MACAO - A Journey into CAx Interoperability and Collaborative Design

Florian Arnold
Research Group for Computer Application in Engineering Design (Prof. C. W. Dankwort)
University of Kaiserslautern, Germany
http://rkk.mv.uni-kl.de
arnold@mv.uni-kl.de

Abstract

The increasing parallelisation of development processes as well as the ongoing trends towards virtual product development and outsourcing of development activities strengthen the need for 3D co-operative design via communication networks. Regarding the field of CAx, none of the existing systems meets all the requirements of very complex process chain. This leads to a tremendous need for the integration of heterogeneous CAx systems. Therefore, MACAO, a platform-independent client for a distributed CAx component system, the so-called ANICA CAx object bus, is presented. The MACAO client is able to access objects and functions provided by different CAx servers distributed over a communication network. Thus, MACAO is a new solution for engineering design and visualisation in shared distributed virtual environments. This paper describes the underlying concepts, the actual prototype implementation, as well as possible application scenarios in the area of co-operative design and visualisation.

1. Introduction

Today, no single CAx system is able to meet all requirements of modern engineering design and manufacturing process chains, e.g. in automotive industry. Hereby, the abbreviation CAx stands for all computer-aided techniques applied during product development, e.g. CAD (Computer Aided Design), CAE (Computer Aided Engineering), CAM (Computer Aided Manufacturing), and CAS (Computer Aided Styling). An enormous number of different CAx tools is used for diverse purposes and the design of different parts of complex products.

Thus, to ensure the needed CAx interoperability, up to now a file-based data exchange by means of neutral file formats or direct converters has been essential and unavoidable. But this general practice has serious shortcomings like discontinuity of the design activity, limitation of parallelisation, loss of information, occurrence of conversion errors, and frequently the need for manual error correction.

As result of this, the idea of a distributed object-oriented CAx system consisting of components from various system suppliers and a common interface (a common access layer, respectively) to interlink the heterogeneous components across system and platform borders arose [1]. The targeted distributed component CAx system should be configurable according to process demands and it should allow for co-operative 3D design in a distributed shared virtual environment.

2. Fundamentals

To support a better understanding of the main concepts and techniques of the new approach to component CAx systems [2], some fundamentals are first explained: The international standards CORBA and STEP as well as the concept of the CAx object bus.

2.1. CORBA

The *Common Object Request Broker Architecture* (CORBA) [3] is a standard for the interoperability of distributed systems defined by the *Object Management Group* (OMG).

The central component of the specification is the *Object Request Broker* (ORB) which allows for interoperability and interaction between objects and applications in heterogeneous distributed environments. Therefore, an ORB can be seen as a (programming) language- and platform-independent object bus.

In order to use an ORB, the interfaces of the objects to be distributed via the object bus have to be defined using the CORBA *Interface Definition Language* (IDL).

2.2. STEP

STEP (*STandard for the Exchange of Product model data*) is an international standard (ISO-10303, *Industrial*

automation systems and integration - Product data representation and exchange [4]) for the computer-interpretable representation and the exchange of product model data. STEP is becoming the *lingua franca* in the area of modelling and exchanging product data.

One part of STEP is the formal specification language EXPRESS describing an information domain in terms of entities which are pure data descriptions, i.e. they do not contain any functionality for accessing or manipulating the data. Within STEP, so-called *Application Protocols* (APs) have been specified for several application domains.

To meet the specific demands of automotive industry, the AP 214 *Core Data for Automotive Mechanical Design Processes* [5] has been defined. STEP AP 214 describes a data model for the process chain of automotive development from product definition, styling, engineering design, prototyping, production planning etc. to quality control.

The STEP *Standard Data Access Interface* (SDAI) [6] was designed to be an API for access to data repositories containing data organised around EXPRESS schemas and does neither offer appropriate high-level access functions nor functions to manipulate CAx models. Therefore, the functionality offered by SDAI is not sufficient for online CAx interoperability including access to data structures and high-level functionality.

