Integration von Gestaltung und Berechnung mittels CORBA

vorgelegt von
Diplom – Ingenieur
Gregor Haderer
aus Berlin

Vom Fachbereich 11 - Maschinenbau und Produktionstechnik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
– Dr.–Ing. –
genehmigte Dissertation

Promotionsausschuß:

Vorsitzender: Prof. Dr.-Ing F.-L. Krause
Berichter: Prof. Dr.-Ing. H. Mertens
Berichter: Prof. Dr.-Ing. M. Abramovici
Berichter: Dr.-Ing. O. Tegel

Tag der wissenschaftlichen Aussprache: 14. Juli 2000

Berlin 2000
D 83

Inhaltsverzeichnis

1 Einleitung **9**

1.1 Entwicklungstrends 10
1.2 Problemfelder 10
1.3 CORBA als Integrationsplattform 11
1.4 Gliederung der Arbeit 12

2 Gestaltung und Berechnung **15**

2.1 Definitionen und Ziele 15
2.1.1 Allgemeiner Produktentwicklungsprozeß 15
2.1.2 Modelle und Partialmodelle 16
2.1.3 Virtuelle Produktentstehung 17
2.2 Klassifizierung von Berechnungsmethoden 19
2.3 Modelltechnische Sicht 21
2.3.1 Geometriemodelle 21
2.3.2 Modelle für Berechnungen 22
2.3.3 Zusammenhänge zwischen Geometrie und Ersatzmodellen für die Berechnung 24
Allgemeine Verfahren 24
Spezielle Verfahren 25
2.3.4 Semantik in den Modellen 26
2.4 Prozeßtechnische Sicht 27
2.4.1 Gestaltungsprozeß 27
2.4.2 Einordnung von Berechnungen in den Produktentwicklungsprozeß 27
2.4.3 Arbeitsschritte beim Berechnen 29
2.4.4 Auszutauschende Daten 31
Modelldaten 31
Metadaten 31
2.4.5 Probleme bei der Integration 33
2.5 Systemtechnische Sicht 34
2.5.1 Mögliche Kopplungsformen 34
2.5.2 Zugriff auf die internen Modelle 35
Programmierschnittstellen 36

Integrationsgrad 36
Dateischnittstellen 37
Vernetzung der Systeme 37
2.5.3 Plattformabhängigkeit 38
2.6 Anwendersicht 39
2.6.1 Bereitstellung von Berechnungsmethoden 39
Firmeninterne Berechnungsmethoden 39
Berechnungsmethoden von externen Partnern 39
2.6.2 Qualifizierung der Bearbeiter 39
2.6.3 Qualität der Methoden 40
2.7 Zusammenfassung 40

3 Integrationsansätze 43
3.1 Integrierte Produktentwicklung 43
3.1.1 STEP - ein integriertes Produktmodell 43
Zielsetzung 43
Struktur 44
Anwendung 46
3.1.2 CAD–Referenzmodell 46
Anforderungen 47
Referenzarchitektur 47
Anwendung 49
3.2 Integration von Softwaresystemen 49
3.2.1 Überblick über Client-Server Umgebungen 50
3.2.2 CORBA - eine Systemarchitektur für verteilte Objekte . . 51
Struktur 52
Der *Object Request Broker (ORB)* 52
Komponenten 54
Container 55
Verteiltes Objektmodell 56
Granularität 56
Die *Common Object Services* 56
Common Facilities – Frameworks 58
Vorteile gegenüber anderen Ansätzen 58
3.2.3 JAVA als Programmiersprache 59
JAVA *Virtual Machine* 59
Portabler Programmcode 59
JAVA und CORBA 60
3.3 Integrierte Systeme 61
3.3.1 Produktdatenmanagementsysteme 61
3.3.2 Feature Modellier–System FEAMOS 61
3.3.3 Objektorientierter Integrationsprozessor OBIP 62
3.3.4 Projekt ANICA 63

Kommunikationssystem 63
Definition einer Zugriffsschnittstelle 63
Anwendung 63
Datenmanagement 64
3.3.5 Integrationsumgebung CORSICA 64
3.3.6 Konstruktionsanalyse- und Leitsystem KALEIT 65
3.3.7 Das System mfk 66
Produktmodell 66
Integration von Berechnungsmethoden 66
3.3.8 iViP–Projekt 67
Systemarchitektur 67
3.3.9 Virtuelle Berechnungskompetenzzentren 68
3.3.10 Vergleich der Systeme 69
3.4 Integrationsstrategien 71
3.5 Zusammenfassung 72

4 Anforderungen an das Integrationssystem 75
4.1 Anforderungen der Teilbereiche 75
Allgemeine Problembeschreibung 75
Modelle 75
Programme 76
Prozesse 76
Sicherheit 77
4.2 Anforderungsliste 80
4.3 Zusammenfassung 80

5 Integrationskonzept für CAx–Anwendungen auf der Basis von CORBA 81
5.1 Ausgangssituation 81
5.2 Stand der Standardisierung 82
5.3 Client–Server–Strukturen 83
5.3.1 Zentraler Datenserver 83
5.3.2 Applikationen als Server 84
5.3.3 Client–Server–Strukturen mit CORBA 84
5.3.4 Bewertung der Strukturen 85
5.4 Produktmodellstrukturen 86
5.4.1 Partialmodelle in den Applikationen 87
5.4.2 Semantische Verknüpfung von Objekten 88
5.4.3 Definition eines Strukturmodells 88
Baustruktur 89
Bezugselemente 89
Parameter 89
Bedingungen und Relationen 90

Prozeßinformationen 90
5.5 Kapselung der Applikationen 91
Einfache Kapselung der Applikationen 92
CAx-Framework 92
5.6 Systemkonzept für einfach gekapselte Applikationen 93
5.6.1 Systemstruktur 93
5.6.2 Anwendungsszenarien 93
5.7 Systemkonzept für ein CAx-Framework 94
5.7.1 Produktmodell und Partialmodelle 95
Referenzstruktur 95
Assoziationen 96
CAD–Komponenten 97
Berechnungskomponenten 97
Durchgängiges Prozeßmodell 98
5.7.2 Allgemeine und spezielle Dienste 98
Bereitstellung von Normen 98
Hilfesystem 99
Protokollierung und Dokumentation 100
5.7.3 Speicherung der Modelle 100
5.7.4 Systemstruktur 100
5.7.5 Anwendung 101
5.7.6 Implementierung 103
Arbeitsschritte 103
Erweiterungsmöglichkeiten 103
Realisierbarkeit 104
5.8 Vergleich der Systemkonzepte 104
Aufwand 104
Nutzen 104
Fazit 105

6 Implementierung eines Integrationssystems 107
6.1 Übersicht 107
6.1.1 Systemstruktur 107
6.1.2 SOM als CORBA–Laufzeitumgebung 109
6.2 Geometriemodellierer 110
6.2.1 Pro/ENGINEER 110
6.2.2 Schnittstellendefinition des Geometriemodellierers 110
6.2.3 Implementierungsdetails 113
Programmodule 113
Struktur der Objekte 113
6.3 Berechnungssystem 115
6.3.1 Schraubenberechnungsprogramm BOLT 115
6.3.2 Struktur der Schnittstelle 115

6.3.3 Implementierungsdetails 117
Programmodule 117
Funktionsweise 117
6.4 Modellmanager 118
6.5 Ablauf einer integrierten Berechnung 119
6.5.1 Interaktionsdiagramm 119
6.5.2 Anwendungsbeispiel 121
ModellManager 123
6.6 Variationsmöglichkeiten 128
6.7 Bewertung des Lösungsansatzes 129
6.7.1 Kopplung der Modelle 129
6.7.2 Kopplung der Programme 130
6.7.3 Kopplung der Prozesse 132
6.8 Zusammenfassung 133

7 Ausblick **135**
7.1 CORBA als Integrationsplattform 135
7.2 Vernetzte Berechnungs–Competence–Center 136
7.3 Komponentenbasierte CAx–Systeme 137

Glossar **139**

Abkürzungen

API Application Pogrammers Interface, Schnittstelle eines Softwaresystems für den programmgesteuerten Zugriff.

CAE Computer Aided Engineering, computerunterstütztes Berechnen.

CAx Oberbegriff für rechnerunterstützte Systeme.

CORBA Common Object Request Broker Architecture.

DCE Distributed Computing Environment.

DCOM Distributed Component Object Model.

DLL Dynamically Linked Library, eine zur Laufzeit gebundene Programmbibliothek.

FEM Finite Elemente Methode.

FTP File Transfer Protocoll, ein Protokoll für die Übertragung von Dateien über das Internet.

HTML Hypertext Markup Language.

IDL Interface Definition Language, eine systemneutrale Beschreibungssprache für objektorientierte Strukturen.

IIOP Internet Inter–ORB Protocol, ein Protokoll für die Verknüpfung mehrerer Object Request Broker.

ISO International Organisation for Standardisation.

MKS Mehrkörpersimulationsverfahren.

OMA Object Managing Architecture, die Systemarchtitektur von CORBA.

OMG Object Managing Group, ein Zusammenschluß nicht profitorientierter Firmen und Institutionen zur Förderung der Objekttechnologie. Entwickler von CORBA.

ORB Object Request Broker.

SOM System Object Model, eine CORBA–Implementierung der Firma IBM.

STEP Standard for Exchange of Product Model Data, ein Produktdatenmodell.

SQL Standard Query Language, eine Datenbankabfragesprache.

WWW World Wide Web.

1 Einleitung

Die Entwicklung von technischen Produkten ist mit einer Vielzahl von Tätigkeiten verbunden. Dabei nehmen die Tätigkeiten *Gestalten* und *Berechnen* besondere Rollen ein. Bei der Gestaltung wird aus einem Konzept oder Entwurf erstmals eine geometrisch stoffliche Gestalt festgelegt. Die Gestaltung ist damit eine der wesentlichen schöpferischen Tätigkeiten des Ingenieurs. Bei der Berechnung hingegen wird versucht, das Verhalten des realen oder geplanten Produkts vorherzusagen, indem in Form von Ersatzmodellen bestimmte Annahmen bezüglich der Eigenschaften des Produkts gemacht werden. Je nach Aufgabenstellung und Konkretisierungsgrad können dabei im Laufe des Entwicklungsprozesses sehr unterschiedliche Ersatzmodelle zum Einsatz kommen, die jeweils eine eigene Sicht auf das Produkt darstellen.

Gestaltung und Berechnung sind seit jeher zwei eng miteinander verknüpfte Tätigkeiten. Die Verknüpfungen bestehen einerseits in den gemeinsam genutzten Modellen und andererseits in den wechselseitigen Abhängigkeiten, die sich aus verschiedenen Auslegungs- und Nachweisrechnungen zwischen dem Gestaltmodell und den verwendeten Rechenmodellen ergeben. Bedingt durch die bei einer Berechnung erforderliche Abstraktion oder Reduktion auf ein Ersatzmodell, ist stets eine Transformation zwischen einem Gestaltmodell und einem zugehörigen Rechenmodell und umgekehrt erforderlich.

Für beide der oben genannten Tätigkeiten stehen inzwischen leistungsfähige Softwaresysteme für eine rechnerunterstützte Durchführung zur Verfügung. Allerdings sind die Entwicklungen bisher weitgehend isoliert abgelaufen, mit der Konsequenz, daß die entwickelten Systeme nicht miteinander operieren können. Für die Durchführung von in den Gestaltungsprozeß integrierten Berechnungen bedeutet dies, daß die oben genannten Transformationen durch den Bearbeiter vorgenommen werden müssen. Der dafür erforderliche Zeitaufwand, sowie die möglichen Fehlerquellen, bedingt durch Übertragungsfehler, Redundanzen und Inkonsistenzen sind für einen effizienten Entwicklungsprozeß nicht tragbar. Aus dieser Erkenntnis entsteht die Forderung, die bestehenden Programme in ein System zu integrieren, mit dem eine durchgängig rechnerunterstütze Durchführung von Gestaltung und Berechnung möglich ist.

1.1 Entwicklungstrends

Die derzeitigen Entwicklungen im Bereich der rechnerunterstützten Produktentwicklung verstärken den Bedarf nach einem Integrationsansatz, der es ermöglicht, Softwaresysteme für die Bearbeitung verschiedener Teilaufgaben zu einem Gesamtsystem zusammenzufügen. Wesentliche Veränderungen sind dabei sowohl in der Organisation der Entwicklungsprozesse als auch in den Möglichkeiten der Rechnerunterstützung festzustellen. Die Forderung nach einer Verkürzung der Entwicklungszeiten wird vielfach durch eine Parallelisierung der Arbeitsschritte in Form von *Simultaneous Engineering* erfüllt. Ein effizientes Arbeiten mit verteilten Modellen ist aber nur möglich, wenn den verschiedenen Entwicklungsarbeitsplätzen dieselbe Datenbasis zur Verfügung steht. Daher ist die Anwendung von *Simultaneous Engineering* häufig mit der Einführung von Produktdatenmanagementsystemen verbunden. Verschärft wird diese Problematik dadurch, daß verstärkt Entwicklungsleistungen an Zulieferer abgegeben werden.
Seitens der Rechnerunterstützung ergeben sich durch die in jüngster Zeit erfolgte nahezu flächendeckende Vernetzung der bestehenden Computersysteme völlig neue Formen der Kooperation. Zusammen mit entstehenden Standards für die Kommunikation zwischen verschiedenen Systemen sind Kopplungen vorstellbar, die über einen bloßen Dateiaustausch weit hinausgehen.
Versionswechsel bei den verwendeten Betriebssystemen führen, zusätzlich zu der bereits vorhandenen Vielfalt, zu einer noch größeren Zahl systemspezifischer Software. Dabei stellt die Wartung und Portierung bestehender Softwaresysteme einen erheblichen Aufwand dar.
Die Entwickler von CAD–Systemen versuchen entwicklungsprozeß- und fachdisziplinübergreifende Anwendungen zu schaffen. Vielfach werden zu einem CAD–System Produktdatenmanagmentsysteme mit einer hohen Interoperabilität und Module für die frühen Phasen und die nachgelagerten Phasen der Entwicklung angeboten. Die schnelle Entwicklung auf diesem Gebiet führt zu häufigen Releasewechseln und damit zu einem erheblichen Anpassungsaufwand für die bestehende Software.
Die Berechnungssoftware hingegen besteht häufig aus problemspezifischen Entwicklungen, die oft an eine vorhandene Systemumgebung angepaßt sind. Die Bibliothek von Berechnungsprogrammen kann somit einen beträchtlichen Teil des Wissens in einem Unternehmen darstellen.

