriptions, suitable methods for the manipulation of the data have to be added to form complete and reasonable CAx classes (figure 2).

In ANICA, the pragmatic approach of finding such suitable methods by analysing the APIs of several existing CAx systems has been carried out. Details may be found in [Janocha et al. 1998].

The ANICA approach offers the following advantages:

- The user is able to configure his own system in an optimal way according to his purpose by combining the best CAx components available on the market
- The user will be charged only for the components he actually needs
- Using optimal CAx components, the effort for training may be reduced
- The CAx data are common and accessible to all CAx components and for all partners in the process chain
- The user no longer depends on one single system supplier

But the ANICA approach doesn't provide any mechanism to store the objects of the CAx object bus. If a distributed CAx system as proposed in the ANICA project should ever become reality, the problem to store the CAx data still have to be solved.

4 Persistence Mechanisms in Distributed CAx Systems

A distributed system is a "collection of (probably heterogeneous) automata, whose distribution is transparent to the user so that the system appears to be on one local machine. This is in contrast to a network, where the user is aware that there are several machines, and their location, storage replication ... and functionality is not transparent." [FOLDOC98] There are several possibilities of building a distributed CAx system and the presented concept of the CAx object bus is only one possibility to realise such a environment. The storage mechanisms presented in the following chapters are valid for several concepts of distributed CAx systems.

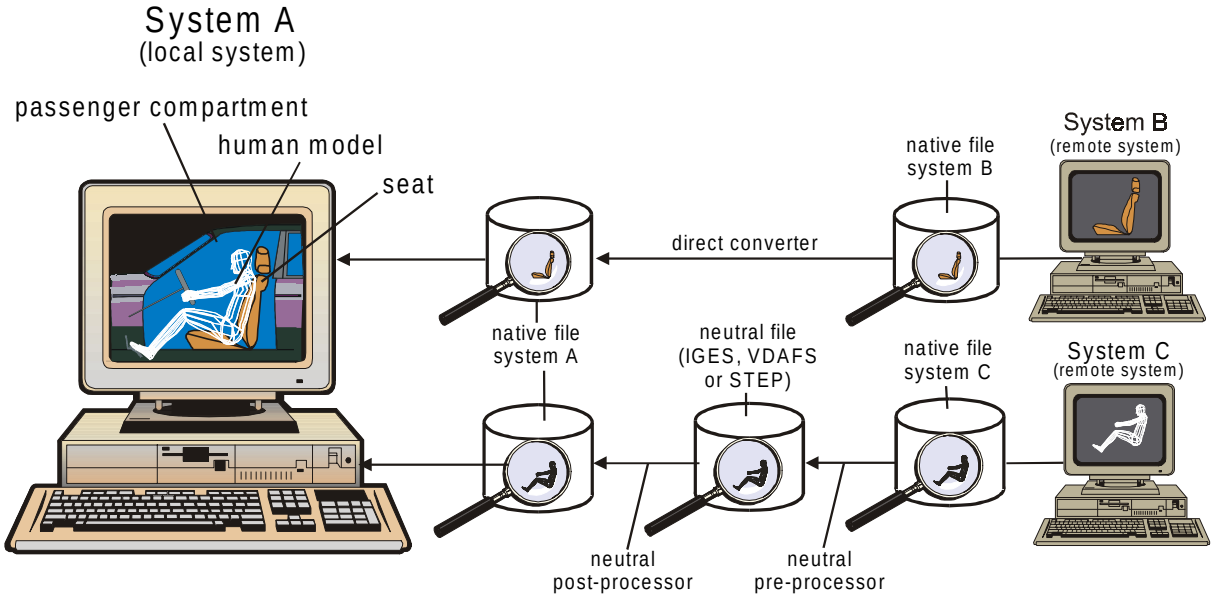

Figure 3: Passenger compartment including seat and human model merged using the conventional file based workflow

Figure 3 shows a typical application scenario of the synthesis of three different CAD models in one CAD system (system A). The partial models reside in different CAD systems: The passenger compartment is designed in system A, the seat is developed with system B and the human model is generated in system C.

In the conventional workflow, the files containing the native data of system B and C will be transferred via neutral data interfaces or by direct converters into native files of system A. If the data of system B or C have to be changed, the user has to repeat this transfer procedure.

In contrast, using a system based on the CAx object bus, there is no longer a need for file based data transfer between different integrated systems. Only the necessary representation information of the data models has to be transmitted as CAx objects through the CAx object bus.

4.1 CAx Data in Distributed Systems

By using a distributed CAx system based on the concept of the CAx object bus, the user generates a data model of the product where different parts of this model are distributed via the object bus and exist on more than one CAx system. This happens, when e.g. the local system is not able to create a special object (like a human model) but a remote system provides functionality to solve this specific problem. After the creation of a remote object, only the necessary data e.g. for visualisation, exists in the local system. The complete data, in this example the human model, is only represented in the remote system.