2.3. The ANICA CAx Object Bus

To fill the concept of CAx interoperability with life, the project ANICA (*ANalysis of access Interfaces of various CAx systems*) [7], [8] was carried out by the *Research Group for Computer Application in Engineering Design* at the University of Kaiserslautern, Germany, from 1995 to 1998. ANICA has been supported by the German automotive industry (*Audi*, *BMW*, *DaimlerChrysler*, *Porsche*, *Volkswagen*), the *Rhineland-Palatinate Foundation for Innovation*, as well as several leading CAD system suppliers.

The objective of this project was to create the basic architecture of a distributed platform- and system-independent CAx system. Thereby the individual server components offer their functionality as services across a common interface which interlinks the components. This unified common interface defined in CORBA IDL is called CAx object bus (see Figure 1).

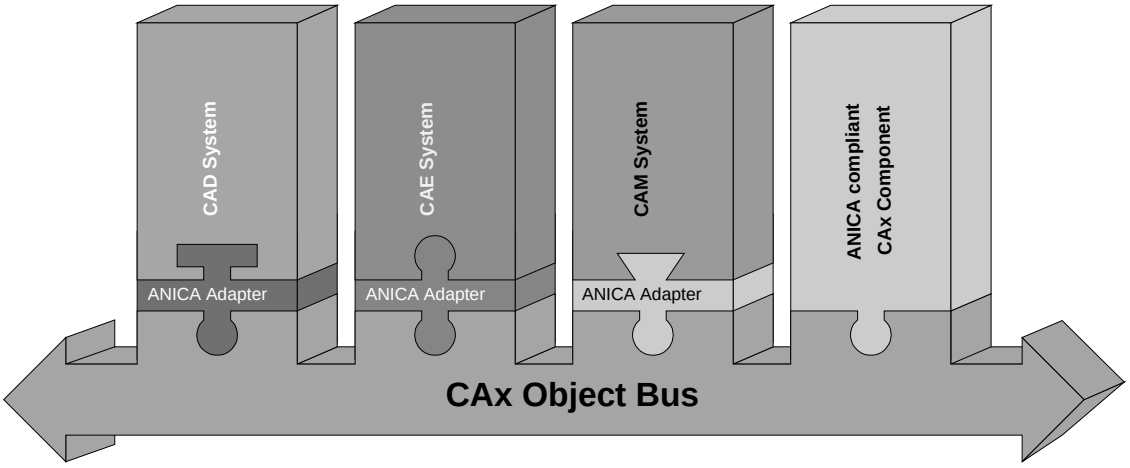

Figure 1. Different ways to connect CAx components to the ANICA CAx object bus which is based on CORBA and STEP standards

The fundamental approach for the definition of the CAx object bus was to analyse various conventional CAx systems in order to develop a uniform common access interface for CAx systems. Thus, during the ANICA project, the CAD systems *CATIA* (*Dassault Systèmes*), *DesignPostDrafting* (*Computervision*), *SolidEdge* (*Intergraph*), the CAD kernels and development environments *ACIS* (*Spatial Technology*) and *CAS.CADE* (*Matra Datavision*), as well as the anthropometric human model *RAMSIS* (*Tecmath*) have been analysed.

The definition of the common interface was on the one hand based upon the CORBA standard to make interoperability between different distributed components possible. On the other hand it was influenced by the (hierarchy of) entities defined in STEP AP 214 subset CC1 (Conformance Class 1). Since these entities are pure data descriptions, suitable operations had to be added to form complete and reasonable CAx classes respectively interfaces. The specification of such “canonical“, ”normalised” or “standardised“ operations defined on

STEP entities was derived from the analysis and comparison of different peculiarities of semantically equivalent or similar functionality in the APIs (Application Programming Interface) of the analysed systems.

This new approach of an online connection between different CAx systems allows for transparent access to data and functionality of the integrated CAx systems without any file-based data exchange. The connection between existing conventional CAx software systems and the object bus is carried out by system-specific software adapters which map the API of the respective legacy CAx system to the common interface of the CAx object bus and vice versa.

