Von Monolithen zu Komponenten: CAx-Architekturen im Wandel

Prof. Dr. C. Werner Dankwort
Dipl.-Ing. Andrzej Janocha
Lehrstuhl für Rechneranwendung in der Konstruktion
Universität Kaiserslautern
Gottlieb-Daimler-Straße
D-67663 Kaiserslautern

Zusammenfassung

Mit der schnellen Verbreitung der CAx-Techniken in der deutschen Automobilindustrie wächst die Notwendigkeit einer besseren Integration der CAx-Systeme in die Prozeßketten und der Beherrschung der Produktinformationsflüsse. Aufgrund dieser Tatsachen ist in den letzten Jahren ein Wandel der CAx-Systemarchitekturen von geschloßenen, monolithischen zu offen integrierten Systemen erkennbar. Im folgenden wird dieser Prozeß sowie dessen Implikationen auf die Anwendung und auf die Systemhersteller analysiert.
Ausgehend von der Initiative der deutschen Automobilindustrie wurde das Projekt ANICA (*An*alysis of *I*nterfaces of various *CA*D/CAM-Systems) gestartet. In diesem Projekt werden die Schnittstellen zu den Systemkernen einiger CAx-Hersteller untersucht und ein Konzept für kooperierende CAx-Systeme in der Automobilindustrie wird entwickelt.

1. Einleitung

Die immer größere Durchdringung der Automobilindustrie mit CAx-Techniken und die wachsende Komplexität der CAx-Anwendungen führt mittlerweile zu einem kritischen Umdenken in den betroffenen Bereichen. Aufgrund der noch oft vorhanden veralteten Technologie sind die großen monolithischen Systeme kaum noch wartbar und erweiterbar. Dies aber steht im Widerspruch zu immer schneller wachsenden Anforderungen nach branchenspezifischen und wirtschaftlichen Lösungen. Vor allem aber ist die *Integration* der CAx-Systeme in den Prozessketten der Automobilindustrie und die Durchgängigkeit der CAx-Daten nicht ausreichend.
Da sich diese Defizite offensichtlich nicht auf der Basis von herkömmlichen Systemen ausgleichen lassen, sind die Systemhersteller gefordert, eine neue Generation von CAx-Systemen zu entwickeln. So ist der Generationswandel auf drei wesentliche Ursachen zurückzuführen:

- den Wunsch der Anwender nach mehr Integration und Datendurchgängigkeit
- die nicht mehr tragbaren Kosten für Erweiterung und Wartung der alten Systeme
- die allgemeine Entwicklung der Informationstechnologie (IT), insbesondere im Bereich von Objektorientierung und verteilten Systeme

Der Generationswechsel bedeutet nicht Erweiterung der herkömmlichen Systeme um neue Funktionalitäten oder Einführung neuer Methoden wie Parametrik oder Feature-Technologie, vielmehr ist hiermit ein Wandel der den Systemen zugrundeliegenden Organisation – der Systemarchitektur gemeint.
Im Kontext von Software-Engineering hat der Begriff „Architektur“ keine formale Definition. Es existiert aber ein allgemein akzeptiertes Grundverständnis dieses Begriffs als „Organisation von Software-Systemen“ (And93). In diesem Sinne gehören zur Architektur eines Software-Systems:

- Auswahl geeigneter Systemkomponenten
- Interaktion zwischen diesen Komponenten
- Gruppierung der Komponenten in Subsysteme
- Beschreibung der Komponenten und Subsysteme, sowie deren Interaktion

Außer der Struktur, werden innerhalb einer Architektur noch andere Aspekte, wie Funktionalität, Laufzeitverhalten, Bedienkomplexität usw. berücksichtigt. An eine Softwarearchitektur werden diverse Anforderungen gestellt. Die wichtigsten davon sind:

- Offenheit
- Skalierbarkeit
- Portabilität
- Performance
- Robustheit
- Interoperabilität

- Sicherung der Investition
- Sicherheit

Die Integrationsfähigkeit der Systeme steht in einem direkten Zusammenhang mit der Architektur. In (MZ95) werden folgende Stufen der „Integrationsreife“ innerhalb einer Organisation bzw. Anwenderfirma definiert:

1. kommerzielle, von Softwaredistributoren erhältliche Lösungen
2. diverse Methoden, aufbauend auf Eingenentwicklungen
3. ausgereifte Remote Procedure Calls (wie OSF DCE[1])
4. verteilte Objekte (CORBA[2])
5. Frameworks
6. Standardarchitekturen

Organisationen der 1. Stufe verfügen weder über aufwertende Integration noch über kundenspezifische Software. Sie verwenden kommerziell erhältliche Softwarepakete.
Organisationen der 2. Stufe entwickeln eigene Software, diese bleibt aber von der verfügbaren Integrationstechnologie (TCP/IP, Sockets, ONC RPC[3]) weitgehend unberührt. Z.B. kann ein gewisser Grad an Integration von CAx-Systemen aufgrund der Benutzung des gleichen Systemkerns erreicht werden.
Organisationen der 3. Stufe haben die Notwendigkeit der Integration erkannt und benutzen zu diesem Zweck ausgereifte RPC-Mechanismen, wie z.B. das OSF DCE.
Organisationen der 4. Stufe benutzen aktiv CORBA und profitieren von der ORB[4]-Technologie. Allerdings haben diese Organisationen kein Konzept für Softwarearchitektur und setzen die ORB-Technologie nur für die Aspekte der Verteilung und Heterogenität ein. Für sonstige Schnittstellen werden andere, zum Teil veraltete Mechanismen eingesetzt.Viele dieser Organisationen verwenden die CORBA-Spezifikation nicht konsequent und benutzen den ORB produktspezifisch.
Organisationen der 5. Stufe haben für ihre Entwicklungszwecke projektspezifische Frameworks entworfen und profitieren von den niedrigeren Kosten im gesamten Lebenszyklus der Software. Ein entscheidender Defizit besteht hier in der fehlenden Interoperabilität zwischen mehreren Projekten in einer Organisation.
Organisationen der 6. Stufe definieren selbst Standardarchitekturen, die über Projektgrenzen hinausgehen. Im Gegensatz zu produktspezifischen Implementierungen der ORB-Hersteller benutzen sie extensiv die CORBA-Spezifikation. Hierdurch wird diesen Organisationen ermöglicht, verschiedene Plattformen und sogar verschiedene Object Request Broker zu unterstützen und damit das Risiko technologischer Alterung zu vermindern.
Alle diese Überlegungen betreffen in erster Linie die CAx-Systemhersteller. Anwenderfirmen, die im CAx-Bereich keine eigene Softwareentwicklung durchführen, profitieren vom technolo-

1. Distributed Computing Environment der Open Software Foundation (OSF)
2. Common Object Request Broker Architecture der Object Management Group (OMG)
3. Open Network Computing Remote Procedure Call von Sun Microsystems
4. Object Request Broker

gischen Vorsprung der Systemhersteller nur indirekt, auch wenn diese zu den Organisationen der 5. oder 6. Stufe gehören.
Aufgrund der Vielzahl an Systemen verschiedener Hersteller in der Automobilindustrie (vergl. dazu (DKLPR94)), muß hier noch ein Schritt weiter gegangen werden: es werden Architekturen und Standards benötigt, die eine *systemherstellerübergreifende Integration* ermöglichen und die neue Strategie der Automobilindustrie für CAx-Systeme umsetzen. Diese Strategie beruht auf folgenden Prinzipien (Dank95):

- Integration der dynamischen Prozesse/Prozeßschritte
  Vereinfachung der Abläufe
- Beherrschung der Produktinformationsflüsse über den Gesamtprozeß
- Nutzung der optimalen CAx-Werkzeuge für jedes Anwendungsfeld
- Verfügbarkeit der benötigten CAx-Funktionalität, d.h. prinzipielle Erstellbarkeit und Integrierbarkeit einer benötigten Funktion
- Kommunizieren aller produkt- und prozeßbeschreibenden Daten
- Offenheit der Systeme

Im folgenden werden in chronologischer Reihenfolge die Architekturen einiger existierender sowie zukünftiger CAx-Systeme analysiert. Die Analyse wird sich im wesentlichen auf folgende Aspekte konzentrieren:

- Struktur der CAx-Systeme
- Durchgängigkeit der produktdefinierenden Daten
- Funktionalität
- Anpaßbarkeit des Funktionsumfangs an die Anwenderbedürfnisse (Customizing)

Zusätzlich wird die Erfüllung folgender Anforderungen analysiert:

- Offenheit
- Interoperabilität
- Sicherung der Investition

2. Monolithisch integrierte Systeme

Die zur Zeit in der Automobilindustrie eingesetzten Systeme entstanden in ihrer Grundkonzeption vor ca. 15 Jahren. Dementsprechend ist die Architektur dieser Systeme festgelegt. Es sind im wesentlichen vertikal aufgebaute Systeme mit sequentieller Steuerung, die gezwungenermaßen – und zwar jedes System für sich – das gesamte Spektrum der Funktionalität abdecken wollen (siehe Abb. 1).
Mit der wachsenden Funktionalität der Systeme wächst auch ihre Komplexität. Als Gradmesser für die Komplexität dient die Anzahl der Schnittstellen zwischen den Modulen, die jeweils

eine Funktionalität implementieren. Die Implementierungskosten einer neuen Funktionalität steigen exponentiell.

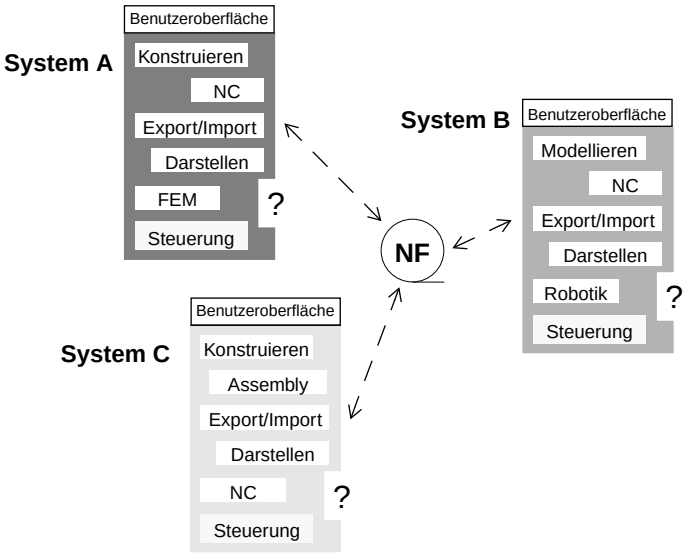

Abb. 1: Monolithisch integrierte CAx-Systeme

In (MZ95) werden solche Systeme charakterisiert als:

- monolithisch
- geschlossen (proprietäre Lösungen)
- strukturlos
- ohne wiederverwendbare Systemteile/Module
- langsam in der Entwicklung und in der Einführung
- kostspielig in der Wartung und Weiterentwicklung

Für die Weiterentwicklung oder branchenspezifische Anpassungen bieten diese Systeme eine Programmierschnittstelle an, die sich unstrukturiert aus den einzelnen Modulschnittstellen zusammensetzt. Die Implementierung neuer Funktionalitäten erfolgt durch softwaretechnische Manipulation. Dabei entstehen immer mehr spezifische Versionen, deren Einführung und Beherrschung sowohl beim Systemhersteller als auch beim Anwender große Kosten verursacht. Der Austausch von produktdefinierenden Daten innerhalb der (Anwender-) Organisation ist über bilaterale Schnittstellen zwischen den einzelnen Systemen oder über neutrale Formate, wie VDAFS, IGES oder STEP möglich (NF in der Abb. 1).
Eine qualitative Einschätzung monolithisch integrierter Systeme bzgl. der wichtigsten Kriterien ist in Abb. 1 dargestellt.