1.2 Problemfelder

In bezug auf die Integration von Berechnungen in den Gestaltungsprozeß ergeben sich durch die genannten Entwicklungstrends einige Problemfelder. Insbesondere die Anbindung bestehender Berechnungssysteme an häufig wechselnde Versionen der aktuellen Geometriemodellierer bereitet Schwierigkeiten. Grün-

de für diese Schwierigkeiten sind hauptsächlich das Fehlen von standardisierten Zugriffsschnittstellen und die Vielfalt bestehender Berechnungssoftware. Die Entwicklung von Berechnungsprogrammen für spezielle Probleme führt oft zu schwer wartbaren Systemen, die an eine Rechnerplattform gebunden sind. Die fehlende Interoperabilität, sowohl zwischen verschiedenen CAD–Systemen als auch zwischen Berechnungssystemen und Geometriemodellierern, erschwert die Schaffung von durchgängig rechnerunterstützten Entwicklungsprozessen.
In der vorliegenden Arbeit wird ein Lösungsansatz für die Integration von Maschinenelementberechnungen in den Gestaltungsprozeß vorgestellt. Für die Berechnung von Maschinenelementen stehen in der Regel durchgehend parametrisierte Berechnungsmethoden zur Verfügung, sodaß eine Kopplung der Geometriemodelle mit den zugehörigen Rechenmodellen zunächst einfach erscheint. Maschinenelemente werden aber meist in einer individuell gestalteten Umgebung eingesetzt, deren Merkmale in Eingangsgrößen für die Berechnung übersetzt werden müssen. Dabei sind für die Berechnung oft Modellausschnitte oder spezielle Sichten auf die Umgebung erforderlich. Für eine Kopplung müssen daher neben Funktionen zur Identifizierung von Norm- und Standardteilen Methoden für die Generierung der Sichten verfügbar sein. Für eine Integration müssen die Modelle und deren Verknüpfungen, sowie die Prozeßschritte zum Wechseln der Modelle untersucht werden. Die Problematik wird anhand der Berechnung von Schraubenverbindungen verdeutlicht.

1.3 CORBA als Integrationsplattform

Ein umfassender Lösungsansatz für eine Integration von Gestaltung und Berechnung muß sich in die oben beschriebene, im Entstehen begriffene Systemlandschaft einfügen und zugleich offen sein für zu erwartende Entwicklungen. Der Lösungsansatz muß plattformübergreifend und netzwerkfähig sein und eine Anbindung bestehender Softwaresysteme, inbesondere der zahlreichen oft in Eigenentwicklung entstandenen Berechnungsprogramme, ermöglichen.
Im Bereich Informationstechnik wird derzeit mit CORBA ein Standard entwickelt, der es ermöglicht, verschiedene Softwaresysteme miteinander zu koppeln, indem die beteiligten Programme über neutrale, objektorientierte Schnittstellenbeschreibungen gekapselt und in das System integriert werden. CORBA steht für *Common Object Request Broker Architecure*. Für die Kommunikation zwischen den beteiligten Systemen und die Verwaltung der Objekte sind in CORBA zahlreiche Dienste in Form von standardisierten Methoden definiert. Damit ist CORBA ein Lösungsansatz, der viele der oben genannten Forderungen zu erfüllen verspricht.
Die Struktur von CORBA läßt dabei unterschiedliche Formen der Integration zu, die sich hinsichtlich des Realisierungsaufwands und der erzielbaren Vorteile unterscheiden. In dieser Arbeit wird untersucht, welche Integrationsformen

sinnvoll sind und inwieweit CORBA tatsächlich für die Integration von Gestaltungssoftware und Berechnungssoftware geeignet ist. Aus den Anforderungen an ein Integrationssystem und den sich durch CORBA ergebenden Möglichkeiten werden in der Arbeit zwei Integrationskonzepte abgeleitet und verglichen. Von den zwei Konzepten wird eins aufgegriffen und implementiert. Aus dem Vergleich mit den gestellten Anforderungen und den Erfahrungen aus der Implementierung ergibt sich, daß CORBA als Integrationsplattform grundsätzlich geeignet ist, da es einen sehr flexiblen Zugriff auf die angebundenen Systeme ermöglicht, daß aber dennoch ein gewisser Aufwand in den Bereichen Modellintegration und Prozeßintegration betrieben werden muß.

1.4 Gliederung der Arbeit

Im einführenden Kapitel 2 werden die Zusammenhänge der Arbeitsschritte *Gestalten* und *Berechnen* analysiert. Dabei wird auf die bestehenden Softwaresysteme, auf die verwendeten Modelle, die Einordnung in den Entwicklungsprozeß und auf die bei einer Integration auftretenden Probleme eingegangen.
In Kapitel 3 werden bestehende Lösungsansätze für verschiedene Ebenen der Integration vorgestellt. Dabei werden aktuelle Konzepte der integrierten Produktentwicklung, ausgewählte Integrationssysteme sowie Standards für die Integration von Softwaresystemen behandelt. In diesem Kapitel wird auch die Integrationsumgebung CORBA vorgestellt. Aus den beschriebenen Ansätzen werden allgemeine Integrationsstrategien abgeleitet.
Aus den Merkmalen der Arbeitsschritte *Gestalten* und *Berechnen* und den Lösungsansätzen werden in Kapitel 4 Anforderungen an ein Integrationssystem formuliert.
Die Lösung der Integrationsaufgabe bei Verwendung von CORBA als Kommunikationsplattform wird in Kapitel 5 beschrieben. In einem einleitenden Abschnitt werden die grundlegenden Merkmale einer CORBA–basierten Arbeitsumgebung erläutert. Nach einem Überblick über den Stand der Standardisierung im Bereich verteilte CAx–Umgebungen werden ausführlich die verschiedenen Strukturen und Komponenten eines CORBA-basierten Integrationssystems vorgestellt. Abschließend werden zwei realisierbare Systemkonzepte für eine einfache Kopplung und für einen integrierten Entwicklungsarbeitsplatz präsentiert.
Eine Implementierung der in Kapitel 5 entwickelten einfachen Kopplung wird in Kapitel 6 beschrieben. Das implementierte System wird in bezug auf die in Kapitel 4 gestellten Anforderungen bewertet.
Zusammenfassende Betrachtungen über die durchgeführten Arbeiten, die praktische Anwendbarkeit der Ergebnisse und die zu erwartenden Entwicklungen schließen im Kapitel 7 die Arbeit ab.
Bild 1.1 zeigt eine Übersicht über die Gliederung der Arbeit.

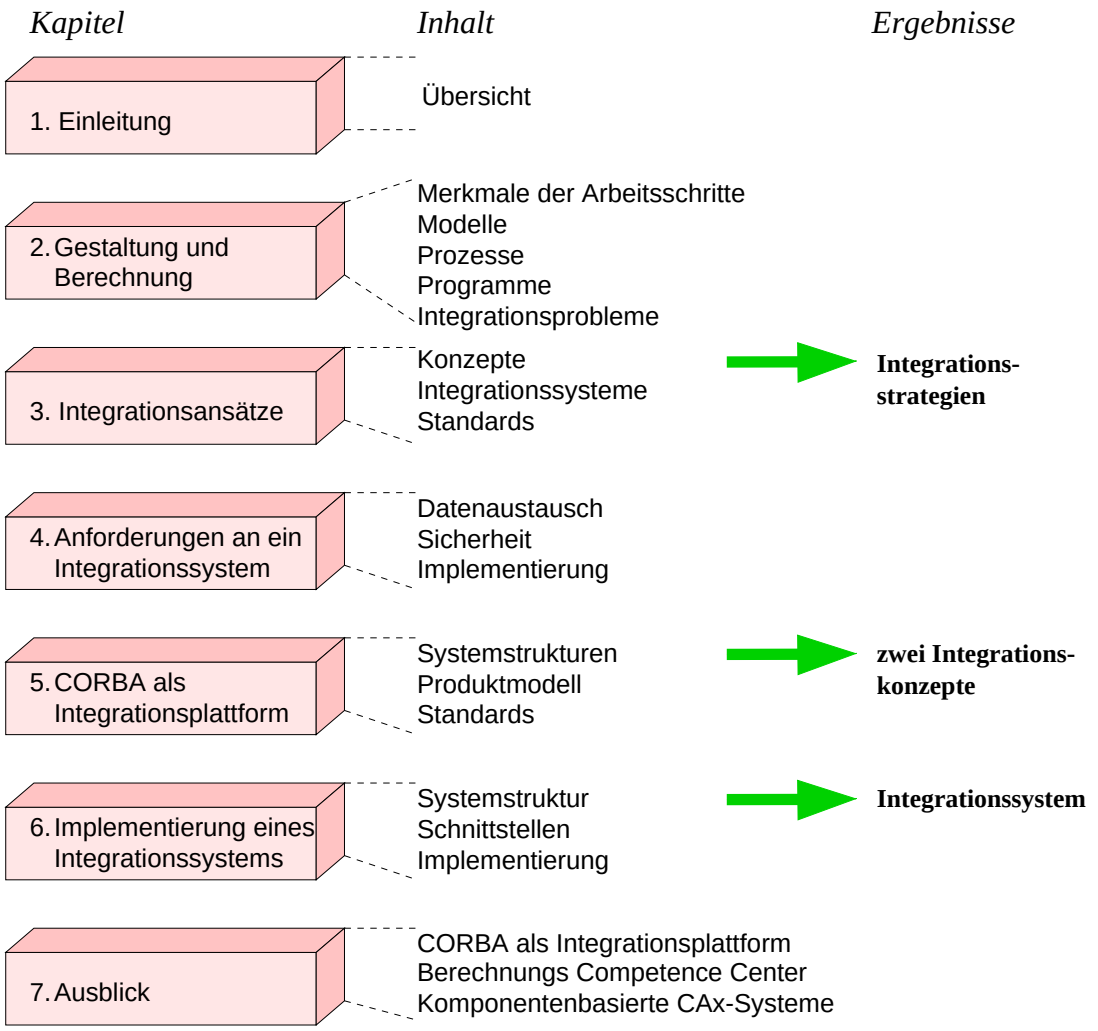


Bild 1.1: Gliederung der Arbeit

2 Gestaltung und Berechnung

In diesem Kapitel werden Merkmale von Maschinenelementberechnungen, ihre Ziele und die Eingliederung in den Produktentwicklungsprozeß erläutert. Dabei wird insbesondere auf die modelltechnischen, die prozeßtechnischen und programmtechnischen Probleme im Hinblick auf eine rechnertechnische Integration von Berechnungen in den Gestaltungsprozeß eingegangen.

2.1 Definitionen und Ziele

2.1.1 Allgemeiner Produktentwicklungsprozeß

Das generelle Vorgehen beim Entwickeln und Konstruieren wird in der VDI Richtlinie 2221 [VDI93] angegeben. Das in der Richtlinie vorgeschlagene Schema teilt den vollständigen Entwicklungsprozeß in die vier Phasen: *Klären der Aufgabe*, *Konzeptphase*, *Entwurfsphase* und *Ausarbeitungsphase* (vgl. Bild 2.1). Das *Klären der Aufgabe* dient der Beschaffung von Informationen über Anforderungen, die an die Lösung gestellt werden, sowie an bestehende Bedingungen und deren Bedeutung. Das Ergebnis dieses Arbeitsschrittes, die *Anforderungsliste*, ist Arbeitsgrundlage und Informationsquelle für alle nachfolgenden Arbeitsschritte.
Beim *Konzipieren* werden durch Abstraktion der Anforderungen die wesentlichen Probleme der Aufgabenstellung herausgearbeitet. Durch Aufstellen einer Funktionsstruktur und durch Suche nach geeigneten Wirkprinzipien und deren Kombination in einer Wirkstruktur wird eine *prinzipielle Lösung* erarbeitet, in der die wesentlichen Komponenten zur Realisierung des Produkts erkennbar sind. Dabei kann die prinzipielle Lösung auch schon eine geometrische Festlegung enthalten [PB97].
Die gestalterische Festlegung der Lösung erfolgt in der *Entwurfsphase*. Ausgehend von der prinzipiellen Lösung wird in diesem Arbeitsschritt die Baustruktur nach technischen und wirtschaftlichen Gesichtspunkten erarbeitet. Das Ergebnis dieses Schrittes, der *Gesamtentwurf*, enthält alle Informationen, um eine Kontrolle der Funktion, der räumlichen Verträglichkeit, der Haltbarkeit und der angestrebten Kosten zu ermöglichen. Die geometrische Beschaffenheit des Produkts wird maßgeblich in diesem Arbeitsschritt festgelegt.
Im abschließenden Arbeitsschritt, der *Ausarbeitung*, erfolgt die endgültige Festle-

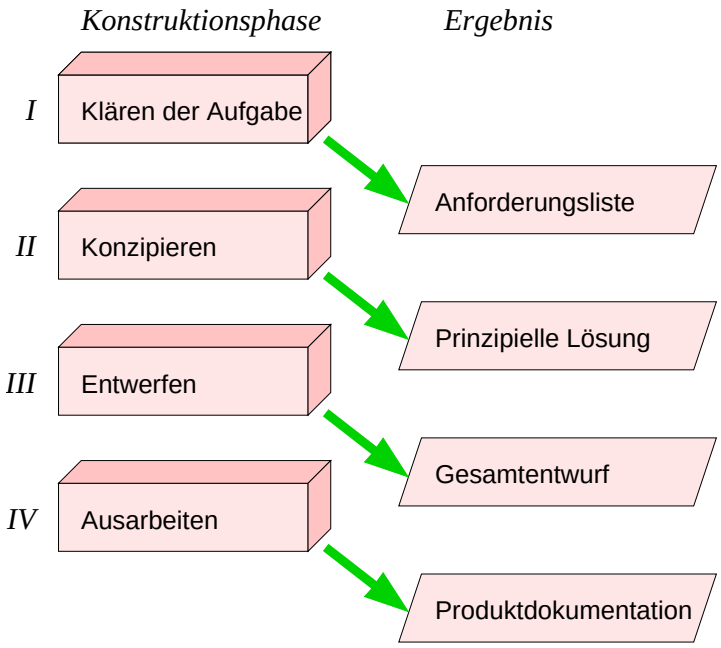


Bild 2.1: Phasen und Arbeitsergebnisse des Konstruktionsprozesses (in Anlehnung an [VDI93])

gung aller Details, die für die Herstellung des Produktes erforderlich sind. Dazu gehören die detaillierte Festlegung der Form, der Werkstoffe, der Oberfläche und des Fertigungsprozesses. Ergebnis dieses Arbeitsschrittes ist die *Produktdokumentation*.
Je nach Aufgabenstellung können die einzelnen Ergebnisse mehr oder weniger ausführlich ausgeführt sein. Untersuchungen in der Praxis haben gezeigt, daß das Schema nur sehr selten mit allen angegeben Schritten durchlaufen wird, daß sich aber die meisten Vorgehensweisen in das Schema einordnen lassen [Rüc97].
Die aktuelle Forschung zielt darauf ab, die Entwicklungstätigkeiten in Form einer *integrierten Produktentwicklung* durchzuführen. Integrierte Produktentwicklung kann als zielorientierte Kombination organisatorischer, methodischer und technischer Maßnahmen und Hilfsmittel definiert werden [Mee98]. In zahlreichen Forschungsvorhaben wird zur Zeit versucht, den allgemeinen Entwicklungsprozeß durchgängig rechnerunterstützt durchzuführen. Eine Auswahl bestehender Lösungsansätze ist in Kapitel 3 beschrieben.

