Die CAx-Integrationsarchitektur ANICA und ihre erste Umsetzung in die Praxis

Dipl.-Inform. Florian Arnold, Dipl.-Ing. Andrzej T. Janocha, Dipl.-Ing. Bernd Swienczek, Dipl.-Ing. Thomas Kilb

Lehrstuhl für Rechneranwendung in der Konstruktion
Fachbereich Maschinenbau und Verfahrenstechnik
Universität Kaiserslautern

Zusammenfassung

Das Problem der Integration heterogener Softwaresysteme stellt sich auch auf dem Gebiet der CAx-Systeme, wie sie in vielfältigen Ausprägungen etwa in der Automobilbranche für die Fahrzeugentwicklung eingesetzt werden. Zunächst werden die heute in diesem Bereich praktizierten Lösungen und die dabei auftretenden Probleme kurz dargestellt. Danach werden der neue Standard für Produktdaten, STEP, und der Standard für die Interoperabilität heterogener Softwaresysteme, CORBA, sowie einige CORBA-Entwurfsmuster erläutert. Als nächstes wird eine auf diesen beiden Standards basierende CAx-Integrationsarchitektur, die im Projekt ANICA entwickelt wurde, vorgestellt und die prinzipielle Vorgehensweise bei ihrer Realisierung beschrieben. Daran anschließend wird über eine erste Umsetzung dieser Architektur in die Praxis berichtet. Zum Abschluß wird kurz auf die gewonnenen Erfahrungen eingegangen und ein Ausblick auf zukünftige Entwicklungen gegeben.

1. Einleitung

Der Hauptgrund, warum Software veraltet und unmodern wird, besteht nicht darin, daß sie „verschleißt“. Vielmehr scheitert häufig ihre Anpassung an die sich verändernde Umwelt. Eine Integration mit neuen Anwendungen und sich entwickelnden Technologien gelingt oft deshalb nicht, weil mehr Entwicklungsaufwand in die interne Struktur des Programms investiert wurde als in die Schnittstellen des Programms zum Rest der Welt [Curtis 1997].

Diese grundsätzliche Problematik stellt sich auch im Bereich der CAx-Systeme, wie sie z.B. in der Automobilindustrie verwendet werden. CAx ist ein Sammelbegriff für alle im Laufe des Poduktlebenszyklus eingesetzten CA-Techniken. Hierzu zählen CAD (Computer Aided Design), CAM (Computer Aided Manufacturing), CAE (Computer Aided Engineering), etc. Im Laufe der Prozeßkette der Automobilentwicklung werden verschiedenste CAx-Systeme für Styling, 3D-CAD-Konstruktion, Berechnung, Simulation, etc. eingesetzt. Diese sehr unterschiedlichen und komplexen Softwaresysteme müssen möglichst nahtlos verbunden werden, um den Praxisanforderungen, wie z.B. Durchgängigkeit der Daten, Vermeidung redundanter Datenhaltung, Verkürzung der Entwicklungszeiten und Parallelisierung der Entwicklungsabläufe, gerecht zu werden. Im Bereich der Automobilindustrie und ihren Zulieferern wird heute auf zwei verschiedenen Wegen versucht, dies zu erreichen.

Der erste Weg besteht im Einsatz systemspezifischer Direktkonverter, die je zwei Systeme miteinander koppeln. Dies führt jedoch zu einer Vielzahl an notwendigen bilateralen Verbindungen, die sehr hohe Aufwände an Entwicklung und Wartung verursachen und schlecht austauschbar bzw. erweiterbar sind. Die Direktkonverter sind dabei immer systemspezifisch und oft sogar auf bestimmte Programmversionen zugeschnitten.

Den zweiten heute praktizierten Weg bildet der dateibasierte Datenaustausch mit Hilfe neutraler Datenformate wie IGES (Initial Graphics Exchange Specification), VDA-FS (Verband Deutsche Automobilindustrie Flächen-Schnittstelle) und seit kurzem STEP (**ST**andard for the **E**xchange of **P**roduct Model Data). Diese Form des Datenaustauschs wurde in den letzten Jahren ständig verbessert, weist aber nach wie vor Probleme auf. So ist im

Bereich der Geometriedaten die Übertragung von Volumenmodellen mit VDA-FS nicht möglich, da VDA-FS keine Volumenmodelle unterstützt. IGES bietet zwar prinzipiell diese Möglichkeit, allerdings stellen die derzeit existierenden Konvertierungsprogramme diese Funktionalität nur in geringem Umfang oder gar nicht zur Verfügung. Weiterhin unterliegen die Daten bei Verwendung neutraler Datenformate oft mehreren Konvertierungen, die zum Verlust von Strukturinformationen und zu numerischen Ungenauigkeiten führen können. Daher ist häufig eine (meist manuell durchgeführte) Nachbearbeitung notwendig. Beide praktizierten Wege können somit keine wirklich zufriedenstellende Lösung bieten.

Im folgenden wird nun der Standard STEP näher beschrieben, da er auf der Datenseite eine Grundlage für den im Projekt ANICA (**AN**alysis of Access **I**nterfaces of Various **CA**x Systems) entwickelten Ansatz zur Integration heterogener CAx-Systeme bildet. Danach wird auf den Interoperabilitätsstandard CORBA eingegangen, der als zweite Basis für die CAx-Integrationsarchitektur ANICA dient, die anschließend detailliert beschrieben wird.

2. STEP

STEP ist ein internationaler Standard (ISO 10303) für ein Produktdatenmodell und trägt den offiziellen Namen *Industrial Automation Systems and Integration – Product Data Representation and Exchange*, ist jedoch besser bekannt unter der Bezeichnung ***ST**andard for the **E**xchange of **P**roduct Model Data*, von der auch die Abkürzung abgeleitet ist. STEP soll die rechnerinterpretierbare Repräsentation und den Austausch von Produktmodelldaten des gesamten Produktlebenszyklus ermöglichen.

Zur konsistenten, widerspruchsfreien und semantisch eindeutigen Beschreibung des Produktmodells von STEP wurde die formale Beschreibungssprache EXPRESS definiert. EXPRESS ist keine Programmiersprache, sondern eine Spezifikationssprache, die objektorientierte und prozedurale Konzepte sowie Konzepte aus dem Bereich der Datenbanken vereinigt. EXPRESS spezifiziert eine Informationsdomäne in Form von Entities, d.h. Klassen von Objekten, die gemeinsame Eigenschaften besitzen. Diese Eigenschaften werden durch assoziierte Attribute und Constraints repräsentiert.