Regarding the example in figure 3, the seat and the human model may reside in the remote CAx systems B and C. To allow the user to work with all models in his local CAx system (A), the necessary data of the remote models have to be transmitted to this system (A) and are visualised on his working screen. The user should be able to modify all (local and remote) models using the user interface of his local system. After finishing this work, the user should be able to store both the local and the remote data. In the case of simple geometries, the remote data may be transmitted from the remote system to the local system, but in some cases the local system may not be able to process these data. At least in this case, a concept of handling data and persistence in a distributed CAx system environment is absolutely necessary.

The following chapters describe several mechanisms to guarantee consistent data storage of CAx models existing in an distributed CAx system environment without file based data exchange.

4.2 Storage of Distributed CAx objects Using Native Formats

To store a product data model with parts residing in different CAx systems, the native data descriptions of this parts can be used. There are two possibilities (Figure 4):

- Each system stores its own objects in its native database
- The whole distributed model is saved in the native database of the local system

The **first** approach to a distributed data management strongly contradicts the demands of the user. Because of process demands such as responsibility, release policy or warranty, a CAD system user must be able to save and to archive the actual status of the complete model at a central server after finishing respectively interrupting his work.

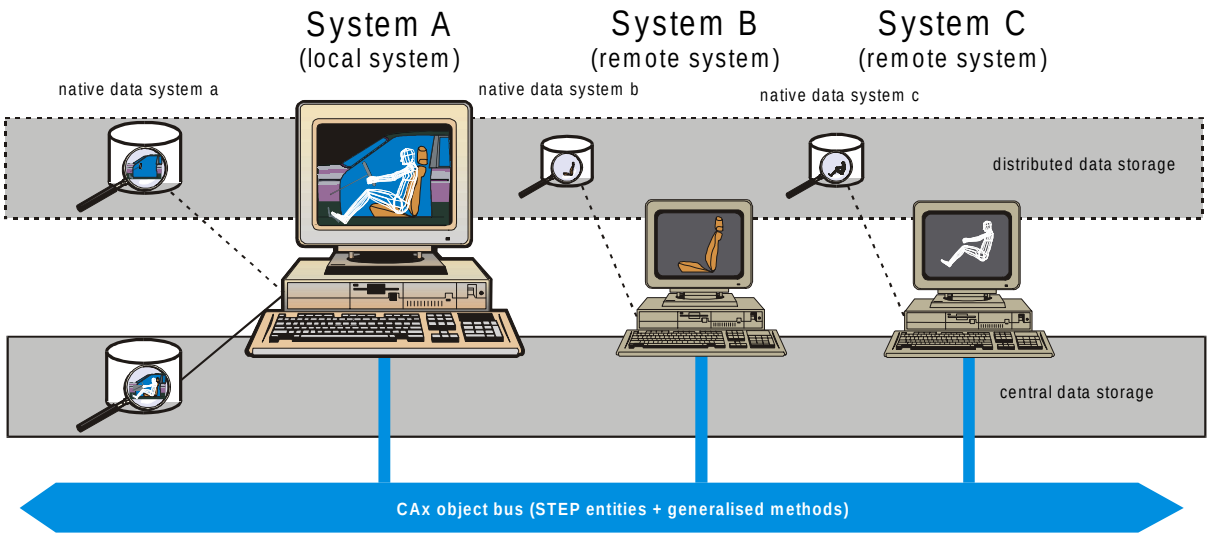

Figure 4: Storage of distributed CAx-objects in native format

The **second** approach presented in figure 4 fulfils the user demands for a central data management using the native format of one dedicated system as a data integration platform. After finishing his work, all linked geometries are stored in the native data format of the local system and the user is able to save his distributed data model in the same way as saving local models. This persistence mechanism offers the possibility to save the whole distributed data model at a central place. But after leaving the local CAx system, the connection to the remote models will get lost.

Based on the second approach, the project ProDMU[7] was started. The aim of this project is to realise Digital Mock-Up between CATIA and Pro/Engineer models using CATIA as local system. The results of the examination containing the colliding faces of both data models plus the intersecting curves are saved in one native file. Regarding the limited applicability of this prototype, the solution to store the results of the examination (in our case the distributed model) in one native file is well suited to support the requirements.

But in a general, the needs regarding the used CAx system to manage the distributed data are very high. The CAx system selected to store the distributed model must be able to handle all data structures appearing in the distributed system in its own data format. But instantaneously, no available commercial CAx system which meets this requirement is known to the authors.

In addition, the native format of a specific system is not suitable to archive CAD data for a longer period of time (e.g. 25 years as recommended for the automotive industry) since system versions and native file formats usually change in much shorter intervals.