2.1.2 Modelle und Partialmodelle

Das Vorgehen beim Entwickeln und Konstruieren wird durch verschiedene *Modelle* unterstützt. Die Modelle können dabei Produkte oder Prozesse oder beides beschreiben, sie können eine vollständige Beschreibung oder nur einen Ausschnitt

enthalten und können als Denkmodelle des Bearbeiters existieren oder als rechnerinterne Modelle in Softwaresystemen bestehen. Tabelle 2.1 gibt einen Überblick über allgemeine und in die in der vorliegenden Arbeit verwendeten Modellbegriffe.
Im Bereich der rechnerunterstützten Produktentwicklung kommt den integrierten Modellen eine besondere Bedeutung zu. Definitionsgemäß sollen sie in der Lage sein, sämtliche im Produktlebenszyklus anfallenden Daten abzubilden. Sie werden damit zum zentralen Integrationswerkzeug, da sämtliche Softwaresysteme das integrierte Produktmodell als Datenbasis verwenden. Das Modell muß so festgelegt und beschrieben werden, daß es die Funktionen der Produktdatenverarbeitung unterstützt. In [GAP93] werden folgende zu unterstützende Funktionen genannt:

- der Produktdatenaustausch
- die Produktdatenspeicherung
- die Produktdatenarchivierung und
- die Produktdatentransformation.

2.1.3 Virtuelle Produktentstehung

Von Industrie und Forschungseinrichtungen wird derzeit die *virtuelle Produktentstehung* als Ziel formuliert. Die zentrale Vision der virtuellen Produktentstehung besteht darin, eine vollkommen neuartige Grundlage für den Produktentstehungsprozeß von der initialen Produktplanung über die Produktkonstruktion und –erprobung bis hin zum Fertigungsanlauf zu realisieren. "Virtuell" bezeichnet hierbei die über alle Phasen durchgängig digitale Produktentstehung mittels dreidimensionaler Modelle über Unternehmensgrenzen und Standorte hinweg. Unter Einbeziehung innovativer Technologien wie Multimedia und Virtual Reality (VR), sowie zukunftsorientierter Organisations–, Kommunikations– und Informationssysteme entsteht das "virtuelle Produkt" auf der Basis eines umfassenden 3D–Modells, zur Simulation aller Eigenschaften, Funktionen und Entstehungsphasen, visualisiert mit Hilfe von VR–Techniken, getestet und erprobt auf dem virtuellen Prüfstand [KTA98]. Das Konzept der durchgängig digitalen Produktenstehung erfordert neue Konzepte für die Integration der am Entwicklungsprozeß beteiligten Werkzeuge und umfassende Ansätze zur Steuerung des verteilten Prozesses. Ein großangelegtes Forschungsvorhaben hat sich die Schaffung einer Arbeitsumgebung für die virtuelle Produktentstehung zum Ziel gesetzt. Einzelheiten sind in Abschnitt 3.3.8 beschrieben.

Begriff	Definition
Modell (allg.)	Abstrakte Abbildung eines Ausschnittes der realen Welt mit dem Ziel, von Eigenschaften des Modells auf Eigenschaften des realen Objekts zu schließen [VDI93] [Rot82]. Im Kontext von Softwareprogrammen: als Datenmodell strukturelle Datenmenge zur rechnerinternen Repräsentation eines Weltausschnittes [Abe95].
Ersatzmodell	Abstraktion eines realen oder geplanten Produkts durch Reduktion auf eine Auswahl von Eigenschaften. Ein Ersatzmodell wird verwendet, um das Verhalten des Produkts durch Vereinfachung berechenbar und damit vorhersagbar zu machen.
Produkt-informations-modell	(Auch: *konzeptionelles Modell*) Definiert die möglichen Bestandteile und denkbaren Relationen zwischen den Bestandteilen für eine Abbildung eines Modells im Rechner [Boo94].
Produktmodell	Die Einheit von Produktinformationsmodell und aller zu einem Produkt gehörenden Daten [Abe95].
Integriertes Produkt-modell	Produktmodell für die Abbildung von Daten aus allen Produktlebensphasen in einem einheitlichen Datenmodell [GAP93]. Ein kontinuierliches Fortschreiben der Daten soll die informationsverlustbehaftete Modelltransformation verhindern. In dem Modell werden alle Daten abgebildet, die sich nicht aus bereits bestehenden Daten ableiten lassen [Vel99].
Partialmodell	Kennzeichnet eine definierte endliche Informationsmenge von Produktmerkmalen als Teil eines Produktmodells [Abe95] [Gra99].
Geometriemodell	Hier: Rechnerinterne Darstellung der Geometriedaten in einem Geometriemodellierer (CAD–System).
Rechenmodell	Hier: Rechnerinterne Darstellung der Daten eines Berechnungsprogramms.
Strukturmodell	Hier: Produktinformationsmodell für die Integration von Gestaltung und Berechnung. Enthält Ausschnitte aus einem Geometriemodell und einem Rechenmodell und die für die Verknüpfung der Modelle erforderlichen Informationen.

Tabelle 2.1: Definitionen der wichtigsten Modellbegriffe

2.2 Klassifizierung von Berechnungsmethoden

In diesem Abschnitt werden bestehende Klassifizierungen für Berechnungen vorgestellt. Die gefundenen Merkmale werden in den nachfolgenden Abschnitten aufgegriffen und in einer der drei Sichten, modelltechnische Sicht, prozeßtechnische Sicht oder systemtechnische Sicht detaillierter behandelt.
In der Literatur existieren verschiedene Klassifizerungsansätze für Berechnungsmethoden. Die VDI Richtlinie 2211 [VDI80] nennt die Genauigkeit, die Komplexität und die Häufigkeit der Benutzung als Unterscheidungsmerkmale. Die in der Richtlinie gezogenen Konsequenzen sind bezogen auf die heutige Rechnertechnik nicht mehr zeitgemäß, weshalb die Richtlinie inzwischen zurückgezogen wurde.
Von Mertens [Mer98] wurde ein Klassifizierungsschema entwickelt, in dem Berechnungsmethoden nach Aufwand und Aussagegüte in jeweils drei Klassen eingeordnet werden können (vergleiche Bild 2.2). Dabei bezeichnet die Klasse A Methoden, die eine hohe Aussagegüte und einen hohen Berechnungsaufwand oder Modellierungsaufwand haben und in der Regel nur von einem Experten bearbeitet werden können. Verfahren der Klasse B erfordern hingegen einen gegenüber A-Methoden verminderten Aufwand bezüglich Aufwand und Laufzeit, benötigen aber immer noch einen auf dieses Themengebiet spezialisierten Bearbeiter für die Modellierung und die Interpretation der Ergebnisse. Berechnungsverfahren, die als allgemein bekannt vorausgesetzt werden können oder die in bekannten Richtlinien oder anderen Quellen veröffentlicht wurden und bezüglich Laufzeit und Aussagegüte gering eingestuft werden können, sind in die Klasse C einzuordnen. Für die verschiedenen Verfahren existieren auch Zwischenstufen, wenn beispielsweise Methoden mit kurzer Laufzeit eine hohe Aussagegüte besitzen. Das Bestreben von Berechnungsabteilungen und Forschungseinrichtungen geht dahin, möglichst viele der bisher den Experten vorbehaltenen Verfahren durch entsprechende Anpassungen auf eine B- oder C–Ebene zu bringen und damit weiteren Anwendern zugänglich zu machen. Die Überführung in C-Methoden bedeutet auch, daß die Randbedingungen so klar definiert sein müssen, daß für die Modellbildung kein Expertenwissen mehr erforderlich sein darf. Dies erhöht zugleich die Chance, Teile der Modellbildung zu automatisieren.
Die Ingenieurdisziplinen *Mechanik*, *Fluidik*, *Thermodynamik*, *Wärmeübertragung*, *Optik*, *Elektrotechnik* und *Akustik* werden von Koller in [Kol98] für die Unterscheidung von Berechnungsverfahren herangezogen. Koller unterscheidet weiter in *allgemeine*, das heißt, für alle Produkte geltende Verfahren, die sich meist aus physikalischen Gesetzen ergeben, und in *spezielle*, auf ein Produkt oder eine Produktgruppe abgestimmte Verfahren, die sich aus einer produktspezifischen Anpassung der allgemeinen Verfahren ergeben.
Eine Klassifizierung, die sich an dem Ziel und den Ergebnissen einer Berechnung orientiert, wird von Galwelat vorgenommen. Dort werden Auslegungsrechnungen oder Dimensionierungen, Nachweisrechnungen, Optimierungen und Vergleichsrechnungen unterschieden [Gal80].

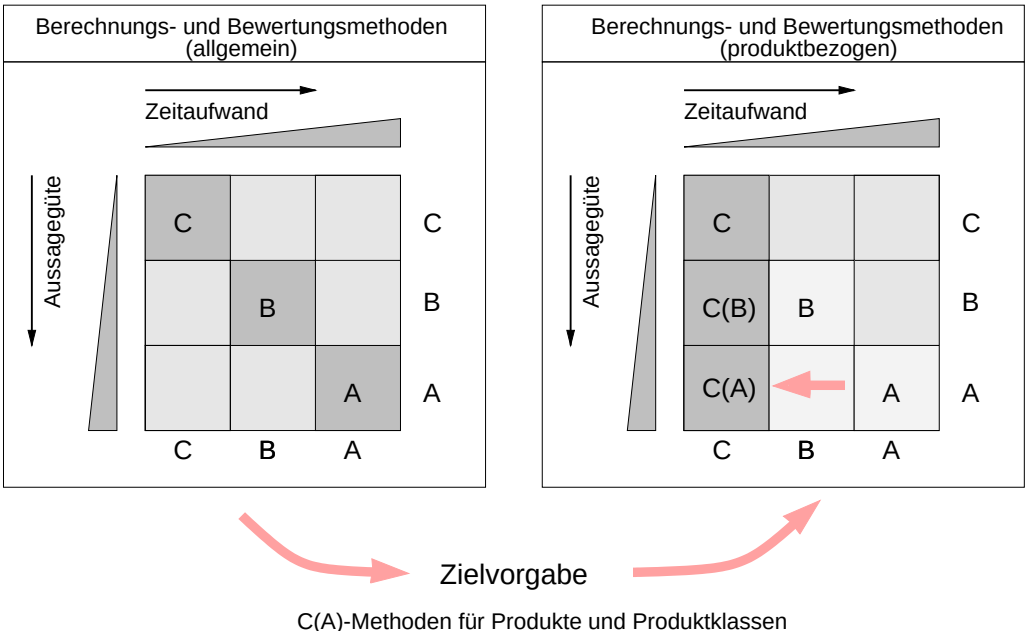


Bild 2.2: Klassifizierung von Methoden nach Zeitaufwand und Aussagegüte [MW99]

Im Hinblick auf eine Integration in den rechnerunterstützten Gestaltungsprozeß ist auch die *Integrierbarkeit* ein Klassifizierungsmerkmal. In diesem Zusammenhang ist *Methode* als implementiertes Programm auf einem Rechner zu verstehen. In [Wöl98] werden Methoden, deren Quellcode und deren Ein– und Ausgabeparameter bekannt und programmatisch zugänglich sind, als *direkt integrierbar*, und Methoden, die nur als übersetztes Programm mit nicht veränderbarem Ablauf, aber bekannter Parameterstruktur vorliegen als *indirekt integrierbar* bezeichnet. Als *lose integrierbar* werden Methoden bezeichnet, die in Gestalt von komplexen CAE–Systemen eine detaillierte Modelldefinition und Parameterermittlung für jede einzelne Berechnung erfordern.

Eine Klassifizerung, in der auch eine grobe Zuordnung zu den Phasen des Entwicklungsprozesses vorgenommen wird, ist in [Löf97] beschrieben. Dort werden parameterorientierte Dimensionierungs- und Auslegungsrechnungen der Konzeptphase, allgemeine Modelle der technischen Mechanik (MTM), beispielsweise Balken oder Platten, der Entwurfsphase und detaillierte FE–Berechnungen der Ausarbeitungsphase zugeordnet.

Eine Übersicht über die genannten Merkmale ist in Bild 2.3 dargestellt.