Für den industriellen Einsatz ist es nicht sinnvoll, den gesamten Umfang von STEP zu verwenden, da für einen speziellen Anwendungsbereich nicht alle Elemente benötigt werden und eine vollständige Umsetzung einen zu hohen Entwicklungsaufwand erfordern würde. Daher werden von der ISO domänenspezifische Applikationsprotokolle (AP) genormt, die jeweils den für ein bestimmtes Anwendungsgebiet benötigten Ausschnitt beschreiben.

So wurde vom Verband Deutscher Automobilhersteller (VDA) zusammen mit internationalen Partnern das Applikationsprotokoll *Core Data for Automotive Mechanical Design Processes* (AP 214) entwickelt. Das AP 214 legt ein Datenmodell für die mechanische Sicht auf die Prozeßkette in der Fahrzeugentwicklung von der Produktdefinition über Styling, Konstruktion, Prototyperstellung, Produktionsplanung usw. bis hin zur Qualitätskontrolle fest. Dazu gehören geometrische und nicht-geometrische Daten des mechanischen Anteils von Bauteilen, Baugruppen, Produkten und Betriebsmitteln. Die Conformance Class 1 (CC1) bildet eine Untermenge mit den für die Praxis wichtigsten Entities des AP 214 (Abbildung 1), die sich in die Entity-Gruppen Produktdatenmanagement, Elementstruktur, Drahtmodell, Flächenmodell und Volumenmodell gliedern.

Auch wenn die Standardisierung von STEP noch nicht abgeschlossen ist, nimmt dessen Bedeutung für die Praxis ständig zu. Daher besitzt STEP bzw. EXPRESS trotz einiger Schwächen (z.B. Inkompatibilitäten zwischen verschiedenen Applikationsprotokollen) das Potential sich mittelfristig als lingua franca im Gebiet der Modellierung und des Austauschs von Produktdaten durchzusetzen.

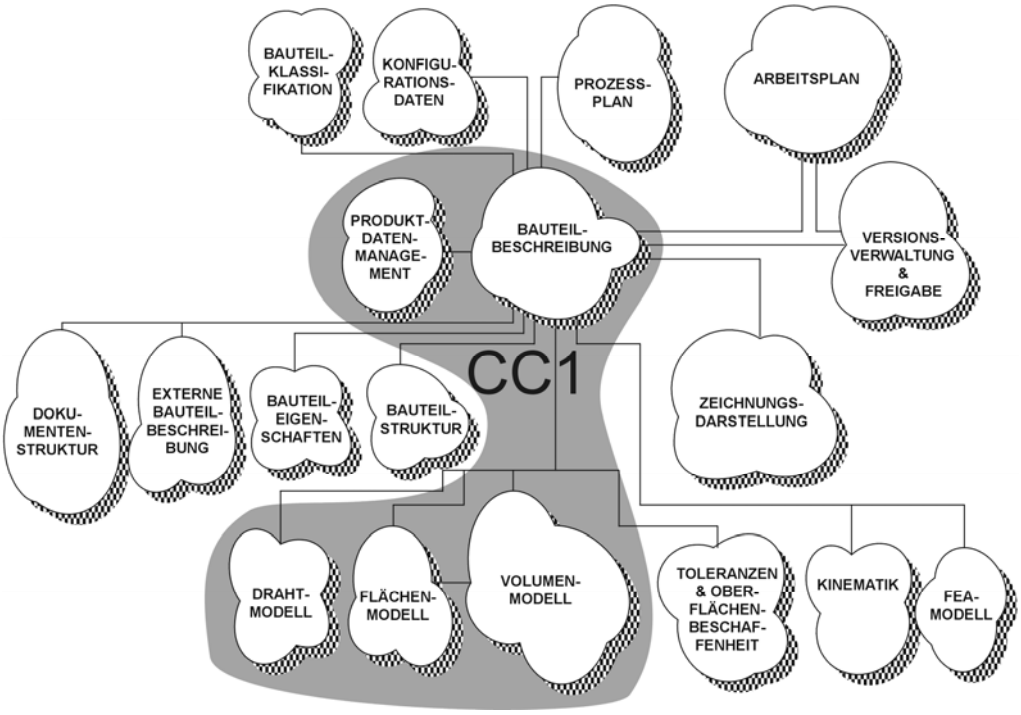


Abbildung 1: Die Conformance Class 1 (CC1) des STEP AP 214

3. CORBA

Die *Common Object Request Broker Architecture* (CORBA) [OMG 1998], [Orfali et al. 1998] ist ein Industriestandard für die Interoperabilität verteilter Systeme. CORBA ist Teil der *Object Management Architecture* (OMA) der *Object Management Group* (OMG), dem größten Softwarekonsortium der Welt. CORBA definiert Mechanismen, welche die transparente Interaktion von Objekten in heterogenen verteilten Umgebungen ermöglichen. Die zentrale Komponente der Spezifikation ist der *Object Request Broker* (ORB). Der ORB ermöglicht und unterstützt Interoperabilität zwischen Applikationen auf verschiedenen Rechnern in heterogenen Umgebungen. Ein ORB kann somit als ein programmiersprachen- und plattformunabhängiger Objektbus angesehen werden. Für den Einsatz eines ORBs müssen die Schnittstellen der über den Bus zu verteilenden Objekte in der im Rahmen von CORBA festgelegten *Interface Definition Language* (IDL) definiert werden. IDL hat sich auch über CORBA hinaus zu einer universellen Notation für die implementierungsneutrale Spezifikation von Softwareschnittstellen entwickelt. Für IDL existieren inzwischen standardisierte Mappings für C, C++, Smalltalk, Java, Cobol und Ada. Das als Teil von CORBA 2.0 definierte *Internet Inter-ORB Protocol* (IIOP) ermöglicht es auch den ORBs verschiedener Hersteller zu interoperieren („InterORBabilität“).

Eine der ersten domänenspezifischen Task Forces der OMG war die *Manufacturing Domain Task Force* (MfgDTF), deren Auftrag es ist, das Entstehen von rentablen, zeitgemäßen, kommerziell verfügbaren und interoperablen Softwarekomponenten für den Fertigungsbereich mit Hilfe der CORBA-Technologie zu fördern [OMG 1996a]. Ein solcher Ansatz zur Etablierung von Standardschnittstellen ist derzeit für den Bereich des Produktdatenmanagements (PDM) in Arbeit. Die dabei angestrebten Schnittstellen sollen sowohl allgemeine Aufgaben zur Unterstützung von verteilten Umgebungen für das Produktdatenmanagement bieten als auch Standardschnittstellen zwischen voneinander abweichenden PDM Systemen [OMG 1996b]. Dagegen wurden im CAx-Bereich bisher von der MfgDTF noch keine konkreten Vorschläge bezüglich standardisierter Schnittstellen erarbeitet.