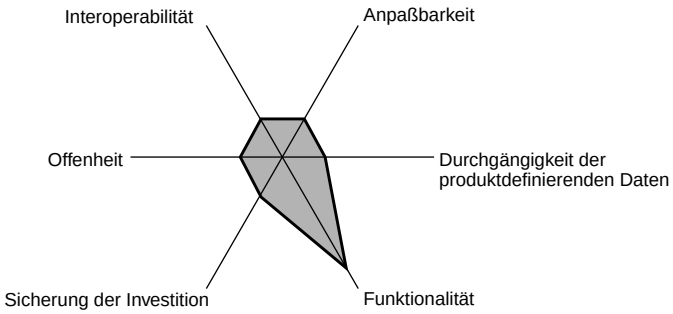

Abb. 2: Qualitative Einschätzung monolithischer CAx-Systeme

3. Offen integrierte Systeme

Die gegenwärtige Entwicklung der Informationstechnologie ermöglicht es, CAx-Systeme zu entwerfen, die mit den Defiziten monolithischer Systeme nicht mehr behaftet sind.
Der Paradigmenwechsel von prozeduraler zu objektorientierter Programmierung wird auch auf die Entwicklung komplexer Systeme übertragen. In Verbindung mit der Client/Server-Technologie resultiert daraus die verteilte objektorientierte Programmierung, auf deren Basis offen integrierte Systeme entstehen können.
Im Gegensatz zu monolithischen Systemen sind diese Systeme:

- offen
- klar strukturiert
- mit wiederverwendbaren Teilen/Modulen ausgestatet
- effektiver in Entwicklung und Einführung
- günstiger in Wartung und Weiterentwicklung

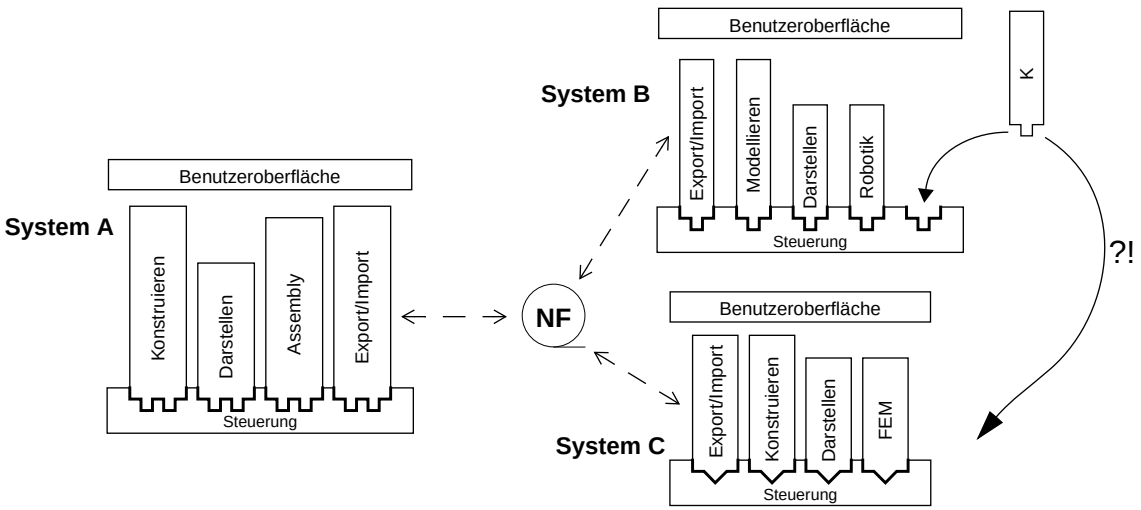

Abb. 3: Offen integrierte CAx-Systeme

Wie in Abb. 3 dargestellt, setzt sich ein offen integriertes System aus Komponenten und einer Integrationsplattform für diese Komponenten zusammen. Das System ist ereignisgesteuert.
Die Schnittstellen zu den Komponenten sind innerhalb eines Systems (= eines Herstellers) einheitlich. Dies ermöglicht das problemlose Hinzufügen oder Austauschen von Komponenten. Allerdings sind diese Systeme bezogen auf die Systemhersteller immer noch proprietär. Eine neue Komponente K paßt zwar in das System B, nicht aber in das System C. Das hat zu Konsequenz, daß der Anwender des Systems C nicht auf Objekte und Funktionalitäten der Komponente K zurückgreifen kann.