4.3 Storage Of Distributed CAx Objects Using A STEP-Based Data Server

Due to these complex user needs, a concept of a STEP-based data server directly connected to the ANICA object bus has been developed. While using a central data server based upon a neutral data format, most of the system and user demands could be satisfied. The functional principle of this data server is shown in figure 5.

[7] ProDMU: Development of a prototype to connect the CAD systems CATIA and Pro/Engineer via the ANICA CAx object bus, carried out by RKK/University of Kaiserslautern and an automotive company.

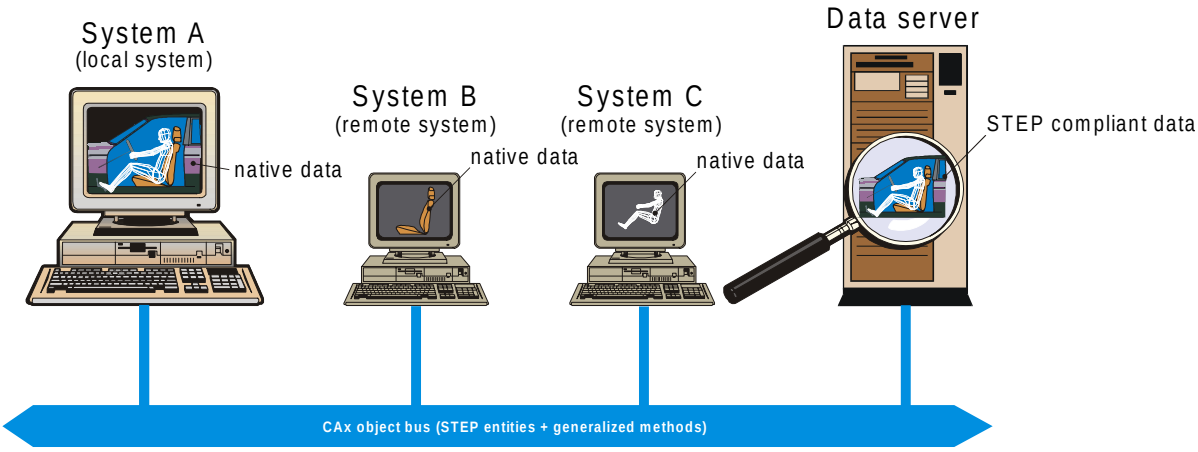

Figure 5: Central data storage using a data server

The user is working in a similar way as in the scenario shown in figure 3. All data generated in the distributed CAx system are administrated online by the data server during the design process. After finishing the design of a part, the STEP representations of all data contained in the distributed model are saved by the data server. The local data of the remote models residing in system A will get lost after leaving the working environment. If the user wants to continue the design work, the distributed model must be reloaded. Thereby, every partial model is loaded in its corresponding CAx system (e.g. the seat in system B) and the necessary data are transmitted to the local system (A).

In a conventional monolithic system, a geometric translator has to convert the native data structures, which are usually different from the STEP representation, into a STEP physical file. By using the ANICA object bus, the transient objects of the bus and the persistent CAx objects in the data server already have a STEP-compliant data structure. Thus, these data can be easily stored into a STEP physical file. This also ensures the ability to communicate with conventional CAx systems.

The architecture of the new data server is shown in figure 6.

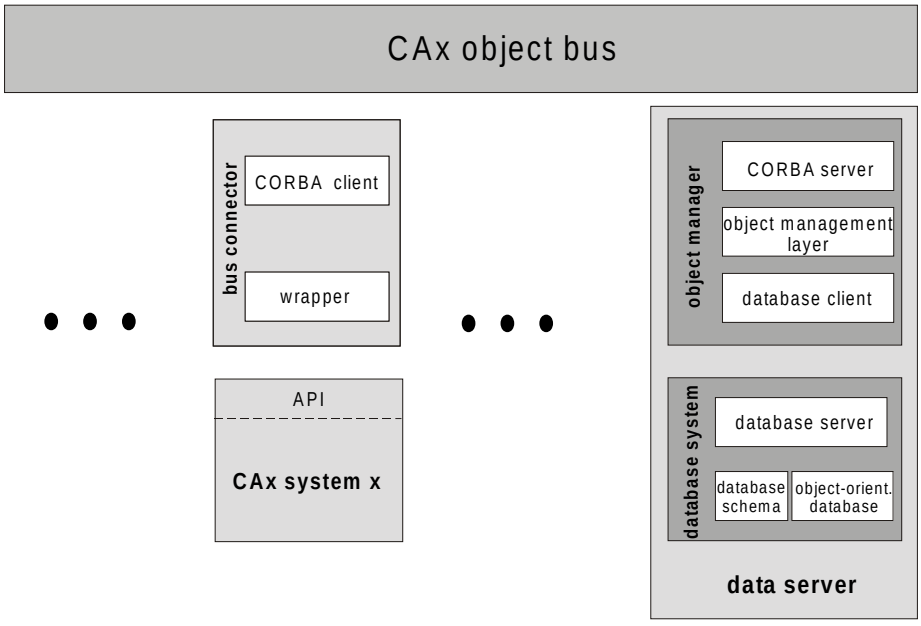

Figure 6: Architecture of the data server

The data server consists of the following two parts:

- object manager
- object oriented database system

The **object manager** is located between the database system and the CAx object bus and is both a database client and a CORBA server and contains the whole functionality of the data server. The main component of the object manager is the object management layer. This layer ensures the consistency of all CAx objects contained in the distributed CAx model. On the one hand, it keeps the links to the transient CORBA objects of the object bus and on the other hand the links to the persistent objects stored in the object-oriented database. The implementation of this software layer may e.g. be realised using the Orbix+ObjectStore-Adapter developed by IONA and Object Design Inc.

The central component of the server is the **database system**. The database system consists of a database server, an object-oriented database and a database schema. Because of the object-oriented structures of EXPRESS it is obvious to use STEP application protocols as database schemas for object-oriented databases. Most of the C++ based object-oriented database systems are well suited for STEP applications, because C++ and EXPRESS are to some extent similar. Schemas written in EXPRESS can easily be translated into corresponding C++ classes [Krebs & Lührsen 1995] [Lührsen 1996]. A prototype of the described architecture of a data server based on ObjectStore (database system) and the ANICA CAx object bus is currently under development.

5 Conclusions

Distributed CAx systems are a promising way to handle the great variety of CAx systems used in the development process. Industrial CAx experts doubt that there will ever exist one single CAx system able to meet all demands of the modern industrial network for product development and manufacturing. However, for the successful employment of distributed CAx systems, international standards, especially STEP, are inevitable.

Engineers using CAx systems as well as the responsible management will only accept a distributed CAx system if there is a mechanism which guarantees a central data management of the distributed data and supports the communication with conventional CAx systems through neutral interfaces. These user demands can be met using a central data model server linked to the communication interface connecting the different components.

The presented concept of a data server is only one step towards a CAx world without an explicit data exchange saving a great loss of time and money. The realisation of an open modular CAx system world is a joint challenge for both system suppliers and application companies.

6 References

[Dankwort et al. 1994] Dankwort W., Kellner P., Leu D., Peterson J., Renz W.: *Entwurf einer möglichen Systemarchitektur für Anwendungen in der Automobilindustrie*, in: VDI-Berichte Nr. 1134, 1994, pp. 431-442

[Dankwort & Janocha 1996] Dankwort, W., Janocha, A. T. : *Von Monolythen zu Komponenten: CAx-Architekturen im Wandel*, Proceeding of CAD '96, Vol. 1, Kaiserslautern, pp. 358-368.

[Dankwort 1997] Dankwort, W.: *CAx System Architecture of the Future*, in Roller D., Brunet P. (Eds): CAD Systems Development, Tools and Methods, (1997), pp. 20-31

[FOLDOC 1998] Free On-Line Dictionary of Computing, http://wombat.doc.ic.ac.uk/foldoc/index.html

[Janocha et al. 1998] Janocha A. T., Arnold F., Gandyra M., Iselborn, B., Kilb, T., Swienczek, B.: *"Projekt ANICA: Der Weg zu CAx-Komponentensystemen"*, in: CAD-CAM REPORT 3/98, March 1998

[Krebs & Lührsen 1995] Krebs, T., Lührsen, H.: *STEP Databases as Integration Platform for Concurrent Engineering*, Proc. 2nd International Conference on Concurrent Engineering (McLean, Virginia, August 23-25 1995), Johnstown, PA: Concurrent Technologies Coorporation, 1995, pp. 131-142.

[Lührsen 1996] Lührsen, H.*: Die Entwicklung von Datenbanken für das Produktmodell der ISO-Norm STEP*, Dissertation, Universität Erlangen, 1996

[Mowbray & Zahavi 1995] Mowbray, T. J., Zahavi, R.: *The Essential CORBA: Systems Integration Using Distributed Objects*, John Wiley & Sons Inc., 1995

[ODMG 1997] *The Object Database Standard: ODMG 2.0*, edited by R. G. G. Cattell and D. K. Barry, Morgan Kaufmann Publishers, May, 1997.

[OMG 1997a] Object Management Group (OMG): *The Common Object Request Broker: Architecture and Specification*, Revision 2.1, August 1997

[OMG 1997b] Object Management Group (OMG): *CORBAservices: Common Object Services Specification*, November 1997

[Reverbel 1997] Reverbel, F.: *Persistence in Distributed Object Systems: ORB/ODBMS Integration*, Ph. D. Dissertation, University of New Mexico, April 1996