4. CORBA-Entwurfsmuster

Ebenso wie für viele andere Bereiche existiert auch für die Entwicklung CORBA-basierter verteilter Softwaresysteme eine ganze Reihe von Entwurfsmustern. Ein Entwurfsmuster ist dabei nach [Mowbray & Malveau 1997] eine effektive Lösung für ein wichtiges Entwurfsproblem, die in dem Sinne wiederverwendbar ist, daß sie auf einen bestimmten Bereich von Entwurfsproblemen unter einer breiten Vielfalt von Begleitumständen anwendbar ist.

Einige der CORBA-Entwurfsmuster eignen sich ausgezeichnet für die Beschreibung der Vorgehensweise und der Ziele des Projektes ANICA. Daher werden nun zuerst einige dieser Muster vorgestellt und danach die im Projekt ANICA entwickelte CAx-Integrationsarchitektur im Kontext dieser Entwurfsmuster beschrieben.

4.1. Common Interface

Die Intention eines *Common Interface* liegt darin, Interoperabilität, Austauschbarkeit, Entkopplung und Wiederverwendung von Softwaremodulen, die unabhängig voneinander entwickelt wurden, zu unterstützen. Dieses Entwurfsmuster ist anwendbar, wenn es möglich ist, eine gemeinsame und einheitliche Abstraktion der Programmierschnittstellen der zu integrierenden Softwaremodule in Form einer Menge von standardisierten Softwareschnittstellen (Common Interfaces) zu spezifizieren. Grundsätzlich ist es möglich, Common Interfaces auf der Basis des kleinsten gemeinsamen Nenners oder über die jeweils besten Lösungen für Teilbereiche zu definieren. Möglich ist auch, mit dem kleinsten gemeinsamen Nenner zu beginnen und diese Lösung sukzessive um hochwertige Nischenfunktionalität zu erweitern.

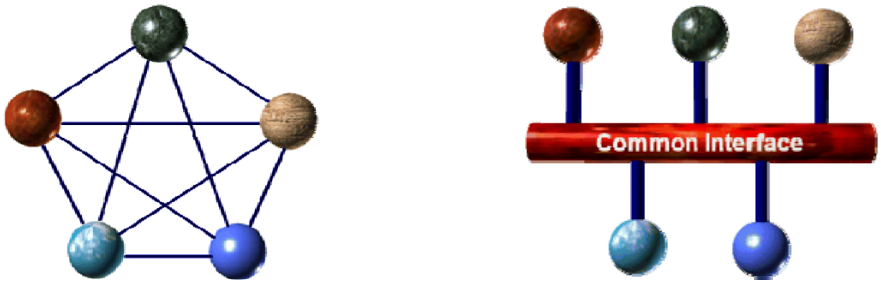


Abbildung 2: Bilaterale Custom Interfaces und ein Common Interface

Im Gegensatz zu *Custom Interfaces*, d.h. bilateralen systemspezifischen Schnittstellen zwischen je zwei Systemen, wird so nur eine gemeinsame bidirektionale Schnittstelle mit N Anbindungen statt O(N*N) bidirektionalen Schnittstellen benötigt (Abbildung 2). Außerdem erfordert das Hinzufügen eines zusätzlichen Systems bei Verwendung eines Common Interface nur eine neue Anbindung an die gemeinsame Schnittstelle statt N neuer Schnittstellen, wobei die bisherige Schnittstelle (im Idealfall) völlig unverändert bleibt bzw. erweitert wird, ohne den bestehenden Teil zu modifizieren.

Spezialisierte, produktabhängige und proprietäre Schnittstellen bezeichnet man auch als *vertikale Schnittstellen*. Im Gegensatz dazu spricht man bei Schnittstellen, die auf eine ganze Kategorie bzw. Klasse potentieller Subsysteme anwendbar sind (und nicht nur auf eine Instanz) von *horizontalen*, d.h. generalisierten Schnittstellen.

4.2. Architecture Mining

Ein Common Interface kann, z.B. für eine Menge von bezüglich ihrer Funktionalität überlappenden CAx-Systemen, durch einen *Architecture Mining* Prozeß abgeleitet werden. Architecture Mining ist ein bottom-up Ansatz und Prozeß, der Design-Wissen aus bestehenden Implementierungen erfaßt und nutzt. Dabei werden die „Nuggets“ des

vorhandenen und in der Praxis erprobten Design-Wissens entdeckt, extrahiert und verfeinert. Dies geschieht in der Regel in folgenden Schritten:

- Schritt 1: Analyse der Application Programming Interfaces (APIs) der beteiligten Systeme
- Schritt 2: Generalisierung zur Definition eines Common Interface
- Schritt 3: Verfeinerung (Wiederholung der bisherigen Schritte)

4.3. Komponentenarchitektur

Die Intention einer *Komponentenarchitektur* besteht sowohl in der flexiblen Substituierbarkeit von Softwaremodulen, die entweder statisch (zur Kompilierzeit) oder dynamisch (zur Laufzeit) erfolgen kann, als auch in der Förderung der Konfigurierbarkeit von Systemen.

Nachdem jetzt eine Reihe von CORBA-Entwurfsmustern präsentiert wurde, wird nun unter Verwendung der vorgestellten Entwurfsmuster ein Projekt zur Integration heterogener CAx-Systeme beschrieben.

5. Die CAx-Integrationsarchitektur ANICA

Das Projekt ANICA (**AN**alysis of Access **I**nterfaces of Various **CA**x Systems) wurde von Juli 1995 bis April 1998 am Lehrstuhl für Rechneranwendung in der Konstruktion der Universität Kaiserslautern durchgeführt. Es wurde von der Stiftung Rheinland-Pfalz für Innovation, der Deutschen Automobilindustrie (Audi, BMW, Mercedes Benz, Porsche, Volkswagen), der Firma ProSTEP sowie von den Systemherstellern Computervision, IBM/Dassault Systèmes, Intergraph, Matra Datavision, Spatial Technology und Tecmath unterstützt. Dieses Projekt sollte die Machbarkeit der vom Arbeitskreis 'CAD/CAM-Strategien' der Deutschen Automobilindustrie vorgeschlagenen neuen CAx-Systemarchitektur [Dankwort et al. 1994] verifizieren.

5.1. Der CAx-Objektbus

Der grundlegende Ansatz des Projektes war es, die Schnittstellen konventioneller CAx-Systeme zu analysieren, um daraus eine einheitliche gemeinsame Zugriffsschnittstelle, d.h. in der Sprache der Entwurfsmuster ein Common Interface in Form einer horizontalen Schnittstelle, für CAx-Systeme zu entwickeln.