The first industrial application of the ANICA concepts was the project ProDMU [9], [7] which was carried out by the *Research Group for Computer Application in Engineering Design* in co-operation with *Volkswagen* from 1997 to 1998.

This project had the objective to improve the process of virtual collision and assembly checks in the scope of co-operative *digital mock-up* (DMU). This aim was achieved through a closer and more flexible integration of the participating CAD systems as with the traditional data exchange via neutral file formats. The ANICA CAx object bus formed the main part of the novel solution in form of an online coupling of the participating legacy CAD systems *Pro/Engineer* (*PTC*) as the client and *CATIA* as the server system. The new DMU process significantly improved the efficiency of working procedures and the quality of communication.

Despite its success, the ProDMU prototype did not address all problems. For example

- There is only a *one* to *one* relationship between clients and servers, not a more general *m* to *n* relationship.
- There are only "fat" servers, i.e. wrapped legacy systems. There are no newly created "thin" CAx components which can be connected to the CAx object bus without an adapter.

3. MACAO

3.1. General Concept

MACAO (*Multi-context Adaptive Client for the ANICA CAx Object bus*) is the first ANICA compliant component for the CAx object bus, i.e. it has been developed from scratch and can be connected to the CAx object bus without the need for an adapter (compare Figure 1).

The MACAO client has been designed as a thin client, i.e. it has no own local CAx functionality, but it has access to all the remote CAx server functionality connected to the CAx object bus. The client is only based upon the standard interface (i.e. the CAx object bus) and thus independent from any specific CAx server system. Hence, other CAx systems can also easily be added to the CAx object bus without the need for redesigning the MACAO client.

In contrast to the approach used in the projects ANICA and ProDMU, the MACAO concept is not based on the transmission of mathematically exact geometrical and topological data from client to server (e.g. degrees, control points, and weights of NURBS surfaces). Instead, the common interface of the CAx object bus has been supplemented to allow for the transfer of tessellation data enhanced by references to the corresponding faces and edges in the server model.

Today, there is no real triangulated shape representation within the STEP standard. Only the *facetted_brep* entities come close to this but they are not well-suited for this task. Therefore, new data structures and appropriate access functions have been defined which have been derived from the data structures found in the APIs of the analysed CAD systems.

The transferred tessellated (or facetted) data consists of triangles representing faces and of points which define polylines representing curves (Table 1). This information is sufficient not only for visualisation but also for interaction because it enables the client to access the associated server-side mathematical description of surfaces and curves if needed.

Table 1. Comparison of the data managed by client and servers

MACAO Client	MACAO Servers
Tessellation (facetted) data:	Mathematically exact geometrical and topological representation of:
• triangles • plus references to the server faces	• faces and underlying surfaces (NURBS surfaces, planes, cylindrical surfaces etc.)
• polylines (sequences of points) • plus references to the server edges	• edges and underlying curves (NURBS curves, lines, arcs etc.)

Despite its lightweight structure, MACAO is much more than only an online viewer: It enables the client to use all the CAx server functions coupled with the CAx object bus. Instead of pure static viewing it allows for dynamic interaction and the manipulation of server models. Thus, its allows for true co-operative 3D design

and visualisation across system and platform borders. To show the feasibility of this concept, a few fundamental CAx functions have been added to CAx object bus and are therefore accessible for client applications (compare chapter 3.2).

Besides the capabilities described above, the engineer working with the client is able to define his own working context by specifying a 3D selection box according to his regions of interest of the remote model. Then, only those faces and edges of the remote server model which lie inside the defined 3D box are transmitted to the client, i.e. the necessity to transfer complete models no longer exists.

The MACAO client also has a context-sensitive GUI, i.e. a GUI which is self-adapting at run-time according to the objects to operate on and the functionality available to manipulate them. The main difference to a conventional context-sensitive GUI is that the set of available objects and functions depends on the kind and number of CAx servers connected to the CAx object bus at runtime. Each server may provide different objects and/or different functionality applicable to these objects. Thus, the available kinds of objects and functions may dynamically change during a session. Therefore, the client needs some meta-information about the capabilities of each CAx server actually coupled with the CAx object bus, and the client must adapt its GUI according to this meta-information at runtime. In this context, general considerations about taxonomy and granularity of CAx components were necessary which are omitted here due to space limitations.