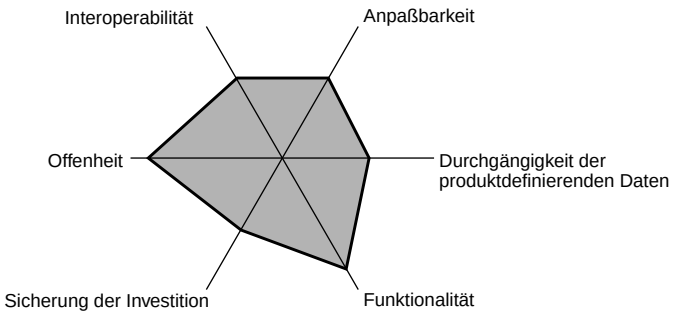

Abb. 4: Qualitative Einschätzung offen integrierter CAx-Systeme

Der Austausch produktdefinierender Daten zwischen Systemen in einer Organisation erfolgt wie bei monolithischen Systemen über bilaterale Schnittstellen oder über ein neutrales Format.

4. Entwicklungsplattformen für offen integrierte Systeme

Zum Zwecke der Entwicklung von offen integrierten Systemen haben mehrere CAx-Hersteller neue Programmierumgebungen und Frameworks entworfen. Einige davon sind bereits im Einsatz und mit ihrer Hilfe wurden neue CAx-Systeme erstellt. Zwei Beispiele hierfür sind:

- CAS.CADE/SF von Matra Datavision
- Jupiter von Intergraph

Auch bei anderen CAx-Herstellern (z.B. Computervision, Dassault Systemes) stehen ähnliche Entwicklungsplattformen zur Verfügung.

4.1 CAS.CADE Software Factory

CAS.CADE/SF ist die Entwicklungsplattform für zukünftige CAx-Produkte bei Matra Datavision. Sie beinhaltet folgende Elemente:

- eine offene und flexible Anwendungsarchitektur

- eine Entwicklungsumgebung
- eine Auswahl von wiederverwertbaren C++-Klassen
- eine Sammlung wiederverwertbarer Modulen

Das Modell der Applikationsarchitektur beschreibt drei Kategorien von Elementen:

- Präsentation, d.h. die äußere Erscheinung der Applikation und die Darstellung der manipulierten Objekte
- Funktion, d.h. die Funktionalität der Applikation
- Steuerung, die für die Konsistenz der Funktion und der Applikation sorgt.

Die Architektur einer CAS.CADE/SF-Applikation ist in Abb. 5 dargestellt.

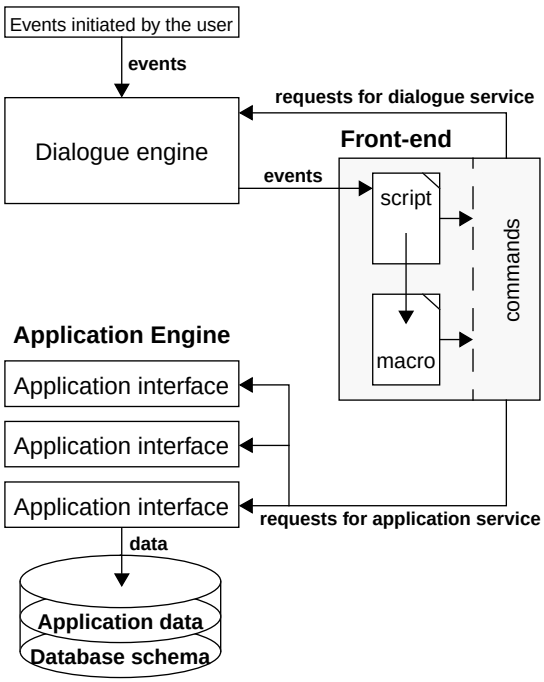

Abb. 5: Struktur einer CAS.CADE-Applikation (MATRA95)

Die Elemente dieser Architektur sind:

- eine sog. Dialogue-Engine
- eine oder mehrere sog. Application-Engines
- Front-end

Die Dialogue-Engine implementiert die Benutzerschnittstelle und empfängt alle vom Benutzer ausgehenden Events (Ereignisse), wie die Selektion durch das Anpicken eines Objekts.
Die Application-Engines stellen die eigentliche Funktionalität der Applikation zur Verfügung, wie das Erzeugen und Manipulieren von Objekten.
Das Front-end verarbeitet die Ereignisse, die von der Dialog-Engine kommen, indem es Requests (Anforderungen) an die Application-Engines und ggf. an die Dialogue-Engine schickt.