Bild 2.3: Merkmale von Berechnungen

2.3 Modelltechnische Sicht

Um Produktentwicklung rechnerunterstützt betreiben zu können, wurden für die Objekte, die Gegenstand der Entwicklung sind, Produktinformationsmodelle geschaffen, in denen die Eigenschaften der Objekte abgebildet werden können. In diesem Abschnitt werden die Eigenschaften der in den bestehenden Programmen vorhandenen Modelle beschrieben und die Probleme bei einer Integration zusammengestellt.
Integrierte Modelle kommen zur Zeit nur in Forschungssystemen zum Einsatz (vgl. Kapitel 3) und sind für die bestehenden Programme noch nicht einsetzbar. Allerdings zielen die aktuellen Entwicklungen im Bereich CAD–Systeme darauf ab, die Programme zu prozeßübergreifenden Universalwerkzeugen zu machen und damit die internen Modelle in Richtung eines integrierten Produktmodells weiterzuentwickeln.

2.3.1 Geometriemodelle

Der wesentliche Gegenstand der Gestaltung ist die Geometrie. Insofern sind die *Geometriemodelle* die Arbeitsmodelle bei der Gestaltung. Für die rechnerunter-

stützte Durchführung dieses Arbeitsschrittes stehen mit den modernen CAD–Systemen leistungsfähige Werkzeuge zur Verfügung, die umfangreiche interne Darstellungen des Modellierungsgegenstandes enthalten.
Grundsätzlich können Geometriemodelle in B–Rep und CSG–Modelle unterschieden werden. In B–Rep Modellen (*Boundary Representation*) wird ein geometrisches Objekt durch seine Umrandungsflächen beschrieben. Diese Modellierungsart wird häufig für Freiformflächen verwendet.Im Gegensatz dazu werden CSG–Modelle (*Constructive Solid Geometrie*) beschrieben, indem von einem Grundkörper durch boolsche Volumenoperationen Formelemente entfernt oder hinzugefügt werden. Im Vergleich zu B–Rep Modellen bieten CSG–Modelle die Möglichkeit, die Entwicklungshistorie des Modells zu speichern und so auch Modifikationen am Modell rückgängig zu machen.
Stand der Technik sind derzeit parametrische 3-D Modelle, die je nach Aufgabenstellung als Flächenmodelle oder Volumenmodelle oder auch parallel in den Systemen verwaltet werden. Bei diesen Modellen werden die geometrischen Eigenschaften des abgebildeten Objekts durch benannte Variablen, die Parameter, beschrieben. Parameter können nachträglich verändert und zueinander in Beziehung gesetzt werden. In Verbindung mit Features, die in diesem Kontext als Konstruktionselemente zu verstehen sind, lassen sich die Modelle schnell und kostengünstig im Rechner erzeugen [Gra99]. Über die Benutzungsschnittstelle und zum Teil auch über die Programmierschnittstelle hat man Zugriff auf die rechnerinterne Darstellung der geometrischen Objekte. Leistungsfähige Programme wie Pro/ENGINEER oder IDEAS bieten dabei sowohl Zugriffsmöglichkeiten auf rein geometrische Informationen, wie auf Punkte, Kanten, Flächen, und deren Eigenschaften als auch auf die logischen Verknüpfungen von Konstruktionselementen und Bauteilen [Par97].
Die aktuellen Entwicklungen zielen darauf ab, sowohl die frühen Phasen als auch die der Gestaltung nachgelagerten Arbeitsschritte durch zusätzlich im Modell abgelegte Informationen zu unterstützen. Ziel ist dabei immer, eine durchgängige Rechnerunterstützung des Prozesses ohne Systemwechsel zu ermöglichen. Parallel dazu werden die Systeme zunehmend für die Anwendung im Simultaneous Engineering weiterentwickelt.

2.3.2 Modelle für Berechnungen

Bauteilberechnungen liegt in der Regel ein physikalisches Ersatzmodell zugrunde. Im Bereich Maschinenbau bestehen diese Ersatzmodelle aus geometrischen Komponenten, denen in Bezug auf die zu untersuchende Eigenschaft ein bestimmtes Verhalten unterstellt wird. Je nach Verfahren und Stand der Forschung können diese Komponenten komplette Baugruppen, einzelne Bauteile, Zonen eines Bauteils oder die Elemente eines Finite-Elemente Netzes sein. Die Verknüpfungen zwischen den Komponenten werden mit Hilfe von Bedingungen formuliert. Das Verhalten der Komponenten wird durch Formeln oder Algorithmen beschrieben.

Die Eigenschaften der einzelnen Komponenten können durch Parameter beschrieben werden.

Mit der oben gefundenen Aufteilung in geometrische Komponenten, Bedingungen und Parametern lassen sich die numerischen Verfahren wie FEM (Finite Elemente Methode) oder MKS (Mehrkörper Simulation) gut beschreiben: die Finiten Elemente und die starren Körper bilden die geometrischen Komponenten, die über Rand- und Übergangsbedingungen miteinander verbunden sind. Die Eigenschaften der einzelnen Komponenten werden durch Parameter und Formeln beschrieben. Der Übergang vom realen Produkt zum Rechenmodell erfolgt, indem das System in viele gleichartige Elemente mit einfacher Struktur zerlegt wird. Ausgehend von einer vorhandenen geometrischen Struktur ist eine Zerlegung fast immer möglich und die physikalischen Zusammenhänge sind bekannt.

Rechenverfahren, die nur für eine ganz bestimmte Gruppe von Maschinenelementen geeignet sind, lassen sich bezüglich der Eingabeparameter mit Hilfe der zugrundeliegenden Baustruktur gliedern. Dabei ist der Gültigkeitsbereich entweder in Form der verwendeten Normteile oder in Form von Gültigkeitsbereichen einzelner Parameter des Ersatzmodells angegeben. Für die Ermittlung der Parameter ist oft nicht die gesamte Geometrie erforderlich, sondern eine Auswahl von charakteristischen Bezugselementen. Alle Eingabeparameter lassen sich in der Regel einer Baugruppe, einem Bauteil oder einem geometrischen Objekt eines Bauteils (beispielsweise einer Fläche oder einer Kante) zuordnen. Darüberhinaus werden diese Rechenverfahren oft in mehrere *Rechenschritte* unterteilt.

Die Parameter selber lassen sich in weitere Merkmalsgruppen einteilen. Eine mögliche Unterteilung ist die Zusammenstellung der Produktmerkmale, wie sie von Pahl und Beitz in [PB97] vorgenommen wurde. In Bild 2.4 sind die Merkmalsgruppen zusammengestellt, die für die Berechnung von Schraubenverbindungen nach der VDI–Richtlinie 2230 [VDI86] benötigt werden. Man erkennt, daß zahlreiche Parameter aus anderen Bereichen wie Fertigung oder Montage Einfluß auf das Rechenergebnis haben. Beispiele hierfür sind der Einfluß der bei der Fertigung erzielbaren Oberflächenrauhigkeiten auf den Setzbetrag einer Schraubenverbindung oder die Vorgabe eines bestimmten Werkzeuges bei der Montage.

Ein großer Teil der Parameter läßt sich bei Verwendung von Norm– und Standardteilen aus den entsprechenden Norm– und Regelwerken ermitteln. Oft sind für genormte Komponenten Tabellenwerke angegeben und für individuelle Strukturen die entsprechenden Formeln als zusätzlicher Rechenschritt angegeben.

Typisch für die spezifischen Rechenverfahren ist auch, daß es eine Reihe von Parametern gibt, für die nur in diesem Kontext gültige Tabellen oder Näherungsformeln existieren, die von den allgemein bekannten Quellen abweichen. Beispielsweise existieren spezielle Tabellen mit zulässigen Flächenpressungen bei Schraubenverbindungen [VDI86].

Tabelle 2.2 zeigt einen Überblick über die Quellen von in Berechnungen verwendeten Parameter und ihrer typischen Eigenschaften.

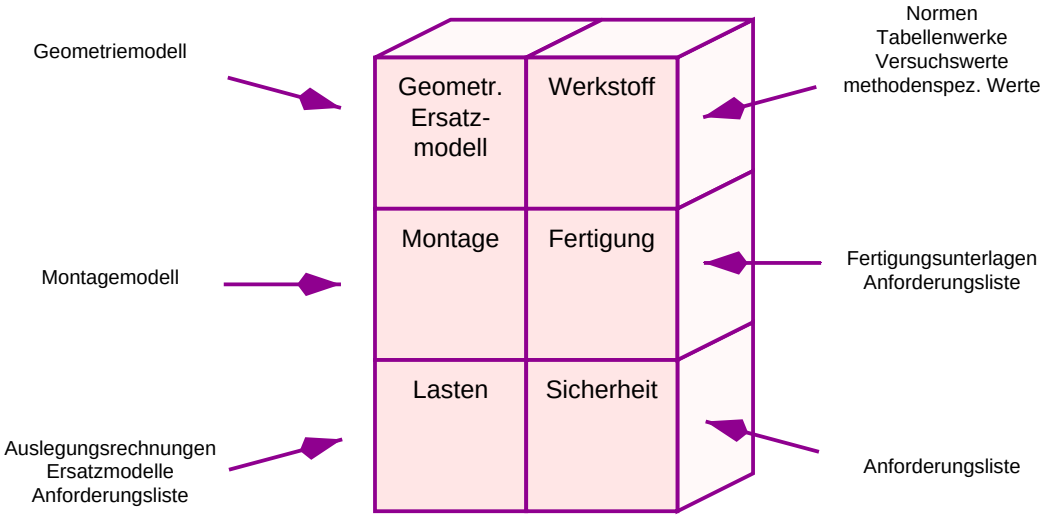


Bild 2.4: Parameter und deren Quellen bei Bauteilberechnungen

2.3.3 Zusammenhänge zwischen Geometrie und Ersatzmodellen für die Berechnung

Inhaltlich sind die geometrischen Daten einer Berechnung nur ein Teil der benötigten Daten (vergleiche Bild 2.4). Je nachdem, ob es sich bei dem Berechnungsverfahren um ein allgemeines (zum Beispiel ein numerisches) Verfahren oder um ein spezielles, auf ein Produkt abgestimmtes Verfahren handelt, sind die für eine Berechnung benötigten Geometriedaten unterschiedlich.

Allgemeine Verfahren

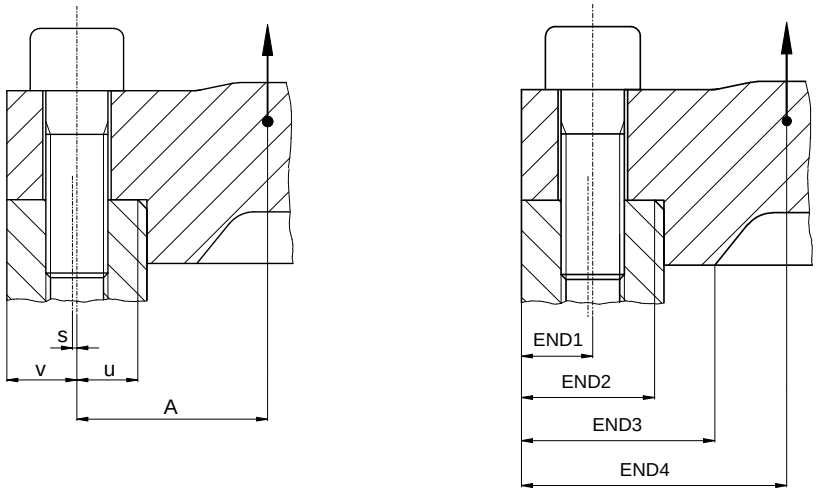
Bei der Verwendung allgemeiner Verfahren, wie MKS oder FEM, wird in der Regel ein detailliertes Rechenmodell aus einem gegebenen Geometriemodell abgeleitet, indem das Geometriemodell in kleinere, gleichartige (finite) Elemente zerlegt wird. Eine Berechnung mit einem der Verfahren setzt demnach voraus, daß bereits ein Geometriemodell in Form eines Oberflächenmodells oder eines Volumenmodells existiert. Parameter oder Featureinformationen werden nicht benötigt. Beim Übergang von einem detaillierten Geometriemodell zu einem Rechenmodell kann es erforderlich sein, Geometriedetails, die das Rechemodell unnötig verkomplizieren würden, vor der Umwandlung aus dem Modell zu entfernen. Das Entfernen der für eine Berechnung nicht relevanten Konstruktionselemente wird auch als *defeatureing* bezeichnet. Zur Zeit muß diese Aufgabe noch durch den Bearbeiter erledigt werden [Zie99], da die Entscheidung, ob ein Konstruktionsdetail für die Berechnung relevant ist oder nicht, sehr komplex ist.

Typ	Beispiel	Eigenschaften
Normen	Schraubendurchmesser Werkstoffdaten	konstant, leicht zu identifizieren
Andere Berechnungen	Lasten Betriebskräfte	veränderlich, hängen ab von Randbedingungen
Fertigungsunterlagen	Oberflächenrauhigk.	unterliegen Streuungen
Richtlinien	Reibwerte Anziehfaktoren	begrenzte Gültigkeit, haben oft Ermessensspielraum
Benutzereingaben	Lasteinleitungsfaktor	nicht reproduzierbar
Anforderungsliste	Lebensdauer	konstant

Tabelle 2.2: Typische Quellen von Parametern und deren Eigenschaften

Spezielle Verfahren

Bei den speziell auf ein Produkt oder eine Produktgruppe abgestimmten Verfahren können die Modelle für die Berechnung sowohl aus einer gegebenen Geometrie abgeleitet werden als auch umgekehrt. Dabei können beim Übergang von der Geometrie zum Ersatzmodell sowohl Informationen vernachlässigt werden, wenn bei der Berechnung Details wie beispielsweise Ausrundungsradien oder Fasen nicht berücksichtigt werden als auch zusätzliche Informationen erforderlich sein, die üblicherweise nicht in einer Geometriedarstellung enthalten sind, Beispiele hierfür sind Gewindeabmessungen, oder die Detailgeometrie eines Freistichs. In der Regel wird eine gegebene Geometrie nicht direkt in einer Berechnung verwendet, sondern als geometrisches Ersatzmodell, das aus dem Original mit Hilfe von Abbildungsvorschriften und zusätzlichen Informationen abgeleitet wurde. Die Ableitung des Ersatzmodells ist einfach, wenn nur Norm– und Standardteile oder parametrisierte Konstruktionsfeatures verwendet werden, da dann die Merkmale für beide Modelle aus den entsprechenden Bibliotheken entnommen werden können. In vielen Fällen ist es allerdings nicht möglich, ein Produkt nur aus Norm– und Standardteilen aufzubauen, oder es zeigt sich, daß verschiedene Berechnungsverfahren eine unterschiedliche Parametrisierung erfordern würden. Beispielsweise kann eine Schraubenverbindung durch das Programm BOLT [BOL94] mit unterschiedlichen Modellannahmen berechnet werden. Bild 2.5 zeigt auf der linken Seite die Parametrisierung, wie sie für eine Berechnung als zylindrische Einschraubenverbindung erforderlich wäre und auf der rechten Seite die Parametrisierung, wenn die verspannten Teile als balkenartige Struktur betrachtet würden.