3.2. The MACAO Prototype

The actual prototype implementation consists of the MACAO client and two CAx servers. The client was built using *Java* and the *Java 3D API*, the ORB *Visibroker for Java* (*Inprise*) on a *Pentium II* PC with *Windows NT*. The *Java 3D API* [10] is a scene-graph-based 3D application programming interface for the development of complex 3D applications with Java. Today, Java 3D implementations for *Microsoft Windows*, *Linux*, *Sun Solaris*, *HP-UX* and *SGI IRIX* exist. Additional implementations for other platforms are under way. The usage of Java technology makes the client highly portable.

Each CAx server consists of a legacy CAD system (including its API), an ORB implementation, and some adapter code which also performs some data and reference management tasks. Both servers have been built using *C++* and *ORBacus* (*Object Oriented Concepts, OOC*) on an *SGI O2* with *IRIX* operating system. One server makes use of *Pro/Toolkit*, the API of the CAD system *Pro/Engineer* (*PTC*), while the other server is built upon *UG/Open* and *UG/Open++*, the APIs of the CAD system *UG* (*Unigraphics Solutions*).

To show the general feasibility, a few fundamental CAx functions have been added to the CAx object bus to be accessible for client applications (see table 2). Whenever the client calls a server function (e.g. blend edge) which changes a server-side model, an automatic online update of the changed, created and deleted faces and edges takes place. This means that the client- and server-side models are synchronised automatically. Therefore, true simultaneous engineering becomes possible. After making such changes to remote server-side models from client-side it is possible to save the changed models at server-side from within the client.

The *UG*-based server can only be used in interactive mode whereas the *Pro/Engineer*-based server can be used in both interactive and batch mode.

Table 2. CAx server functionality coupled with the CAx object bus and therefore accessible from client-side

CAx server based on *Pro/Engineer*	**CAx server based on *UG***
Pro/Engineer functionality coupled with the CAx object bus: • highlight face • highlight edge • update the client-side model (after changing the server-side model locally) • transfer partial model (transfer only that part of a model which lies inside a particular 3D selection box defined on client-side)	*UG* functionality coupled with the CAx object bus: • highlight face • highlight edge • blend edge • chamfer edge

Figure 2 shows two quite complex models loaded in the MACAO client prototype (upper middle screenshot) as well as each of the two parts in its original CAx server system. Both models consist of more than 300 faces. One model comes from the server based on *UG* (lower left screenshot) and one model is provided by the server based on *Pro/Engineer* (lower right screenshot).

Altogether, three faces have been highlighted at client-side: two faces of the upper model and one face of the bottom model. This highlight information has also been transmitted back to the servers. The highlights can therefore also be seen in the server-side models, i.e. the engineer working with *UG* sees one highlighted face in his