Eine Applikation besteht in der Regel aus einem Front-end und mehreren Application-Engines. Die einzelnen Engines werden als getrennte Prozesse implementiert.
Die Entwicklungsumgebung von CAS.CADE/SF beinhaltet mehrere Werkzeuge, die die Durchführung großer Projekte mit mehreren Entwicklern unterstützten. Während der Entwicklung kann auf die Ressourcen von CAS.CADE/SF zurückgegriffen werden, vor allem auf die wiederverwertbaren Klassenbibliotheken und die wiederverwertbaren Applikationsmodule.
Bei den wiederverwertbaren Applikationsmodulen handelt es sich um die 2D- und 3D-Viewers (Betrachter, Module, die geometrische Objekte visualisieren) und um das Datenbankmanagementsystem.
Die wiederverwertbaren Klassenbibliotheken beinhalten ca. 3000 C++-Klassen, die in Form von sog. Toolkits für den Entwickler verfügbar sind.
Auf Basis von CAS.CADE sind mehrere Produkte entstanden, u.a. EUCLID Designer, Nachfolger des Systems EUCLID 3.

4.2 Jupiter

Im Rahmen des Projekts Jupiter hat die Firma Intergraph eine Entwicklungsplattform für CAx-Applikationen auf Win32 (Windows 95 und Windows NT) geschaffen. Als Integrationsplattform benutzt Jupiter das *OLE for Design and Modeling* (OLE4D&M), eine Erweiterung des Konzeptes des Compound Document auf den technischen Bereich. So sind die Container des OLE4D&M in der Lage:

- 3D-Objekte aufzunehmen und zu verwalten
- 3D- und 2D-Objekte ineinander zu transformieren
- die Objekte lokalisieren

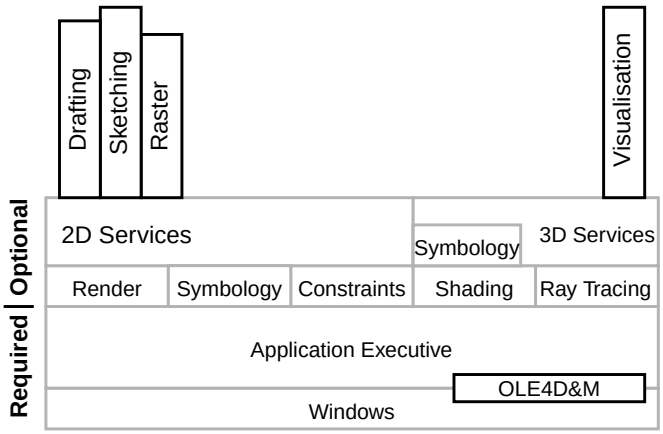

Abb. 6: Die Jupiter-Entwicklungsplattform (Send95)

Eine ausführliche Beschreibung des OLE4D&M ist in (Send95) zu finden.
Auf der Basis von Jupiter sind bereits mehrere Produkte entstanden, u.a. ein 3D-Modeller

SOLID EDGE und ein 2D-Programm für Zeichnungserstellung Imagineer Technical.

5. Kooperierende Systeme

„Verteilte objektorientierte Systeme werden in Zukunft zu Netzwerken kooperierender Objekte. Dabei werden miteinander kooperierende Objekte in Zukunft von unterschiedlichen Herstellern kommen.“ (BBK95).
Kooperierende Systeme sind ein weiterer Schritt in Richtung optimaler Ausnutzung des CAx-Potentials der Automobilindustrie. Komponenten verschiedener Hersteller werden hier über eine gemeinsame Integrationsplattform zu einem System verbunden. Die Schnittstellen zu den Komponenten sind standardisiert (Abb. 7). Ein CAx-System im traditionellen Sinne gibt es nicht. Vielmehr wird es durch eine anwendungsspezifische Konfiguration von Komponenten verschiedener Hersteller ersetzt.