VDI 2230 Balkenmodell

Bild 2.5: Unterschiedliche Parametrisierungen einer Schraubenverbindung im Programm BOLT in Abhängigkeit vom Berechnungsverfahren ([BOL94])

2.3.4 Semantik in den Modellen

Semantik wird in [Wah86] als Lehre von der Bedeutung der Wörter oder als Lehre von den Zeichen und Symbolen als Denkinhalte einer Wissenschaft definiert. Im Bereich Informationstechnik wird der Begriff *semantisches Datenmodell* für Datenmodelle verwendet, in denen alle Strukturen und zulässigen Operationen bestimmten Typen von Informationen aus der realen Welt entsprechen [Stü90]. Demzufolge können Datenmodelle eine unterschiedlich komplexe Semantik besitzen, je nachdem wieviele der Eigenschaften der realen Welt in das Datenmodell übernommen wurden.

Von den CAD–Systemen werden sehr komplexe Datenmodelle gehalten und über die Programmierschnittstelle zugänglich gemacht. In den Systemen sind Dinge wie Bauteile, Flächen, Kanten oder Punkte und Methoden für die Analyse dieser Dinge und den Relationen zwischen ihnen definiert [Par97]. Darüberhinaus wird mit der Featuremodellierung versucht, den Objekten eine über die reine Geometrieinformation hinausgehende Bedeutung zuzuordnen [Rie94] [Löf97], und diese Bedeutung auch für andere Aufgaben, wie die Erstellung von NC-Programmen oder die Materialwirtschaft zu nutzen. Detailbetrachtungen zeigen, daß, obwohl mehrere Systeme eine ähnliche Funktionalität aufweisen, die Analysemethoden und damit die Semantik in den Modellen Unterschiede aufweisen [AJSK98]. Beispielsweise kann eine Kante beschrieben sein durch vier Koordinatenwerte oder durch zwei Endpunkte, die jeweils aus zwei Koordinatenwerten bestehen.

Der komplexen Semantik der Geometriemodelle steht eine relativ einfache bei

den Rechenmodellen gegenüber. Die meisten Programme für die Berechnung von Maschinenelementen verwenden für die Darstellung der Modelle benannte Parameter [BOL94] [HEX95] [Dam96]. Die Bedeutung wird einem Parameter durch den Benutzer zugeordnet. Verknüpfungen zwischen Parametern oder die Zugehörigkeit zu einem Bauteil können aus den Systemen nicht ermittelt werden. Die unterschiedliche Semantik in den Systemen erschwert die modelltechnische Kopplung, da für die Verknüpfung der Modelle zusätzliche Informationen erforderlich sind.

2.4 Prozeßtechnische Sicht

Während des Entwicklungsprozesses werden in nahezu allen Phasen Berechnungen mit unterschiedlichen Zielsetzungen durchgeführt. Allen Berechnungen ist gemein, daß sie eine Vorhersage des Verhaltens oder der Eigenschaften eines Produkts oder einer Komponente liefern sollen. Dabei werden die Eigenschaften des realen oder geplanten Produkts auf ein *Modell* abgebildet, das dem Vorbild im Hinblick auf die zu untersuchenden Eigenschaften möglichst gut entspricht [Gra99].
Der Ablauf von Entwicklungsaktivitäten wird in zunehmendem Maße durch rechnergestützte Prozeßmanagementwerkzeuge gesteuert [KOM96]. Die Aufgabe dieser Werkzeuge besteht darin, sowohl die für den jeweiligen Arbeitsschritt benötigten CAE–Werkzeuge bereitzustellen als auch den Datenfluß zwischen den CAE–Werkzeugen prozeßbegleitend zu steuern und zu überwachen. Die Merkmale der Schritte *Gestalten* und *Berechnen* im Hinblick auf eine prozeßtechnische Integration werden in diesem Abschnitt erläutert.

2.4.1 Gestaltungsprozeß

Je nach Aufgabenstellung kann sich der Gestaltungsprozeß über die drei Konstruktionsphasen *Konzipieren*, *Entwerfen* oder *Ausarbeiten* erstrecken. Der Hauptanteil der Gestaltung erfolgt allerdings in der Entwurfsphase [PB97]. Neben der Entwicklung einer maßstäblichen Darstellung gehören zur Gestaltung die Festlegung der Werkstoffe und Fertigungsverfahren. Die Gestaltung wird dabei durch die Methoden zur Lösungssuche, Auswahl, Bewertung und Optimierung unterstützt und ist daher zahlreichen Iterationen unterworfen.

2.4.2 Einordnung von Berechnungen in den Produktentwicklungsprozeß

Nach Koller können Produkte erst bemessen (dimensioniert) werden, wenn deren qualitative Parameterwerte, bzw. deren Gestalt feststeht [Kol98]. Damit wären Berechnungen der Entwurfsphase oder frühestens dem Ende der Konzeptphase

zugeordnet. Tatsächlich kommen überschlägige Auslegungsberechnungen schon in früheren Phasen zum Einsatz, wenn es um die Auswahl und Bewertung von Wirkprinzipien geht [PB97]. Zu diesem Zeitpunkt ist die Bauteilgestalt gerade erst im Entstehen oder besteht aus einer Anzahl von Wirkorten. Im Sinne von Koller bedeutet das, daß zumindest die Parameter der zu untersuchenden Struktur feststehen müssen.
Der Detaillierungsgrad eines zu entwickelnden Produkts bestimmt auch die möglichen Zielsetzungen einer Berechnung. In Bild 2.6 sind mögliche Formen von Maschinenelementberechnungen und die Verknüpfungen zwischen den Modellen, bei denen mindestens ein Konzept des Produkts vorliegt, dargestellt. Ausgehend von diesem Konzept wird ein Ersatzmodell für eine Berechnung aufgebaut, das wesentliche Merkmale des zu entwickelnden Produkts abbildet. Das Konzept enthält üblicherweise nur wenige oder keine quantitative Geometriegrößen. Diese werden entweder durch die Berechnung bestimmt, oder müssen aus der Anforderungsliste entnommen oder aufgrund von festgelegten Hauptabmessungen abgeleitet werden. Das Konzept muß aber in seiner Struktur so festgelegt sein, daß der Einsatz der gewählten Berechnungsmethode gerechfertigt ist und eine Zuordnung der Komponenten beider Modelle sowie der für die Modellbildung wesentlichen Merkmale möglich ist. Bei Konstruktionen, denen ein bekanntes und bewährtes Lösungsprinzip zugrunde liegt, existieren meist auch Rechenverfahren für die Festlegung der Hauptabmessungen anhand von definierten Anforderungen [Kol98]. In diesem Fall ist die Abbildung der Konzeptparameter auf das Ersatzmodell nicht erforderlich. Vielmehr liefert eine Auslegungsrechnung als erster Schritt die Hauptabmessungen für die Funktionsträger. Nach Hubka werden in der Praxis viele Parameter auch mit “Konstruktionsgefühl” festgelegt und anschließend durch Nachrechnung überprüft [HE92], sodaß der Entwurf ohne Berechnung aus dem Konzept generiert wird.
Basierend auf den Hauptabmessungen wird im Geometriemodellierer das Produkt bis zum Entwurfsstadium ausgearbeitet. Mit der jetzt vorliegenden Detailgeometrie sind Nachrechnungen der zuvor vorgenommenen Vordimensionierungen möglich. Dafür müssen die im Gestaltungsprozeß erstellten Geometriedaten wieder in das Rechenmodell übertragen werden. Wenn das Ergebnis der Nachrechnung auf eine Über- oder Unterdimensionierung des Bauteils hinweist, können die aus der Nachrechnung gewonnenen Erkenntnisse in Form einer Optimierung wieder Rückwirkungen auf das Geometriemodell haben. Ähnliche Prozeßstrukturen ergeben sich bei der Durchführung von Montagesimulationen oder der Simulation von Fertigungsoperationen. Auch diese Operationen können in mehreren Iterationen durchlaufen werden, können ebenfalls Rückwirkungen auf das Geometriemodell haben und somit eine erneute Berechnung erforderlich machen. Anwenderprozeduren für die Auslegung und die anschließende Gestaltung von Schraubenverbindungen in einem CAD–System wurden Schwenke zusammengestellt [Sch91].
Je nach Laufzeit der Berechnung kann es sinnvoll sein, den Gestaltungsprozeß zu unterbrechen (asynchrone Anwendung), oder die Berechnung bei laufendem

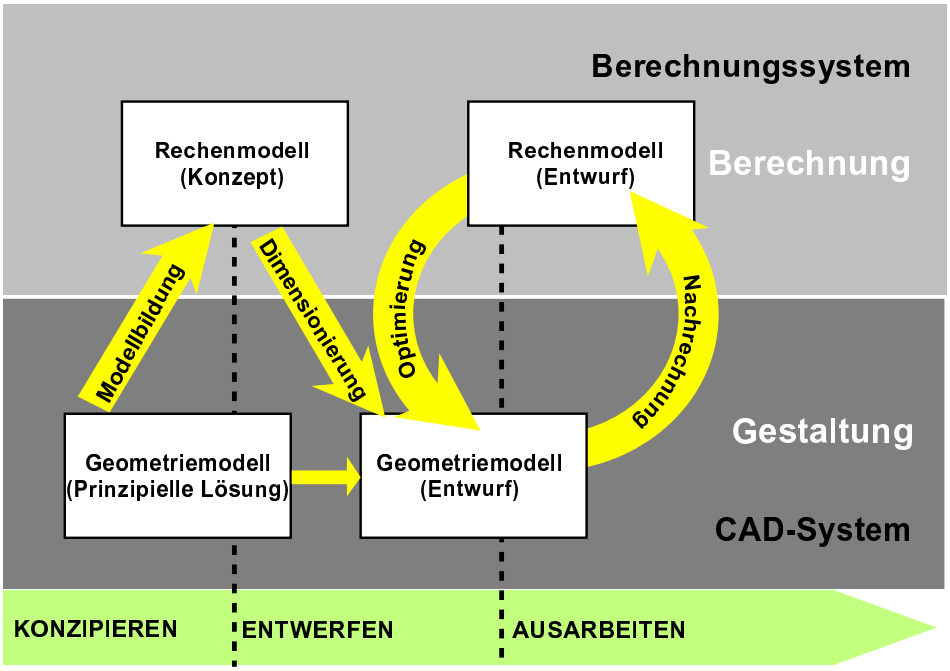


Bild 2.6: Mögliche Berechnungen im Produktentwicklungsprozeß

CAD–System durchzuführen (synchrone Anwendung).
Darüber hinaus können auch Vergleichsrechnungen durchgeführt werden, deren Ziel es ist, den Einfluß bestimmter Parameter oder die Sinnfälligkeit eines Ergebnisses zu verfizieren [Gal80]. Diese Berechnungen können in allen Konstruktionsphasen durchgeführt werden und stehen in Relation zu einer anderen, bereits durchgeführten Berechnung, die bezüglich des verwendeten Ersatzmodells Ähnlichkeit zu der aktuell durchgeführten hat.

2.4.3 Arbeitsschritte beim Berechnen

Eine Berechnung setzt sich aus mehreren Einzelschritten zusammen, die einen unterschiedlichen Bedarf an bereitzustellender Information [HE92] und ein unterschiedliches Potential an Automatisierbarkeit besitzen. In [Wöl98] werden die Schritte:

- Vorbereitung
- Berechnung
- Nachbereitung

unterschieden. Zur Berechnungsvorbereitung gehören die Modell- und Methodenauswahl sowie die Ermittlung der Eingangsgrößen durch Analysieren der Geometrie und Zusammenstellen der nichtgeometrischen Parameter. Wegen der in

Abschnitt 2.3.2 beschriebenen Spielräume bei der Modellbildung wird die Methodenauswahl immer die Erfahrung eines Bearbeiters erfordern. Die der Berechnung nachfolgende Bewertung kann durch Visualisierung der Ergebnisse und Bereitstellung zulässiger Werte unterstützt werden. Auf der Grundlage der Berechnung können dann Variationen oder Optimierungen durchgeführt werden, die durch eine automatische Anpassung der Geometrie oder durch eine programmierte Optimierungsstrategie unterstützt werden können. Die prozeßbegleitende und abschließende Erstellung der Dokumentation kann ebenfalls gut durch eine automatische Protokollierung von Bearbeiter, Datum, Versionen der Modelle und der verwendeten Methoden, sowie der Benutzereingaben automatisiert werden. Tabelle 2.3 zeigt eine Übersicht über die Schritte, die mögliche Rechnerunterstützung und die vom Bearbeiter benötigten Informationen.