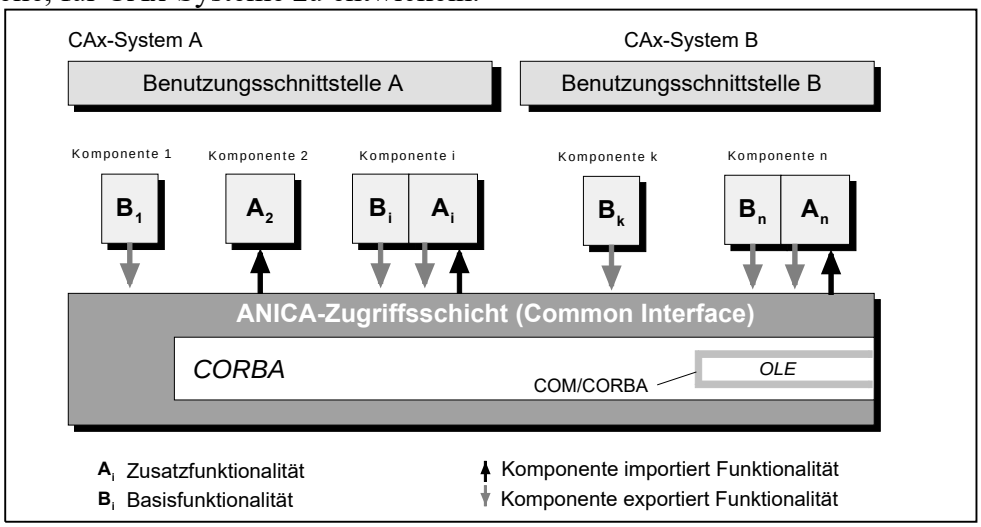


Abbildung 3: Die CAx-Integrationsarchitektur ANICA

Die Zielanwendung besteht dabei in einem verteilten CAx-System, das sich aus Komponenten verschiedener Anbieter bzw. Systemhersteller und einer einheitlichen Zugriffsschicht zu (bzw. einer Schnittstelle zwischen) diesen Komponenten zusammensetzt (Abbildung 3).

Dadurch soll ein Komponentensystem entstehen, das dem Prozeßbedarf entsprechend konfiguriert werden kann und dessen Ganzes mächtiger als die Summe seiner Teile ist. Hierbei stellen die einzelnen Komponenten ihre jeweilige Funktionalität als Dienste zur Verfügung. Die gemeinsame Schnittstelle verbindet die Komponenten und ermöglicht so den Zugriff auf die angebundenen CAx-Objekte aller integrierten Systeme. Daher kann man die einheitliche gemeinsame Schnittstelle auch als einen CAx-Objektbus bezeichnen.

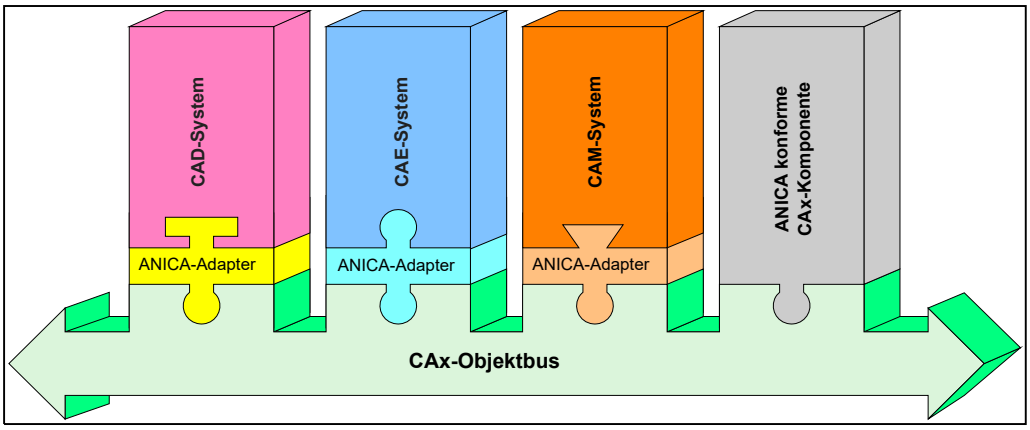


Abbildung 4: Der ANICA CAx-Objektbus

Die Anbindung an einen solchen CAx-Objektbus kann dabei auf zwei verschiedene Arten erfolgen (Abbildung 4):

- Bei der Anbindung von *Legacy Systems*, d.h. bereits bestehenden komplexen Softwaresystemen, die nicht im Hinblick auf eine solche Nutzung entworfen wurden, geschieht die Kopplung über die Implementierung systemspezifischer Adapter (Wrapper). Diese Adapter bilden das jeweilige API des anzubindenden CAx-Systems auf das Common Interface des CAx-Objektbusses ab.
- Dagegen können zukünftige CAx-Komponenten, d.h. Softwarebausteine, die eine bestimmte CAx-Funktionalität anbieten und deren Schnittstellen speziell auf das Common Interface abgestimmt wurden, direkt angekoppelt werden.

Der CAx-Objektbus besteht intern aus einer Menge von in IDL definierten Schnittstellen, die zusammen ein Common Interface bilden, und einer CORBA-Implementierung, die die Infrastruktur für den Zugriff über das Common Interface bereitstellt. Über COM/CORBA sind auch OLE- bzw. (D)COM-basierte Anwendungen an den CAx-Objektbus anbindbar.

5.2. Analyse und Design von CAx-Schnittstellen im Projekt ANICA

Die Schnittstellen des CAx-Objektbusses entsprechen einer Menge von CAx-Klassen. Die Definition dieser Klassen orientiert sich im Bereich der Daten an der Hierarchie der STEP-Entities des AP 214. Da diese STEP-Entities jedoch reine Datenbeschreibungen darstellen, müssen noch geeignete Methoden hinzugefügt werden, um vollständige und sinnvolle CAx-Klassen zu bilden. In einem pragmatischen Ansatz entstehen diese Methoden in einem Architecture Mining Prozeß aus der Analyse und dem Vergleich der verschiedenen Ausprägungen semantisch äquivalenter oder zumindest ähnlicher Funktionen (d.h. deren Signaturen) in den APIs der untersuchten Systeme. Im Rahmen des Projektes ANICA wurden daher die CAD-Systeme CATIA, DesignPostDrafting (Pelorus), SolidEdge (Jupiter), die CAD-Modellierkerne bzw. -Entwicklungsumgebungen ACIS und CAS.CADE/SF, sowie das anthropometrische Menschmodell RAMSIS analysiert. Ausgehend von den Entities der Teilmenge CC1 des STEP AP 214 wurde eine Hierarchie von CAx-Klassen definiert, die zu Beginn nur als reine Datenbeschreibungen vorlagen (Abbildung 5).