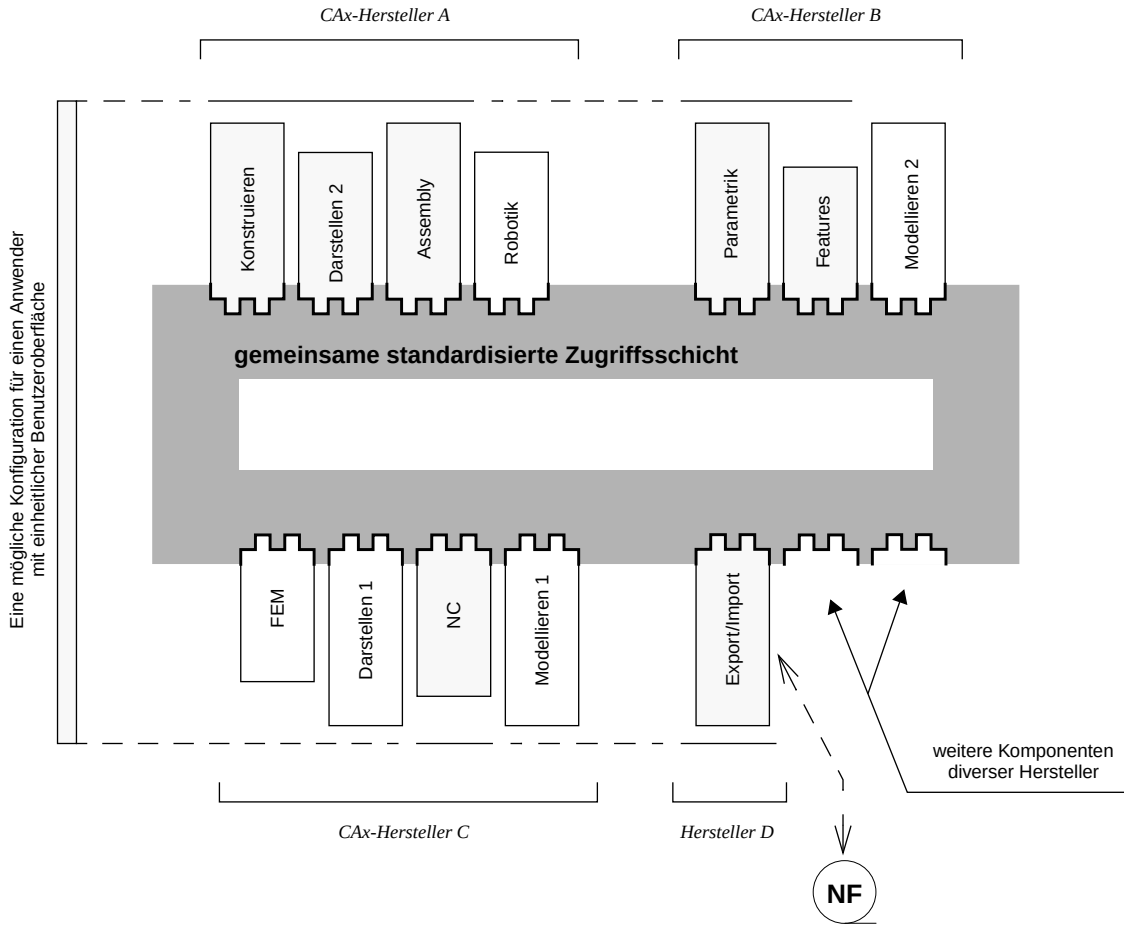

Abb. 7: Kooperierende CAx-Systeme

Kooperierende Systeme haben alle Vorteile offen integrierter Systeme. Die Anpaßbarkeit des Funktionsumfangs an die anwenderspezifische Bedürfnisse und die Sicherung der Investition

sind besonders stark ausgeprägt. Die hohe Anpaßbarkeit ermöglicht eine „schlanke“ Konfiguration des CAx-Systems, die eine geringe Bedienkomplexität und damit mehr Effizienz im Einsatz aufweist. Der Anwender lizenziert und bezahlt nur Dienste bzw. Komponenten, die er tatsächlich benutzt. Zusätzlich wird die Problematik des Versions-/Systemwechsels entschärft. Der Produktdatenaustausch wird direkt gewährleistet und muß nicht mehr auf dem Umweg über ein neutrales Format erfolgen.
In Abb. 1 ist die qualitative Einschätzung kooperierender CAx-Systeme dargestellt.