Prozeßschritt	**Rechnerunterstützung**	**Informationsbedarf**
○ Modellauswahl	○ automatische Analyse der Konfiguration	○ Definitionen ○ Gültigkeitsbereiche ○ verfügbare Modelle
○ Abbildung der Geometrie auf Ersatzmodell	○ Geometrieanalyse ○ Ermittlung von Normdaten	○ Definitionen ○ Regeln ○ Modellannahmen ○ Gültigkeitsbereiche
○ Ermitlung der weiteren Parameter		○ Normen/Richtlinien ○ durchgef. Berechnungen ○ Anforderungen ○ Meßwerte
○ Berechnung	○ Abarbeitung der Formeln	○ verwendete Formeln ○ Modellannahmen ○ Herkunft von Parametern
○ Bewertung	○ Visualisierung ○ Vergleich mit zulässigen Werten	○ Zulässige Werte ○ Vergleichsrechnungen ○ Streubereiche ○ Einflußgrößen
○ Variation / Optimierung	○ Trendanalyse ○ Geometrievariation ○ automat. Optimierung	
○ Dokumentation	○ prozeßbegleitende Protokollierung	○ Bereits vorhandene Berechnungen

Tabelle 2.3: Möglichkeiten der Rechnerunterstützung und Informationsbedarf des Bearbeiters beim integrierten Berechnen

2.4.4 Auszutauschende Daten

Modelldaten

Je nachdem, um welche Art von Berechnungsmethode es sich handelt, variiert die Struktur der zwischen Gestaltungssystem und Berechnungssystem auszutauschenden Daten. Während allgemeine Methoden mit Oberflächen- oder Volumenmodellen auskommen, müssen für speziell auf ein Produkt oder eine Produktgruppe ausgelegte Methoden zusätzlich Informationen übertragen werden. Die zusätzlichen Informationen müssen entweder in den Modellen enthalten sein, oder mit Hilfe von in den Modellen enthaltenen Schlüsseln aus externen Quellen ermittelt werden. Flächenmodelle und Volumenmodelle lassen sich in der Regel problemlos aus einem parametrischen Geometriemodell ableiten. Schwieriger ist es hingegen, wenn aus einem durch eine allgemeine Methode generierten Volumenmodell ein parametrisiertes Geometriemodell abgeleitet werden soll, beispielsweise, wenn aus dem Ergebnis einer Topologieoptimierung ein Parametermodell erstellt werden soll. Dieser Schritt kann zur Zeit nur manuell vollzogen werden.
Die zu übertragenden Daten hängen von den internen Modellen der beteiligten Applikationen ab. Beim Übergang müssen gegebenenfalls Informationen hinzugefügt werden, die im Ausgangsmodell nicht vorhanden sind. Beispielsweise werden für viele Berechnungen die äußeren Lasten benötigt, die nicht im Geometriemodell enthalten sind. Wenn das Ergebnis einer Berechnung ein dimensioniertes Bauteil ist, muß dafür ein Geometriemodell erzeugt werden. Bild 2.7 gibt einen Überblick über den erforderlichen Datenaustausch.

Metadaten

Für eine prozeßtechnische Integration müssen auch die Arbeitsschritte bis zum Zustandekommen einer Berechnung und die nachfolgenden Schritte wie Rückführung und Dokumentation der Ergebnisse beachtet werden. Dieser Punkt gewinnt an Bedeutung, wenn man Berechnungen auch als Dienstleistungen, die ein externer Anbieter seinem Kunden zur Verfügung stellt, versteht.

Methodenauswahl Bei Zusammenstellung der Metadaten für die Methodenauswahl wird davon ausgegegangen, daß der Anwender bereits einen Überblick über die generell verfügbaren Methoden hat und auf der Suche nach einer Methode für ein bestimmtes Problem ist. Im einzelnen sollten folgende Informationen über eine Methode verfügbar sein.

Gültigkeitsbereich. Wesentlich für die Auswahl einer Methode ist die Frage, ob das zu berechnende System im Gültigkeitsbereich der Berechnungsmethode liegt. Zur Festlegung eines Gültigkeitsbereiches gehören neben Bereichsangaben für bestimmte Parameter auch Aussagen über die Verwendung bestimmter Normteile oder bestimmte Modellkonfigurationen.

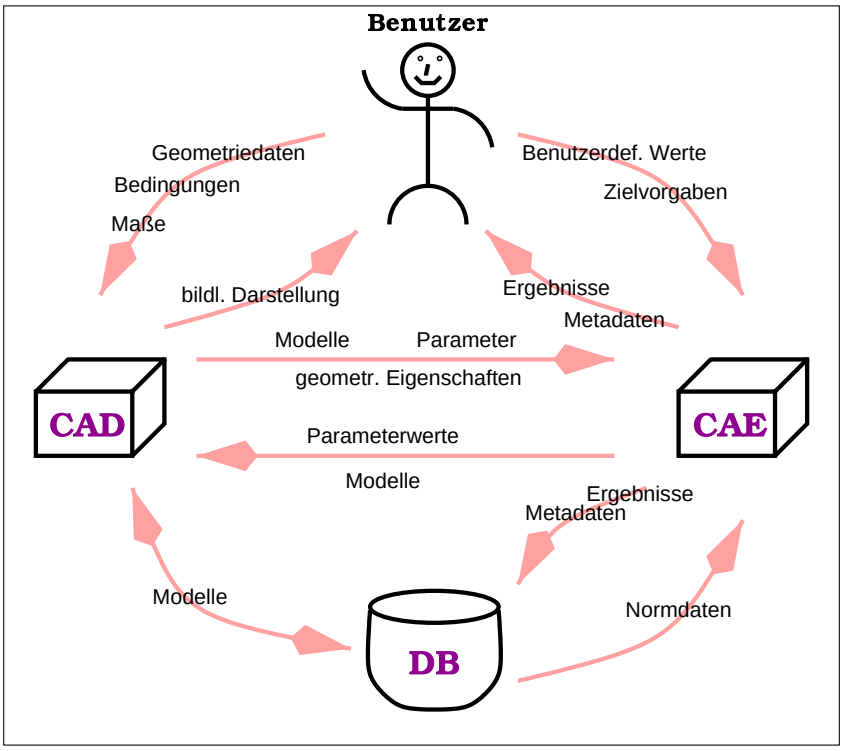


Bild 2.7: Informationsfluß beim integrierten Berechnen

Modellannahmen. Die einer Berechnung zugrunde liegenden Modellannahmen müssen dem Anwender bekannt sein. Oft sind diese Annahmen nicht quantifizierbar und müssen daher durch den Anwender gefühlsmäßig bewertet werden (Beispiel: “Ebenbleiben der Querschnitte”).

Ziel. Aus dem Ziel einer Berechnung resultieren die Einordnung in den Entwicklungsprozeß und die Art der erreichbaren Ergebnisse. Ziele können eine Auslegung, ein Festigkeitsnachweis oder eine Optimierung sein.

Eingangsgrößen. In Zusammenhang mit dem Ziel stehen auch die erforderlichen Eingabedaten. Diese müssen sowohl in maschinenlesbarer Form als auch für den Benutzer lesbar sein. Anhand dieser Daten muß der Anwender entscheiden können, ob die Anwendung einer Methode zum jetzigen Erkenntnisstand überhaupt möglich und sinnvoll ist.

Aussagegüte. Ein weiteres Kriterium für eine Methodenauswahl ist auch die erzielbare Aussagegüte. Dazu gehören auch Hinweise auf Empfindlichkeiten des Ergebnisses bezüglich bestimmter Parameter und Unsicherheiten bei der Bestimmung von Eingabewerten.

Dokumentation Die Dokumentation einer Berechnung soll die Voraussetzungen und Randbedingungen einer durchgeführten Berechnung zu einem späteren Zeitpunkt nachvollziehbar machen und die erzielten Ergebnisse festhalten. Für diesen Zweck sollten folgende Daten archiviert werden:

Quellen. Zu allen Parameterwerten, die nicht programmintern ermittelt werden, sollten die Quellen, aus denen der Wert bezogen wurde, protokolliert werden. Dazu gehören geometrische Größen, Werte aus der Anforderungsliste oder Benutzereingaben.

Durchführung. Bearbeiter, Datum und Uhrzeit der Berechnung gehören in die Dokumentation.

Verknüpfte Modelle. Die für eine Berechnung herangezogenen Modelle mit deren Bearbeitungsstand oder Version müssen ebenfalls protokolliert werden. Anhand dieser Daten kann festgestellt werden, welche weiteren Modelle bei einer Änderung betroffen sind.

Ergebnisse. Die Ergebnisse einer Berechnung müssen ebenfalls protokolliert werden. Die Ergebnisse können ermittelte Sicherheitsfaktoren, die durch eine Dimensionierung gefundenen Modelle oder Änderungsanforderungen an das Geometriemodell sein.

2.4.5 Probleme bei der Integration

Die verschiedenen möglichen Erscheinungsformen von Berechnungen erschweren eine genaue Einordnung eines Berechnungssystems in eine Prozeßkette. Vielmehr ist ein Berechnungssystem als ein Werkzeug zu sehen, das an verschiedenen Stellen des Entwicklungsprozesses eingesetzt werden kann. Berechnungen können für die Auswahl, die Bewertung und die Optimierung verwendet werden und lassen sich damit gut in den allgemeinen Gestaltungsprozeß einordnen (vergleiche Abschnitt 2.4.1).
Voraussetzung für eine Benutzung eines Berechnungsprogramms als Werkzeug zur Geometriemodellierung ist eine hinreichende Integration der beteiligten Modelle, da Eingangsgrößen und Ergebnisse einer Berechnung Bestandteile des Geometriemodells sein können. Außerdem muß das Berechnungsprogramm vom Arbeitsplatz des Konstrukteurs aus aufrufbar sein und sollte sich in die Benutzungsoberfläche und die Anwenderprozeduren des CAD–Systems einfügen. Methoden mit kurzer Laufzeit sollen ausführbar sein, ohne daß das CAD–System beendet werden muß. Methoden mit langer Laufzeit oder mit einem hohen Bedarf an Systemressourcen sollen auch eigenständig lauffähig sein.
In der Praxis haben die beteiligten Systeme unterschiedliche Vorgehensweisen bei der Benutzung. Erschwerend kommt hinzu, daß sowohl innerhalb der CAD–Systeme als auch innerhalb der Berechnungsprogramme verschiedene Anwenderprozeduren und Benutzungsoberflächen für die Modellierung existieren. Die

unterschiedliche Semantik der Modelle erschwert direkte Kopplung der Prozesse. Um die Arbeitsschritte bei der Geometriemodellierung, Geometrieerzeugung, Geometrievariation, Positionierung von Bauteilen und ähnliches mit Hilfe von Berechnungsprogrammen durchführen zu können, sind zusätzliche Informationen erforderlich, wie die Berechnungsergebnisse im Geometriemodell abgebildet werden können. Entsprechend müssen die Geometriedaten für eine Berechnung in die entsprechende Parameterdarstellung gebracht werden.
Die erforderlichen Metainformationen für die Auswahl der Methoden und die Verknüpfung der Modelle sind in keinem der Systeme vorhanden und müssen daher zusätzlich in das System eingebracht werden.

2.5 Systemtechnische Sicht

Gestaltungsaufgaben und Berechnungsaufgaben sind durch zahlreiche Programme und Programmsysteme rechnerunterstützt durchführbar. Dabei existieren für beide Aufgaben breite Spektren von Lösungen auf verschiedenen Rechnerplattformen mit unterschiedlichen Umsetzungen der oben genannten Modelle. Für eine Integration müssen Modelldaten zwischen den beteiligten Systemen ausgetauscht werden, die Systeme müssen in ein durchgängiges Ablaufschema integriert werden und die auf verschiedenen Plattformen bestehenden Applikationen müssen zusammengeführt werden. In diesem Abschnitt werden die möglichen Vorgehensweisen bei der Durchführung von integrierten Berechnungen, die Zugriffsmethoden auf die internen Modelle und die Plattformabhängigkeit der beteiligten Systeme beschrieben.

2.5.1 Mögliche Kopplungsformen

Unter der Voraussetzung, daß Berechnungsmethoden als Werkzeuge angesehen werden, die bei der Durchführung einer Gestaltungsaufgabe an verschiedenen Stellen eingesetzt werden können, muß eine Integration dafür sorgen, daß das Berechnungssystem mit den aktuellen Daten des Gestaltmodells arbeitet. Die Zusammenführung von Modelldaten und Methoden kann auf unterschiedliche Weise erfolgen:

Modellübertragung zwischen den Systemen (lose Kopplung): Vor der Berechnung wird das Modell oder ein Ausschnitt davon vom Geometriemodellierer zum Berechnungssystem übertragen und dort weiterverarbeitet. Das Original des Modells verbleibt dabei im erzeugenden System und kann nur dort variiert werden. Ein solche Kopplung wird auch als *lose Kopplung* bezeichnet [Hah98]. Für die Kopplung ist ein Dateiaustausch erforderlich, wobei zunehmend auch Übertragungsverfahren, die das Internet als Medium verwenden, zum Einsatz kommen. Üblich sind hier *FTP (File Transfer*

Protokoll) oder an *Emails* angehängte Dateien [VDI99]. Die Verwaltung bestimmter Modellversionen und die Archivierung der entstehenden Dokumente kann dabei von PDM–Systemen übernommen werden (vergleiche Abschnitt 3.3.1). Die Übertragung von Modellen kann sinnvoll bei Berechnungsprogrammen eingesetzt werden, die über eigene Geometrieanalyseroutinen verfügen und das übertragene Modell, zum Beispiel für eine Elementierung, weiterverarbeiten. Ein typisches Anwendungsbeispiel hierfür ist die Durchführung von FE–Analysen, die als Methoden mit langer Laufzeit meist asynchron und auch von Experten als Bearbeiter durchgeführt werden [Wöl98].

Methodenübertragung zum Anwender: Im Gegensatz zur Übertragung von Geometriemodellen können auch die Berechnungsmethoden vom Anbieter zum Anwender übertragen, und als Erweiterung des Geometriemodelliers verwendet werden. Die Methode wird dann lokal am Arbeitsplatz des Anwenders eingesetzt und arbeitet mit dem CAD–System und dem Geometriemodell des Anwenders. Die Übertragung von Methoden ist sinnvoll, wenn ein Anwender seine Geometriemodelle, beispielsweise aus Sicherheitsgründen, nicht weitergeben möchte. Ein Anwendungsbeispiel hierfür sind Auslegungsrechnungen für Zulieferkomponenten, bei denen die Berechnungsprogramme durch den Zulieferer zur Verfügung gestellt werden.

Direkte Verknüpfung der Programme: Die für eine integrierte Berechnung erforderliche Kopplung der Systeme kann auch mit einer direkten Netzwerkverbindung der Programme realisiert werden. Sowohl das Geometriemodell als auch die Berechnungsmethode blieben beim jeweiligen Eigentümer und nur die für die Berechnung erforderlichen Informationen würden übertragen. Diese Kopplung kann erforderlich werden, wenn die Methode nicht auf der Anwenderplattform lauffähig ist, aber trotzdem für die Berechnung die Funktionalität des Geometriemodellierers benötigt wird.