Nun wurden die einzelnen Systeme daraufhin untersucht, welche der internen Datenstrukturen dieser Systeme welchen STEP-Entities entsprechen. Basierend auf der Menge der so gefundenen Datenstrukturen wurde die von den Systemen angebotene Funktionalität für diese Datenstrukturen erfaßt und untersucht. Dazu wurden für jedes System Softwarewerkzeuge (Perl-Scripte etc.) zur Unterstützung des Analyseprozesses entwickelt, mit denen sowohl Quelltexte als auch Hilfetexte ausgewertet wurden.

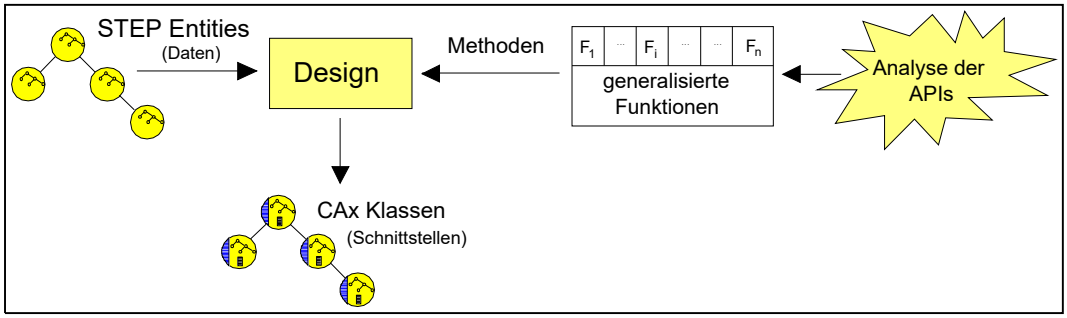


Abbildung 5: Ablauf des Designs der CAx-Klassen bzw. -Schnittstellen

Bei der Analyse traten Probleme vor allem aufgrund unterschiedlicher Formatierungen sowie fehlender oder inkonsistenter Informationen innerhalb der Dokumente auf, die eine manuelle Nachbearbeitung erforderlich machten. Die betrachteten Signaturen wurden bzgl. Funktionsname, Rückgabewert, Eingabe-, Ausgabe- und Kontrollparameter verglichen, in einer Datenbank erfaßt und Funktionen[1] gleicher oder ähnlicher Semantik in den verschiedenen Systemen identifiziert. Ein manuell durchgeführter Normalisierungsprozeß, in dem die unterschiedlichen Ausprägungen einer Funktion generalisiert (bzw. unifiziert) wurden, lieferte dann die Definitionen vereinheitlichter Funktionen und damit die Methoden der einzelnen Schnittstellen bzw. CAx-Klassen. So entstand ein Namensraum vereinheitlichter CAx-Funktionssignaturen. Dabei bildete die Menge der Parameter einer generalisierten Funktion in der Regel eine Obermenge der Parameter der systemspezifischen Ausprägungen dieser Funktion.

5.3. Ein Beispiel für das Design einer CAx-Schnittstelle

Ein Beispiel für die Definition einer CAx-Klasse bzw. deren Schnittstelle zeigt Abbildung 6. Ausgehend vom STEP Entity *cartesian_point* wird ein interface *cartesian_point* in IDL definiert, das über Attribute verfügt, die äquivalent zu denen des STEP-Entity sind. Diese rudimentäre Klassendefinition wird dann um Methoden zur Manipulation der Attribute der Klasse angereichert, von denen im Beispiel nur einige dargestellt sind.

```
TYPE length_measure = REAL;
END_TYPE;

ENTITY cartesian_point
  SUBTYPE OF (point);
  coordinates : LIST[1:3] OF length_measure;
END_ENTITY;
```

```
typedef double length_measure;

interface cartesian_point : point {
  attribute sequence<length_measure, 3> coordinates;

  void Create_From_Other (in ac_cartesian_point other);
  void Create_From_Parameters(in ac_REAL x, in ac_REAL y, in ac_REAL z);
  void Get_Coordinates (out ac_REAL x, out ac_REAL y, out ac_REAL z);
  void Set_Coordinates (in ac_REAL x, in ac_REAL y, in ac_REAL z);
  void Transform (in ac_transformation trsf);
  ac_REAL Get_Distance (in ac_cartesian_point other);
  // ...
};
```

Abbildung 6: Beispiel eines STEP-Entity in EXPRESS und der zugehörigen CAx-Schnittstelle

Da jedes System eigene Datentypen und Funktionen verwendet, ist es notwendig, diese in einem Mapping-Prozeß auf das Common Interface abzubilden. Die systemspezifischen

[1] Der Begriff *Funktion* wird hier im Sinne von Funktionalität verwendet. Bei einer Funktion kann es sich daher in diesem Zusammenhang um eine (klassische) Funktion, eine Prozedur oder eine Methode einer Klasse handeln.

Datentypen werden dabei auf die entsprechenden STEP-Entities abgebildet. Für die einzelnen Signaturen der zugeordneten Funktionen muß ebenfalls für jedes System ein Mapping auf das Common Interface und ein Re-Mapping vom Common Interface auf die systemspezifischen Funktionen definiert werden. Eine automatisierte Ausführung der Mappings und Re-Mappings ist dabei nicht ohne weiteres möglich, da neben syntaktischen vor allem semantische Informationen berücksichtigt werden müssen.

Ein Beispiel soll die Vorgehensweise verdeutlichen (Abbildung 7): zwei verschiedene Systeme bieten semantisch äquivalente Funktionen (*create_rectangle* und *cr_rect*) zur Erzeugung eines Rechtecks an. Die Funktion *create_rectangle* erhält als Eingabe vier Fließkommazahlenwerte, die die Koordinaten zweier gegenüberliegender Eckpunkte des zu erzeugenden Rechtecks spezifizieren.