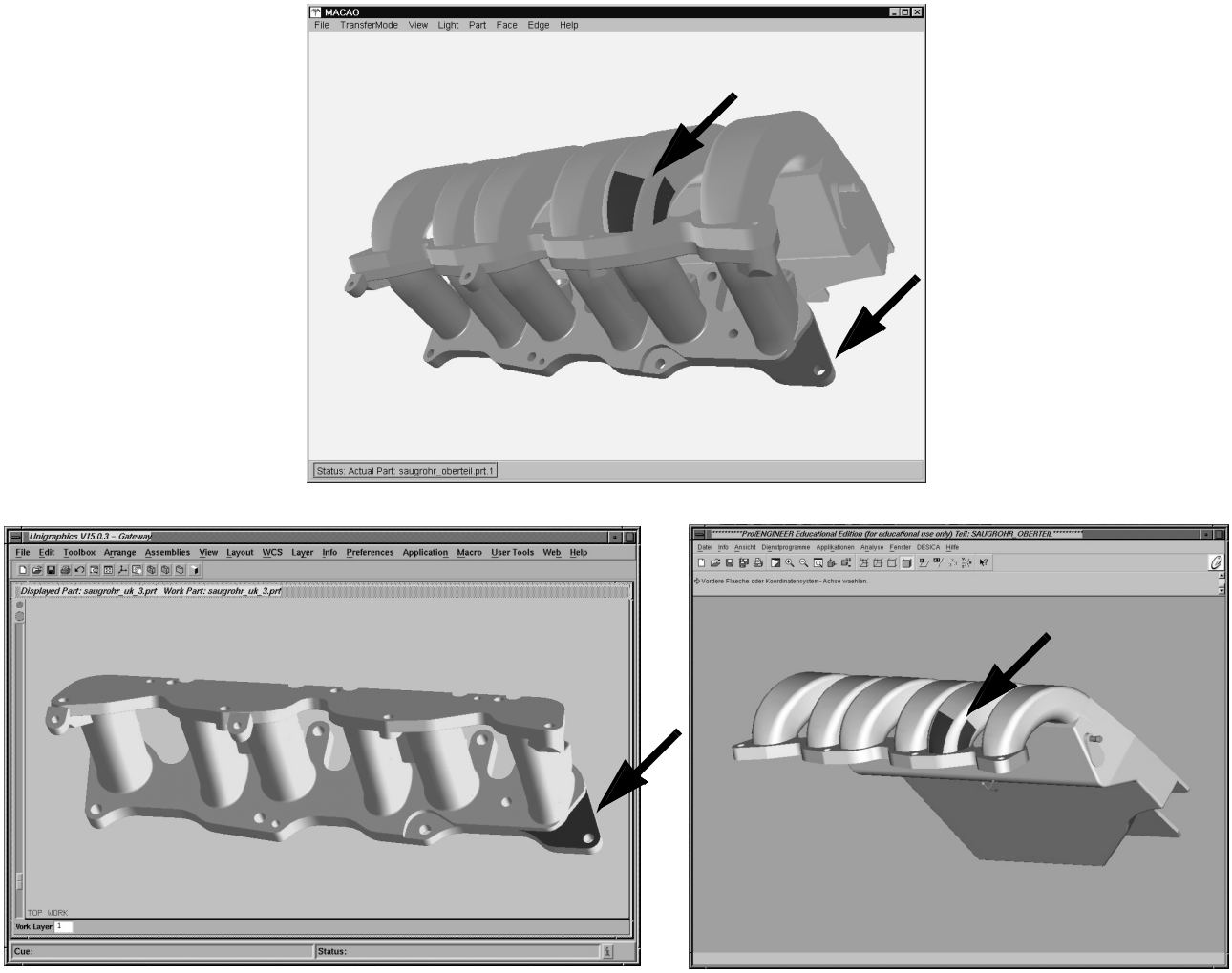

Figure 2. Screenshots of the MACAO client (upper middle screenshot), the UG-based server (lower left screenshot) and the Pro/Engineer-based server (lower right screenshot). Altogether, three faces have been highlighted at client-side. This highlights can also be seen in the server-side models.

local *UG* model and the engineer working with *Pro/Engineer* sees two highlighted faces in his local model.

The performance of the MACAO prototype is already noticeably better than the ProDMU solution as well as the file-based data exchange using STEP physical files.

3.3. Application Scenarios

One possible area of application scenarios for MACAO is the collaboration between different departments of the same company (or between a manufacturer and his suppliers) e.g. working on different parts of the same product while using two different CAD systems. For example, one CAD systems may be used for developing all parts and assemblies situated in the engine compartment and another CAD system for car body development. In this example, the MACAO client may be used by DMU working groups which have to carry out DMU examinations with CAD models originating from both CAD systems.