Idealerweise bietet das Integrationssystem die Möglichkeit, alle drei genannten Kopplungsformen zu realisieren, da sich für jeden Form sinnvolle Anwendungsfälle ergeben. Die Übertragung von Modellen hat allerdings den Nachteil, daß sie üblicherweise mit einer Modelltransformation und damit mit Datenverlust verbunden ist [Sei85]. Anzustreben ist daher eine direkte Kopplung von Berechnungssystem und CAD–System.

2.5.2 Zugriff auf die internen Modelle

Da in der vorliegenden Arbeit existierende Systeme zu einem integrierten System zusammengesetzt werden sollen, müssen die betrachteten Systeme auf die Zugänglichkeit ihrer internen Modelle untersucht werden. Grundsätzlich existieren zwei Möglichkeiten, Zugriff auf das Datenmodell eines Programms zu erhalten:

der programmgesteuerte Zugriff über eine Programmierschnittstelle und der Austausch über Dateien.

Programmierschnittstellen

Programmierschnittstellen (engl.: *Application Programmers Interface API*) bieten die Möglichkeit, zur Laufzeit des Programms auf die Funktionalität und auf die internen Datenmodelle zuzugreifen. In [KOM96] werden grundsätzlich drei Mechanismen für den programmgesteuerten Zugriff beschrieben:

Applikationsbezogen. Durch Nutzung der Programmierschnittstelle einer Applikation können neue Routinen zu der Grundfunktionalität des Systems hinzugefügt werden. Die Routinen sind an die Programmierschnittstelle des System gebunden und müssen auf dem Rechner, der das Grundsystem hält, lauffähig übersetzt vorliegen.

Kommunikationsorientiert prozedural. Durch die Bereitstellung von netzwerkfähigen Programmierschnittstellen können Applikationen, die auf verschiedenen Rechnern laufen miteinander kommunizieren. Klare Definitionen der Schnittstellen ermöglichen einen Modularen Aufbau des Gesamtsystems. Die zugreifenden Applikationen sind aber dennoch an die Schnittstellendefinition gebunden und müssen bei Änderung der Schnittstelle angepaßt werden.

Kommunikationsorientiert objektorientiert. Um die Vorteile des objektorientierten Ansatzes, Wiederverwendbarkeit, Erweiterbarkeit und Modifizierbarkeit auch für verteilte Applikationen nutzen zu können, wurden die Strukturen der prozeduralen Kommunikationsmechanismen für die Verwendung in objektorientierten Applikationen erweitert.

Für den programmgesteuerten Zugriff existieren im Bereich Geometriemodellierung und Berechnung keine standardisierten Schnittstellen. Vielmehr existieren zahlreiche applikationsspezifische Schnittstellen, die jeweils ein Abbild der Funktionalität und des internen Datenmodells der jeweiligen Applikation enthalten. Einheitliche Schnittstellen bieten die Systeme an, die auf dem selben Modellierkern, beispielsweise *ACIS* beruhen. Im Forschungsprojekt *ANICA* wurde versucht, aus den Programmierschnittstellen mehrerer bedeutender CAD–Systeme **eine** einheitliche Schnittstelle zu generieren, die die gemeinsame Funktionalität aller Schnittstellen abbildet [AJSK98]. Eine ausführliche Beschreibung des Projekts ANICA findet sich in Abschnitt 3.3.4.

Integrationsgrad

Da innerhalb eines Unternehmens meist ein spezielles CAD-System bevorzugt angewendet wird, werden integrierte Berechnungen oft durch systemspezifische

Erweiterungen der Berechnungsprogramme ermöglicht.
Je nach Branche und Produktspektrum kann dabei der Integrationsgrad unterschiedlich hoch gewählt werden. Bei einfachen Produkten mit immer wiederkehrenden Berechnungsmethoden können durch geeignete Wahl der Parametrisierung und durch Modularisierung der Methoden Modelle geschaffen werden, die direkt aufeinander abbildbar sind. In [The99] wird beispielsweise ein System vorgestellt, bei dem aus einer Reihe von Eingabedaten die günstigste Variante einer Getriebebaureihe und das zugehörige Geometriemodell generiert werden. Dagegen können in Bereichen mit wechselnden Randbedingungen und variantenreicher Struktur der Produkte sehr komplexe Abhängigkeiten zwischen den Modellen entstehen, die eine manuelle Anpassung erfordern. Beispiel hierfür ist die Auswahl von Wälzlagern für einen beliebigen Anwendungsfall, bei der die Steifigkeiten der Lagerumgebung in die Rechnung miteinbezogen werden müssen [LG99].

Dateischnittstellen

Ein anderes Verfahren, um Informationen zwischen Applikationen auszutauschen, ist die Kommunikation über Dateien. Für den Austausch von Geometriedaten existieren zahlreiche Dateiformate mit unterschiedlichem Leistungsumfang. Gelöst werden kann dadurch die Übertragung von 3–D Flächenmodellen oder Drahtmodellen. Allerdings existiert bislang kein Format, das in der Lage ist parametrische 3–D Modelle zwischen verschiedenen Systemen zu übertragen. Für die Kopplung verschiedener Applikationen, die, wie oben beschrieben in mehreren Iterationen durchlaufen werden kann, bedeutet dies, daß nach einer Iteration die Parametrik eines Modells verloren ist.
Eine Kommunikation über Dateien bedeutet auch, daß die Daten empfangende Applikation in der Lage sein muß, die Daten zu analysieren, und, was bei der Integration von Berechnungen auch sinnvoll ist, zu variieren. Ein Berechnungssystem wäre damit auch gleichzeitig Geometriemodellierer.
Mit STEP [ISO94] soll neben einem Produktmodellschema auch ein Dateiaustauschformat festgelegt werden, das in der Lage ist, **alle** produktbezogenen Daten zu übertragen. STEP wird im Abschnitt 3.1.1 genauer beschrieben.
Die existierenden Berechnungsprogramme verwenden alle eigene Dateiformate für die Speicherung von Berechnungen. Die Dateiformate sind überwiegend parameterorientiert [BOL94] [Dam96]. Einige Systeme stellen Zeichnungsausschnitte der berechneten Komponente im 2D–Geometriedatenaustauschformat DXF zur Verfügung [HEX95].

Vernetzung der Systeme

Sofern die auf verschiedenen Rechnerplattformen bestehenden Systeme nicht auf einer einzigen Plattform neu implementiert werden, muß davon ausgegangen werden, daß die Systeme auf verschiedenen Rechnern eines Netzwerkes laufen (siehe

auch Abschnitt 2.5.3). Die heutigen CAD–Systeme sind nicht für eine Anwendung in einer Client Server Umgebung konzipiert. Vielmehr werden die Systeme als lokale Einzelplatzlösungen eingesetzt. Die Modelle werden als Dateien in einem proprietären Format abgelegt. Die aktuellen Entwicklungen zielen darauf ab, daß neben den geometrischen Daten auch zahlreiche nichtgeometrische abgelegt werden und damit das CAD-System zum zentralen, universellen Entwicklungswerkzeug wird. Ein Grund für die Entwicklung der standallone–Lösungen ist auch der Entwicklungstrend in der Hardware. Die heutigen Arbeitsplatzsysteme, in Form von PC oder Workstations sind in bezug auf Rechenleistung und Grafikfähigkeiten so leistungsfähig, daß eine Benutzung auch komplexer Anwendungen problemlos möglich ist. Darüber hinaus erfordert das Arbeiten mit Geometriemodellen, insbesondere mit schattierten Darstellungen zur Zeit noch die Übertragung großer Datenmengen, die eine beträchtliche Netzwerkbelastung verursachen würden. Die Entwicklung einer netzwerkfähigen Beschreibungssprache für die räumliche Darstellung von Körpern, wie die *Virtual Reality Markup Language VRML* könnte hier zu neuen Lösungen führen [VRM99]. VRML ermöglicht den Transport von räumlichen Darstellungen über Netzwerke mit relativ geringem Datenvolumen.

Daraus folgt, daß derzeit auch bei einem Integrationssystem das CAD–System lokal am Arbeitsplatz vorhanden sein sollte. Das Berechnungssystem kann auch über eine Netzwerkverbindung angesprochen werden. In diesem Fall ist aber zusätzliche Software für die Abwicklung der Kommunikation zwischen den Systemen erforderlich.

2.5.3 Plattformabhängigkeit

Übersetzter Programmcode kann in den meisten Fällen nicht auf verschiedenen Rechnerplattformen ausgeführt werden. Die Ausnahme hierfür bildet die Programmiersprache Java (vgl. Abschnitt 3.2.3). Diese Plattformabhängigkeit führt dazu, daß Programme, wenn sie auf verschiedenen Systemen eingesetzt werden sollen, in verschiedenen Versionen übersetzt werden müssen. Die Portierung auf eine andere Rechnerplattform ist oft auch mit einer Anpassung der Software an unterschiedliche Konzepte für Speicherzugriffe oder Benutzungsschnittstellen verbunden.

Eine Portierung bestehender Software ist oft mit erheblichem Aufwand verbunden, sodaß in der Praxis die Betriebssysteme zugunsten der bestehenden Software nicht auf dem neuesten Stand sind oder verschiedene Systeme parallel betrieben werden [Abe97] [VDI99]. Die Tatsache, daß die Arbeitsumgebung aus heterogenen Systemen besteht, muß bei der Wahl des Integrationsansatzes berücksichtigt werden.

2.6 Anwendersicht

Berechnungsmethoden sollten sich möglichst reibungslos in die Tätigkeiten des Konstrukteurs integrieren. In diesem Abschnitt werden Anforderungen an die Methoden aus der Sicht des Anwenders und Szenarien für die Anwendung von Berechnungsmethoden vorgestellt.

2.6.1 Bereitstellung von Berechnungsmethoden

In vielen Unternehmen stellen die verwendeten Berechnungsmethoden einen nicht unerheblichen Teil des *Firmen-Know-How* dar [LG99] [The99]. Die zugrundeliegenden Modelle und Algorithmen werden oft über lange Zeit weiterentwickelt und verfeinert, sodaß sehr leistungsfähige, aber auch sehr komplexe Programmsysteme entstehen. Dabei ist die Programmentwicklung meist an eine Programmiersprache und an eine Hardwareplattform gebunden, da ein Systemwechsel mit einer aufwendigen Neuimplementierung der bestehenden Software verbunden wäre.

Firmeninterne Berechnungsmethoden

Für einen effizienten Einsatz der vorhandenen Berechnungssoftware sollten alle Methoden am Arbeitsplatz des Konstrukteurs verfügbar sein. Sofern die Programme alle auf der selben Rechnerplattform implementiert sind, die auch dem Konstrukteur zur Verfügung steht, kann eine Installation auf dem Arbeitsplatzrechner erfolgen. Anderenfalls kann eine Bereitstellung von Methoden im Firmennetzwerk (Intranet) erfolgen. Bei einer großen Anzahl verfügbarer Methoden ist es ratsam, Metainformationen für eine qualifizierte Auswahl von Methoden bereitzustellen. Zu den Metainformationen gehören Aussagen über Laufzeit, Aussagegüte, Gültigkeitsbereiche und Ähnliches.

Berechnungsmethoden von externen Partnern

Anbieter von Zulieferkomponenten stellen ihren Kunden oft die zu ihren Produkten gehörenden Auslegungsverfahren in Form von Berechnungsprogrammen zur Verfügung. Dabei müssen oft die individuellen Merkmale des Kundenprodukts in der Berechnung berücksichtigt werden. Im oben genannten Beispiel der Auslegung von Wälzlagern muß die Steifigkeit der Lagerumgebung entweder bekannt sein oder durch das Berechnungsprogramm ermittelt werden [LG99]. Im letzten Fall müssen dem Berechnungsprogramm alle benötigten Daten der Lagerumgebung zur Verfügung gestellt werden.

2.6.2 Qualifizierung der Bearbeiter

Eine sinnvolle Anwendung von Berechnungsmethoden kann nur erfolgen, wenn der Bearbeiter eine ausreichende Qualifikation für die Vorbereitung der Berech-

nung und die Bewertung der erzielten Ergebnisse besitzt, da sonst die Verläßlichkeit der Ergebnisse in Frage gestellt ist [Abe97]. Je nach Komplexität dieser Arbeitsschritte kann die Bearbeitung durch einen Experten erforderlich sein. Inbesondere bei der Bereitstellung von Berechnungmethoden über das Internet kommt diesem Punkt eine besondere Bedeutung zu. Aus der großen Zahl der möglicherweise angebotenen Methoden muß der Bearbeiter die für sein Problem am besten geeignete Methode auswählen können. Dazu müssen ihm die zugrundeliegenden Modellannahmen und Gültigkeitsbereiche bereitgestellt werden. Darüber hinaus müssen ihm Informationen über die erzielbare Aussagegüte und über Empfindlichkeiten oder Unsicherheiten bei der Bestimmung bestimmter Parameter aufgezeigt werden. In der Praxis sind zu den meisten Berechnungsprogrammen Ansprechpartner vorhanden, die die entsprechenden Auskünfte liefern können [LG99]. Oft werden auch die Berechnungen während eines Beratungsgesprächs beim Kunden durch den Anbieter durchgeführt.
Eine zusätzliche Qualifizierung des Bearbeiters auf dem Gebiet der Informationstechnik soll für die Anwendung eines Integrationssystems nicht erforderlich sein.

2.6.3 Qualität der Methoden

Bei der Bereitstellung von Methoden über das Internet ist die Qualität der angebotenen Methoden ein weiterer Aspekt. Damit ist gemeint, daß sowohl die theoretischen Grundlagen der Methode hinreichend abgesichert sind als auch die Implementierung genau dem entwickelten Rechenmodell entspricht. Bei innerbetrieblichen Anwendungen wird davon ausgegangen, daß die Entwicklungsabteilung die bereitgestellten Programme ausreichend getestet hat, um die einwandfreie Funktion sicherzustellen. Ebenso verhält es sich mit Programmen, die einem Anwender von einem Zulieferer bereitgestellt werden. Unsicher ist allerdings, wie ein solches Vertrauensverhältnis zu einem noch nicht bekannten Anbieter aufgebaut werden kann. In der Diskussion sind derzeit sogenannte “Broker”, die Anfragen zu bestimmten Problemen an qualifizierte Einrichtungen weitervermitteln. Einen Ansatz hierfür bildet das *Berliner Kreis Kompetenz Netzwerk* [N.N99].