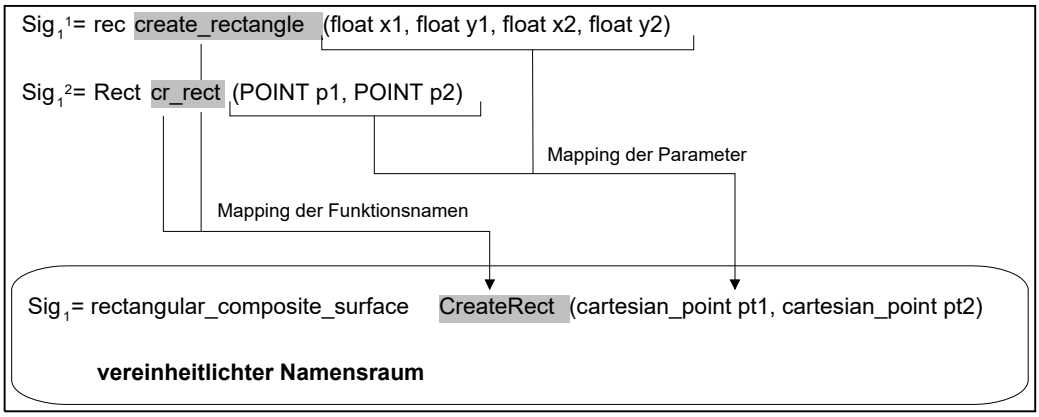


Abbildung 7: Ein Beispiel für das Mapping von Funktionssignaturen

Dagegen erwartet die Funktion *cr_rect* zwei Eingabewerte des systemspezifischen Typs *POINT*, die ebenfalls die Eckpunkte des zu erzeugenden Rechtecks repräsentieren. Die beiden systemspezifischen Signaturen werden auf die generalisierte Signatur abgebildet. Dabei muß für die erste Signatur zusätzlich eine Zuordnung der vier Fließkommaparameter zu den generalisierten Parametern des Typs *cartesian_point* erfolgen.

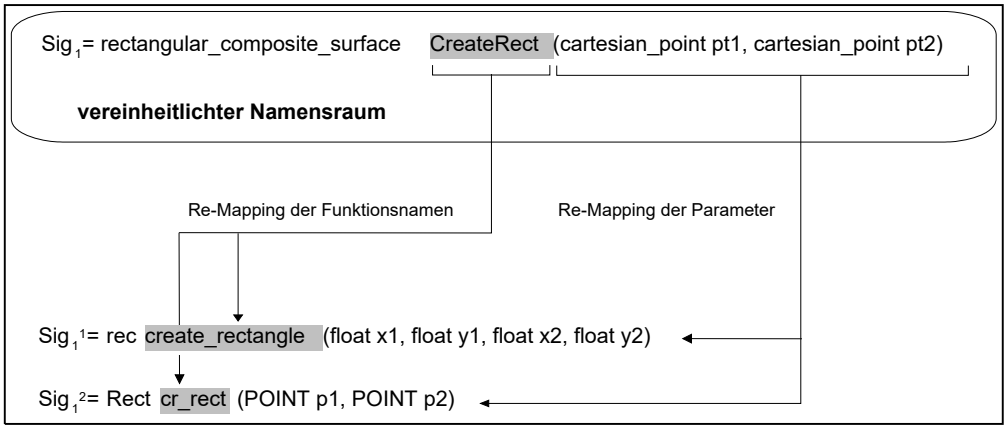


Abbildung 8: Ein Beispiel für das Re-Mapping von Funktionssignaturen

Nun muß in einem inversen Prozeß noch ein Re-Mapping der generalisierten Signatur auf die beiden systemspezifischen Signaturen durchgeführt werden (Abbildung 8).

Nachdem die CAx-Integrationsarchitektur ANICA vorgestellt wurde, folgt nun die Beschreibung einer ersten konkreten Umsetzung dieser Architektur in die Praxis. Das im folgenden beschriebene Anwendungsprojekt ProDMU zeigt dabei nur eines von vielen möglichen Einsatzgebieten der ANICA-Architektur.

6. Das Anwendungsprojekt ProDMU

Das Projekt ProDMU ist ein Anwendungsprojekt im Rahmen des Projektes ANICA und wird vom Lehrstuhl für Rechneranwendung in der Konstruktion der Universität Kaiserslautern in Zusammenarbeit mit einem Automobilhersteller durchgeführt. Das Projekt ProDMU ist ein Versuch, den Ablauf von virtuellen Einbauuntersuchungen im Rahmen des kooperativen Digital Mock-Up bei dem Automobilhersteller hinsichtlich der Qualität der Kommunikation und der Effizienz der Arbeitsabläufe zu verbessern.

Der Begriff des *Digital Mock-Up* (DMU) entstand ursprünglich aus der Idee, analog zu einem physischen Modell (Mock-Up) ein rechnerinternes Modell für die Baubarkeitsüberprüfung eines Produktes zu verwenden, um die Zahl der physischen Mock-Ups möglichst zu minimieren. In der Automobilindustrie wird DMU heute als Prozeß verstanden, in dem ohne physische Prototypen der Zusammenbau von Komponenten verschiedener Firmen bis hin zu Funktionstests und Simulationen durchgeführt wird (Trend zum *'Digital Car'*).

6.1. Virtuelle Einbauuntersuchungen im heutigen industriellen Ablauf

Bei dem betreffenden Automobilhersteller wird im Bereich der Fahrzeugentwicklung u.a. der virtuelle Zusammenbau der einzelnen Bauteile im Motorraum eines Pkw durchgeführt. An der Durchführung dieser virtuellen Einbauuntersuchungen sind zwei CAD-Systeme beteiligt: CATIA von IBM/Dassault Systèmes und Pro/ENGINEER (Pro/E) von Parametric Technology Corporation (PTC). Dabei wird Pro/E bei der Aggregatekonstruktion (Motor, Lichtmaschine etc.) und CATIA für alle übrigen Konstruktionsaufgaben sowie für Kollisions- und Einbauuntersuchungen verwendet. Im Rahmen des bisherigen Ablaufs der virtuellen Einbauuntersuchungen kommunizieren die beiden Systeme nicht direkt miteinander, sondern nur mittels dateibasiertem Datenaustausch (linke Seite der Abbildung 9):

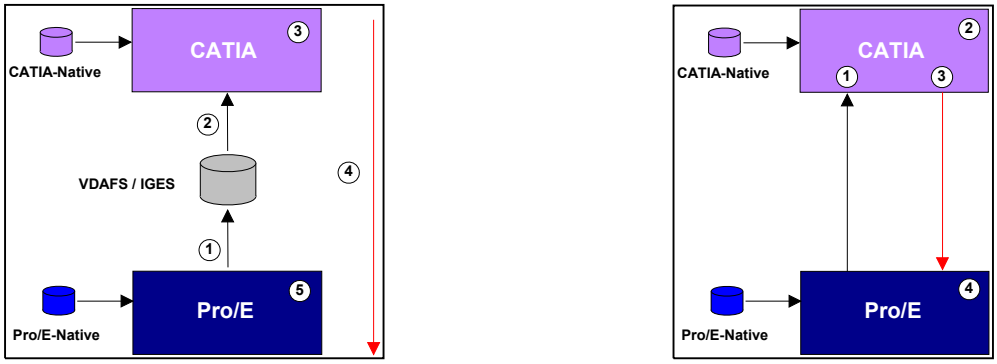


Abbildung 9: Bisheriger Ablauf und optimierter Ablauf der virtuellen Einbauuntersuchungen