For such digital mock-up purposes, the client could be enhanced to carry out local collision detection algorithms based either on tessellation data or mathematically exact data which would additionally have to be transferred from the servers. The communication could happen within LANs or virtual private networks. The usage of MACAO in combination with video- or teleconferencing tools

would be particularly reasonable. By these means, true interactive and co-operative DMU sessions with participants from a DMU working group as well as from the two development departments may take place.

Comparing the application of the MACAO technology with e.g. the traditional data exchange using STEP physical files, some important differences can be recognised: The communication between the involved engineers can be noticeably improved by the complete transmission of information about problematic areas, e.g. transfer of highlighted faces and edges back into the server system which are therefore visible within the original model. Furthermore, the previously existing interruptions of the design process, which have been caused by the activities necessary for the file-based data export and import, can be avoided.

4. Conclusions

The MACAO client in combination with the ANICA CAx object bus allows for a new form of distributed collaborative 3D engineering design in shared virtual environments. The MACAO prototype shows the feasibility of the concepts as well as its suitability for practical applications in the fields of co-operative design, DMU, and virtual product development via communication networks.

Regarding performance, the MACAO prototype is already superior to the traditional file-based data-exchange and substantial gains in productivity can be reached.

The prototype implementation, especially the chosen CAx systems, represent only one special case of application. The integration of additional CAx systems can be done without the need to change the common interface of the CAx object bus or the MACAO client. Obvious candidates for integration are other CAD systems (*CATIA*, *I-DEAS*, etc.) and *computer-aided engineering* (CAE) tools for *finite element analysis* (FEA). In the latter case, changes to the product model based on FEA calculations could be carried out from within the CAE tool.

Another possible enhancement of the MACAO client for even more flexible DMU examinations is the combination of CAD models transferred via the CAx object bus with e.g. VRML models locally loaded (from files) by the client. The integration of Java 3D object loaders for VRML, DXF, 3DS (*3D-Studio*), OBJ (*Wavefront*) etc. with the MACAO client can easily be done, because file loaders for these formats are already available.

References

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

[2] Further information about ANICA, ProDMU, the CAx object bus and CAx components: http://rkk.mv.uni-kl.de/ComponentCAx/ComponentCAx_engl.html

[3] Object Management Group (OMG), *"The Common Object Request Broker: Architecture and Specification"*, Revision 2.3.1, October 1999

[4] International Organization for Standardization, ISO 10303-1: Industrial Automation Systems and Integration - Product Data Representation and Exchange (STEP), Part 1 (International Standard), *"Overview and fundamental principles"*, Geneve, 1994

[5] International Organization for Standardization, ISO/IS, DIS 10303-214, Industrial Automation Systems and Integration - Product Data Representation and Exchange, Part 214 (Draft International Standard): *"Application Protocol: Core Data for Automotive Mechanical Design Processes"*

[6] International Organization for Standardization, ISO 10303-22, Industrial Automation Systems and Integration - Product Data Representation and Exchange (STEP), Part 22 (Draft International Standard): *"Standard data access interface specification"* (SDAI)

[7] A. Janocha, "CAx-Systemintegration auf Basis von CORBA und STEP", *Produktdaten Journal*, Nr. 1, June 1996, pp. 45-48

[8] F. Arnold, A. Janocha, B. Swienczek, T. Kilb, "Die CAx-Integrationsarchitektur ANICA und ihre erste Umsetzung in die Praxis", in: *Tagungsband zum Workshop Integration heterogener Softwaresysteme (IHS '98)*, 28. GI-Jahrestagung Informatik '98 - Informatik zwischen Bild und Sprache, Magdeburg, September 1998, pp. 43-54

[9] B. Swienczek, F. Arnold, T. Kilb, A. Janocha, R. Sartiono, "Online-Kopplung von CAx-Systemen für die virtuelle Produktentwicklung: Ein Vergleich mit dem dateibasierten Datenaustausch", in: VDI Berichte 1435: *"Prozeßketten für die virtuelle Produktentwicklung in verteilter Umgebung"*, 1998, pp. 219-238

[10] H. Sowizral, K. Rushforth, M. Deering, *"The Java 3D API Specification"*, Java Series, Addison-Wesley, 1998