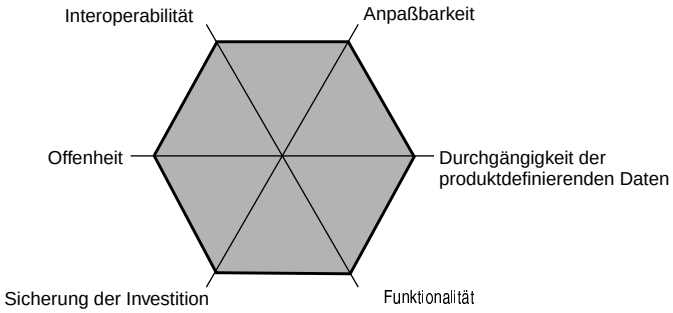

Abb. 8: Qualitative Einschätzung kooperierender CAx-Systeme

6. Projekt ANICA

Im Projekt ANICA (*An*alysis of *I*nterfaces of various *CA*D/CAM-Systems) werden die Schnittstellen zu den Systemkernen einiger CAx-Hersteller analysiert und ein Konzept für kooperierende CAx-Systeme in der Automobilindustrie wird entwickelt. Eine prototypische Implementierung ergänzt das Projekt.
Das Projekt wird zusammen mit deutschen Automobilfirmen und mit fünf CAx-Systemherstellern durchgeführt. Von der *Stiftung Rheinland-Pfalz für Innovation* erhält das Projekt finanzielle Unterstützung.

7. Ausblick

In den letzten Jahren haben sich CAx-Systeme immer mehr von Monolithen zu offen integrierten System entwickelt – zum Nutzen der Anwender. Offene Architekturen erlauben mehr Integration der Systeme in die Prozeßketten der Automobilindustrie, schnellere Erweiterung der Systeme um neue Funktionalitäten und bessere Anpassung des Funktionalitätsumfangs an die Bedürfnisse der Anwender.
Angestrebt wird ein Konzept für kooperierende CAx-Systeme, welches erlaubt, die Produktinformationsflüsse über den Gesamtprozeß zu verstärken und optimale CAx-Werkzeuge für jedes Anwendungsgebiet zu konfigurieren.

Literaturverzeichnis

(And93) B. Anderson u.a.: Software Architecture: The Next Step for Object Technology, in: OOPSLA 1993 Preceedings.

(BBK95) F. Bomarius, H. Biesiada, W. Krüger: SoftwareBus – Über die Entwicklung einer standardisierten Softwareschicht für verteilte objekt-orientierte Systeme in der Meß- und Versuchstechnik. VDI Berichte Nr. 1189, 1995.

(Dank95) Dankwort, C.W.: CAx-Systemarchitektur der Zukunft. VDI Berichte Nr. 1216, 1995.

(DKLPR94) W. Dankwort, P. Kellner, D. Leu, J. Petersen, W. Renz: Entwurf einer möglichen CAx-Architektur für Anwendungen in der Automobilindustrie. VDI Berichte Nr. 1134, 1994.

(MATRA95) MATRA Datavision: The CAS.CADE Software Factory. Technical Overview. Version 1.2. January 1995.

(MZ95) T. J. Mowbray, R. Zahavi: The Essential CORBA. Systems Integration Using Distributed Objects. John Wiley & Sons, Inc. 1995.

(RR90) D. Roller, H. Ruess: An Approach to an Open CAD System Architecture. In: F.-L. Krause and H. Jansen (Eds.): Advanced Geometric Modelling for Engineering Applications. Elsevier Science Publishers B.V. (North-Holland) 1990.

(Send95) Sendler, U.: CAD & Office Integration. OLE für Design und Modellierung – Eine neue Technologie für CA-Software. Springer-Verlag, Berlin Heidelberg 1995.