- Zuerst werden die zu übertragenden Daten von Pro/E in einem neutralen CAD-Datenformat (IGES oder VDA-FS) in einer Datei abgelegt (Schritt 1).
- Danach wird diese Datei von CATIA eingelesen (Schritt 2).
- Anschließend kann die eigentliche Kollisionsuntersuchung mit bereits von CATIA geladenen Modellen im CATIA-Native Format erfolgen (Schritt 3).
- Treten Kollisionen oder Durchdringungen zwischen den untersuchten Bauteilen auf, so wird der Pro/E-Konstrukteur hierüber mündlich oder schriftlich informiert (Schritt 4).
- Der Pro/E-Konstrukteur kann dann die erforderlichen Änderungen vornehmen (Schritt 5) und danach den aktualisierten Datensatz erneut exportieren (Schritt 1).

Oft muß der beschriebene Zyklus mehrfach durchlaufen werden, bis alle Kollisionen beseitigt sind, so daß ein beträchtlicher Zeit- und Kostenaufwand entsteht.

Dieser Workflow ist auch unter verschiedenen anderen Gesichtspunkten nicht optimal. So stellt die zusätzliche Speicherung von Modelldaten in einem neutralen Format (Schritt 1) eine redundante Datenhaltung mit allen damit verbundenen Konsistenzproblemen dar. Weiterhin stören die für den expliziten dateibasierten Datenaustausch notwendigen Tätigkeiten (Schritt 1 und 2) den Arbeitsablauf der Konstrukteure, da sie für die eigentlichen Konstruktionsaktivitäten unproduktiv bzw. nicht relevant sind und den kreativen Konstruktionsprozeß immer wieder unterbrechen und verzögern. Dabei ist die Kommunikation zwischen den beteiligten Personen bzgl. der notwendigen Änderungen dadurch erschwert, daß das Ergebnis der Kollisionsprüfungen nur mündlich bzw. textuell, z.B. per E-Mail, aber nicht in Form eines CAD-Modells (oder zumindest eines Pixelbildes) weitergegeben wird, wodurch eine inhärente Unsicherheit gegeben ist. Für einen optimierten Ablauf sind folgende Forderungen zu erfüllen:

- Die Lösung muß kompatibel zu STEP AP 214 sein, da dies der zukünftig maßgebende Standard für die Datenkopplung in der deutschen Automobilindustrie sein wird.
- Redundante Datenhaltung muß möglichst vermieden werden.
- Die Performanz einer praktikablen Lösung muß, bzgl. der durchschnittlichen Dauer des gesamten Arbeitsablaufs, möglichst gleich gut oder besser als die bisherige sein.
- Die Modellgröße von Testmodellen muß eine praxisrelevante Komplexität aufweisen.
- Die unterschiedlichen numerischen Genauigkeiten der zu koppelnden Systeme sind bei der Konvertierung von Modellen zu berücksichtigen.

6.2. Der optimierte Ablauf der virtuellen Einbauuntersuchungen

Bei der im Projekt ProDMU entwickelten neuen Vorgehensweise ist es möglich, für die Kollisionsprüfungen von Beginn an nur diejenigen Elemente oder Bereiche der Teilmodelle zu übertragen, die für die Untersuchung relevant sind, z.B. nur die in einer definierten Umgebung (z.B. einer dreidimensionalen Box) um bestimmte Bauteile liegenden Flächen angrenzender Teilmodelle. Dadurch läßt sich die zu übertragende Datenmenge deutlich reduzieren. Der optimierte Workflow der virtuellen Einbauuntersuchungen ist gegenüber der alten Vorgehensweise deutlich vereinfacht (rechte Seite der Abbildung 9):

- Wahlweise werden nur bestimmte Teilbereiche oder das vollständige Pro/E-Modell aus CATIA heraus angefordert, übertragen und in CATIA dargestellt (Schritt 1).
- Danach wird die eigentliche Einbauuntersuchung durchgeführt (Schritt 2) und die Kollisionen markiert (Redlining).
- Anschließend werden die Markierungen zu Pro/E übertragen (Schritt 3) und damit für den Pro/E-Konstrukteur in dessen Modell sichtbar.
- So kann der Pro/E-Konstrukteur dann die notwendigen Änderungen schnell und präzise durchführen (Schritt 4). Gegebenenfalls kann nun wieder mit Schritt 1 begonnen werden.

Zu beachten ist, daß die Online-Verbindung zwischen den Modellen in CATIA und Pro/E während der ganzen Zeit bestehen bleibt, d.h. bei Änderungen am Orginalmodell in Pro/E müssen nur die geänderten Teilbereiche zu CATIA übertragen werden.

Werden von Beginn an nur ausgewählte Teilbereiche der Pro/E-Modelle übertragen, so bietet die Online-Kopplung im Vergleich zum bisherigen dateibasierten Datenaustausch mit neutralen Datenformaten eine bessere Performanz. Bei der Erstübertragung vollständiger Modelle liegt die Performanz der in ProDMU entwickelten Lösung derzeit noch hinter der bisherigen, ist aber bei Aktualisierung nach der Durchführung von Änderungen - und somit auch über den gesamten Arbeitsablauf gesehen - meist besser.

Außerdem bietet der in ProDMU entwickelte optimierte Ablauf folgende weitere Vorteile:

- Verbesserung der Kommunikation zwischen den beteiligten Konstrukteuren durch die vollständige Übermittlung der (auch im Originalmodell sichtbaren) Problembereiche.
- Erweiterung der CATIA-Funktionalität durch die Online-Anbindung der Pro/E-Funktionalität (für in beiden Systemen bearbeitbare STEP-kompatible Daten).
- Vermeidung der Unterbrechung der Konstruktionstätigkeit durch den dateibasierten Export und Import von Daten.
- Die zusätzliche und redundante Speicherung der Daten in einem neutralen Format (wie in Schritt 1 beim bisherigen dateibasierten Datenaustausch) entfällt.

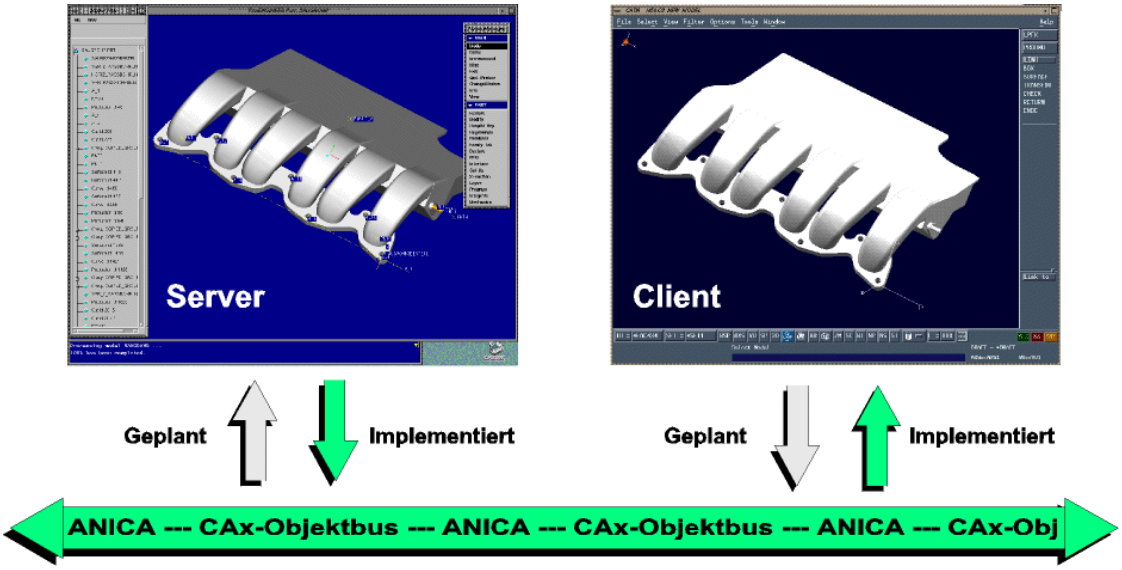
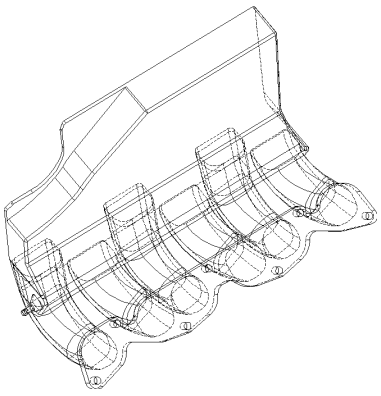


Abbildung 10: Online-Verbindung der CAD-Systeme Pro/E (Server) und CATIA (Client)

In Abbildung 10 ist für Server (Pro/E) und Client (CATIA) des ANICA CAx-Objektbusses je ein Screenshot der graphischen Benutzeroberflächen mit dem Modell eines 'Saugrohrs' aus dem Motorraum eines Pkw im schattierten Modus zu sehen. Im derzeitigen Prototyp ist die Richtung von Pro/E (Server) zu CATIA (Client) für Entities aus dem Bereich der Kurven- und Flächendarstellung des STEP AP 214 implementiert.



STEP Entity	Anzahl
trimmed parametric	359
curves on parametric	380
rational b-spline surface	359
rational b-spline curve	2191
transformation matrix	279
line	1072
composite curve	760
circular arc	279

Abbildung 11: Drahtmodell und technische Daten des Testmodells 'Saugrohr'

Abbildung 11 beinhaltet die Darstellung des 'Saugrohrs' als Drahtmodell und ein Tabelle mit den technischen Daten zur Struktur dieses Testmodells. Die Modellgröße ist dabei durchaus repräsentativ für das Umfeld der virtuellen Einbauuntersuchungen, wie es bei dem Automobilhersteller durchgeführt wird.

7. Fazit und Ausblick

Der im Anwendungsprojekt ProDMU entwickelte Prototyp zeigt die technische Machbarkeit der Umsetzung der im Projekt ANICA ausgearbeiteten CAx-Integrationsarchitektur. Die Integration wurde dabei im heterogenen Umfeld der Programmiersprachen FORTRAN, C, C++, einer CORBA-Implementierung (Orbix) und zwei Entwicklungsumgebungen für zwei technisch sehr unterschiedliche CAD-Systeme durchgeführt. Durch einen Einsatz dieser Technologie kann der Ablauf der virtuellen Einbauuntersuchungen deutlich verbessert werden. Die geplante Erweiterung des Prototypen um die Gegenrichtung, d.h. CATIA als Server und Pro/E als Client, erscheint momentan fraglich, da sich die Programmierschnittstelle von Pro/Engineer als weitgehend unidirektional erwiesen hat, d.h. das *Auslesen* von Daten ist i.d.R. möglich, das API bietet jedoch derzeit keine Möglichkeiten zur direkten *Erzeugung* von Freiformgeometrien.

Die erarbeiteten Lösungen wurden von Beginn an auf Erweiterbarkeit ausgelegt, d.h. die Anbindung weiterer CAx-Systeme ist ohne prinzipielle Änderungen an der Architektur oder der bisherigen Implementierung möglich, soweit die APIs der Systeme eine entsprechende Offenheit bieten. Die virtuellen Einbauuntersuchungen im Rahmen des Digital Mock-Up bilden daher nur eines von vielen möglichen Einsatzgebieten der im Projekt ANICA entwickelten CAx-Integrationsarchitektur, wie etwa CSCW (Computer Supported Co-operative Work) oder die Integration mehrerer in einer Prozeßkette aufeinanderfolgender CAx-Systeme. Insbesondere wenn zukünftig Systemhersteller dazu übergehen sollten, maßgeschneiderte CAx-Komponenten auf den Markt zu bringen, kann der vorgestellte Ansatz einen wichtigen Beitrag zur Optimierung der Prozeßketten leisten.

8. Literatur

[Curtis 1997] D. Curtis: *"Java, RMI and CORBA"*, Object Management Group (OMG) White Paper, http://www.omg.org/news/wpjava.htm, 1997

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

[Mowbray & Malveau 1997] T. J. Mowbray, R. C. Malveau: “*CORBA Design Patterns*”, John Wiley & Sons, 1997

[Orfali et al. 1998] R. Orfali D. Harkey, J. Edwards: *"Instant CORBA"*, Addison-Wesley, 1998

[OMG 1996a] Object Management Group (OMG), Manufacturing Domain Task Force (MfgDTF): “*Manufacturing Enterprise Systems – A White Paper*”, OMG Document mfg/96-01-02, February 1996

[OMG 1996b] Object Management Group (OMG*)*), Manufacturing Domain Task Force (MfgDTF)*: “Product Data Management Enablers RFP”*, OMG Document mfg/96-08-01, August 1996

[OMG 1998] Object Management Group (OMG): *“The Common Object Request Broker: Architecture and Specification”*, Revision 2.2, February 1998

[Schenck & Wilson 1994] D. Schenck, P. Wilson : *"Information Modeling the EXPRESS Way"*, Oxford University Press, January 1994