2.7 Zusammenfassung

In Tabelle 2.4 sind zusammenfassend die Merkmale dargestellt, die eine Integration von Berechnungsprogrammen in Geometriemodellierer beschreiben. Dabei sind diejenigen Eigenschaften hervorgehoben, die in der weiteren Arbeit berücksichtigt werden sollen.

Bereich	*Merkmale*			
Methoden	**Laufzeit**	lang	mittel	kurz
	Anwendungs-gebiet	produktspezifisch	Produktgruppe	allgemein
	Integrierbarkeit	direkt	indirekt	lose
	Ziel	Auslegung	Nachweis	Optimierung
Modelle	**Modelltyp**	Geometrie (Flächen, Volumen)	Parametermodell	
	Konfiguration	starr vorgegeben	vorgegebene Varianten	flexibel
	Integration von Geometrie	auf CAD-System abgestimmt	allgemeine Austauschformate	eigenes Format für Geometrie
Prozesse	**Zeitliche Abfolge**	synchron	asynchron	
	Lokalisierung	selber Rechner	Intranet	Internet
	Plattform	selbe Plattform	verschiedene Plattformen	
	Zugriff auf Geometrie	Analyse	Variation	Erzeugung
Systeme	**Kopplungsform**	Modell-übertragung	Methoden-übertragung	direkte Kopplung
	Benutzungs-schnittstelle	einheitlich	gleiche Grundelemente	eigene
	Zugriff	dateiorientiert	Programmier-schnittstelle	
	Programmier-sprache	selbe Sprache	verschiedene Sprachen	
Anwender	**Bearbeiter**	Konstrukteur	Experte	
	Qualifizierung	gering	ausreichend	hoch

Tabelle 2.4: Merkmale einer integrierten Berechnung

3 Integrationsansätze

In diesem Kapitel werden grundlegende Ansätze vorgestellt, die für die gestellte Integrationsaufgabe von Bedeutung sind. Im ersten Abschnitt werden das integrierte Produktmodell von *STEP* als Ansatz für eine modelltechnische Integration und die Systemarchitektur des *CAD-Referenzmodell* als Grundlage für ein Systemkonzept vorgestellt. Im zweiten Abschnitt werden mit CORBA und JAVA zwei Ansätze vorgestellt, die eine plattformübergreifende und netzwerkfähige Nutzung von Softwaresystemen ermöglichen. Im dritten Abschnitt werden wichtige bestehende Integrationssysteme vorgestellt, deren Konzepte teilweise auf den zuvor beschriebenen Ansätzen beruhen.
Aus den beschriebenen Systemen werden abschließend allgemeine Integrationsstrategien abgeleitet.

3.1 Integrierte Produktentwicklung

3.1.1 STEP - ein integriertes Produktmodell

Ein Ansatz, sämtliche Produktdaten in einem einheitlichen Format abzubilden soll mit *STEP* geschaffen werden. *STEP* steht für *Standard for Exchange of Product Model Data*, und wird seit 1984 von der *International Organisation of Standardization (ISO)* entwickelt.

Zielsetzung

Das Ziel der Entwicklung von STEP ist es, eine "eindeutige Repräsentation von computer-interpretierbaren Produktinformationen für den gesamten Lebenszyklus eines Produkts" zu schaffen [ISO94]. Als Phasen des Produktlebenszyklus werden dabei Entwicklung, Fertigung, Vertrieb, Wartung und Entsorgung genannt. Bislang existieren in den verschiedenen am Produktentwicklungsprozeß beteiligten Systemen unterschiedliche Modelle. Mit STEP sollen diese Modelle ersetzt werden durch ein eindeutiges, konsistentes und redundanzfreies Modell. Dadurch soll eine implementierungsneutrale Beschreibung der Produktdaten in einem Produktmodell erreicht werden [GEP94]. Aus der historischen Entwicklung heraus war STEP als Dateiaustauschformat geplant. Inzwischen wurden

ergänzend dazu mit *Standard Data Access Interface (SDAI)* standardisierte Zugriffsmethoden festgelegt, mit deren Hilfe über Programmiersprachen auf die Produktdatenbasis zugegriffen werden kann.

Struktur

Die Norm besteht aus mehreren aufeinander aufbauenden Serien. Die Zusammenhänge zwischen den einzelnen Serien sind in Bild 3.1 dargestellt und werden nachfolgend erläutert [Nel99].

Die STEP Serien können in sechs Hauptgruppen eingeteilt werden: *description methods*, *integrated information resources*, *application protocolls*, *implementation methods* and *conformance methodolgy*. Die *description methods* bilden die Grundlage von STEP. Sie enthalten in Teil 1 die für alle weiteren Teile geltenden Definitionen. Ebenso gehört zu den Grundlagen die Beschreibung von *EXPRESS*, einer Sprache für die Modellierung von Daten, in der sämtliche Konstrukte von STEP abgebildet sind.

Die eigentlichen Datenmodelle von STEP sind in den *Integrated Information Resources* beschrieben. Diese sind nochmals unterteilt in drei weitere Gruppen. Die erste, die *Generic Resources* bilden allgemeine Formen der in den Anwendungsprotokollen benötigten Objekte ab. Die applikationsspezifischen Ergänzungen dieser Objekte werden in den *Application Resoures* vorgenommen. Die hier definierten Konstrukte können von allen Anwendungsprotokollen verwendet werden und bilden somit die Grundlage für eine einfache Integration und hohe Interoperabilität. In der dritten Schicht der *Integrated Information Resources*, den *Application Interpreted Resources* werden Elemente der beiden anderen Ebenen zu Gruppen zusammengefaßt. Diese Gruppen beschreiben Modelle in bestimmten wiederverwendbaren Abstraktionsebenen und ermöglichen es auf diese Weise, die gleiche Semantik in verschiedenen Anwendungen abzubilden.

Die *Information Resources* werden in den *Application Protocolls* zu definierten Modellen zusammengefügt, die den Zustand eines Produkts zu einem bestimmten Zeitpunkt seines Lebenszyklus beschreiben. Die hier definierten Konstrukte sind die in den Anwendungsprogrammen verwendeten Modelle.

In der Gruppe der *Implementation Methods* sind Abbildungsvorschriften für die Abbildung der STEP Konstrukte in eine Reihe von üblichen Programmiersprachen festgeschrieben. In einem eigenen Bereich, *Conformance Testing*, sind Anweisungen für die Überprüfung der Konformität behandelt. In dieser Hinsicht ist STEP einzigartig, da die Richtlinien zur Konformitätsprüfung in der Norm selbst enthalten sind.

Für die Konformitätsprüfung werden in den *Abstract Test Suites* zu jedem Anwendungsprotokoll Testverfahren beschrieben.

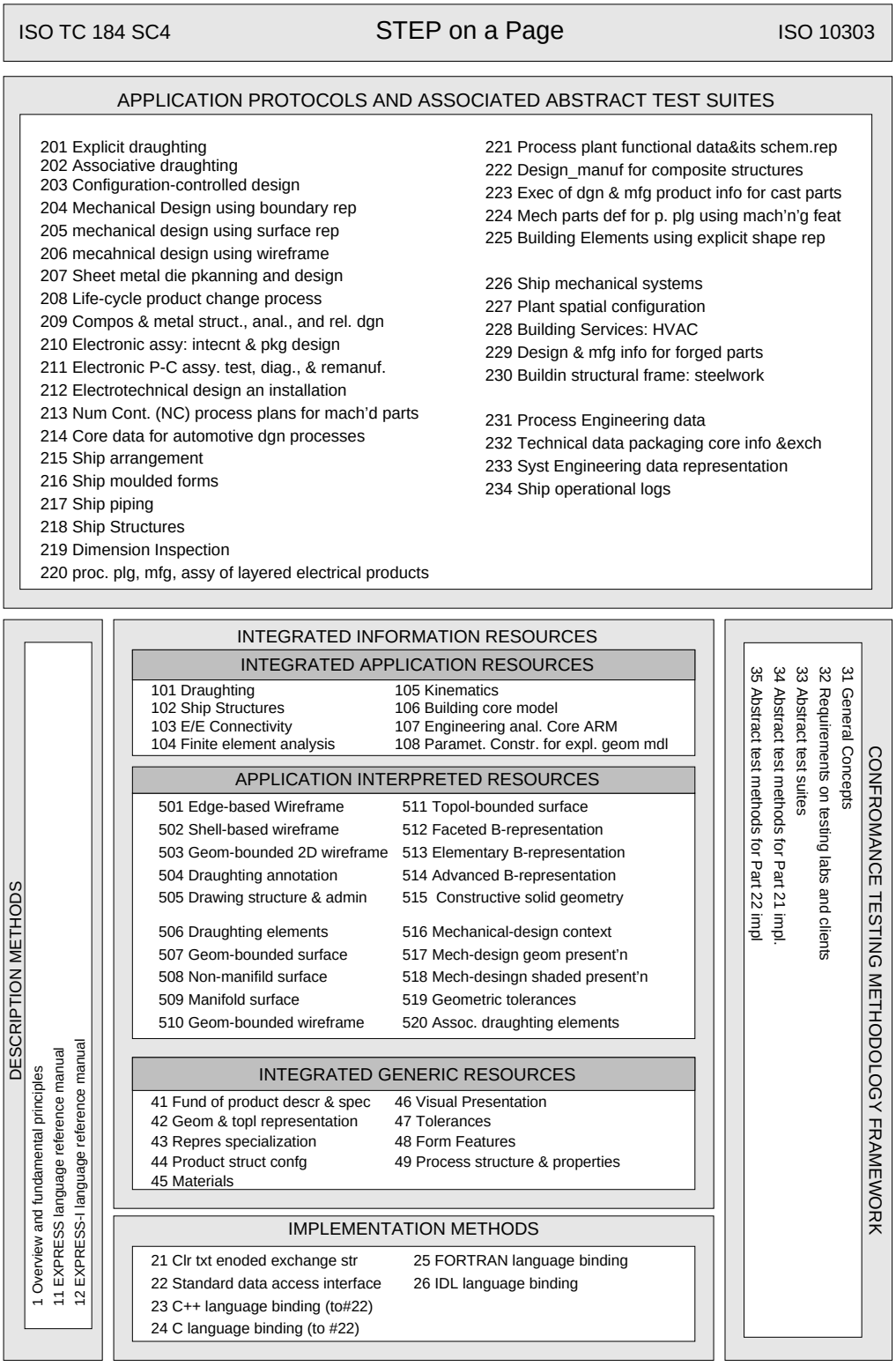


Bild 3.1: Übersicht über die verschiedenen Bestandteile von STEP [Nel99]

Anwendung

Die Anwendung von STEP als Datenbasis für sämtliche produktbezogenen Daten ist zur Zeit noch nicht gegeben. Dies liegt hauptsächlich daran, daß insbesondere für die nichtgeometrischen Daten noch keine Applikationen existieren, die in der Lage sind, STEP Modelle zu verarbeiten. Ansätze, bei denen STEP als Datenbasis für die Abbildung von Geometriemodellen verwendet wird, sind in [Har97] und [SAKJ98] beschrieben.

Im Bereich der Geometrieverarbeitung wird STEP bereits als Datenaustauschformat eingesetzt. Die Fähigkeit, auch Produktstrukturen abbilden zu können [GAS92], macht STEP zu einem geeigneten Werkzeug, um die Geometriemodelle einer heterogenen CAD-Umgebung zu einem digitalen Gesamtmodell zusammenzufügen. In [Pot95] wird eine Lösung für den Austausch von Geometriemodellen über STEP bei ACIS-basierten Geometriemodellierern beschrieben. In [SAKJ98] wird eine Anwendung vorgestellt, bei der mit Hilfe von STEP Volumenmodelle aus dem CAD-System Pro/ENGINEER für eine Bauraumuntersuchung in das CAD-System CATIA importiert wurden. Ein Problem bei der Verwendung von STEP als Austauschformat ist, daß auch STEP nicht in der Lage ist, sämtliche in den CAD-Systemen enthaltenen Informationen abzubilden und es daher zu einem Informationsverlust kommt. Insbesondere können keine parametrischen Modelle übertragen werden. Die Entwicklung von STEP ist dabei von einem Zielkonflikt zwischen den Nutzern und den Entwicklern der CAD-Systeme geprägt. Während die Nutzer erwarten, daß sie CAD-Modelle zwischen beliebigen Systemen austauschen können, sind die Systemhersteller bemüht, ihrem eigenen System besondere Merkmale zu geben, um sich vom Mitbewerber abheben zu können und die Anwender an das eigene System zu binden.

3.1.2 CAD–Referenzmodell

In einem von acht deutschen Forschungsinstituten über zwei Jahre durchgeführten Forschungsvorhaben wurde eine Referenzarchitektur für zukünftige CAD–Systeme erarbeitet. Das Ziel dieses Vorhabens bestand darin, auf der Grundlage des Stands der Technik die Anforderungen an zukünftige CAD–Systeme zusammenzustellen und daraus eine Systemarchitektur abzuleiten, die als richtungsweisend für die Entwicklungen der nächsten Jahre gelten kann. Motiviert war diese Arbeit durch die seit der Einführung der CAD–Systeme entstandene große Zahl von Systemarchitekturen, die eine Interoperabilität zwischen zwei Systemen oder einen Systemwechsel erschwerte. In der Praxis führt das dazu, daß ein großer Teil der Anwender nicht mit den neuesten Entwicklungen Schritt hält, sondern aus Kompatibilitätsgründen bei veralteten Anwendungen stehen bleibt.

Anforderungen

Neben einer Beschreibung eines menschengerechten Entwicklungsarbeitsplatzes wurde in dem Forschungsvorhaben eine sehr ausführliche Liste von Anforderungen an zukünftige CAD–Systeme zusammengestellt [Abe95]. Dabei werden Aspekte der Arbeits- und der Konstruktionswissenschaft, der Konfiguration, des Benutzungsdialogs, der Berechnung, der Analyse und Simulation ebenso behandelt wie Anforderungen bezüglich Wissenverarbeitung, Modellierung, Produktmodell und Integrationsmöglichkeiten. An dieser Stelle sollen die Anforderungen in Bezug auf Berechnung, Analyse und Simulation herausgestellt werden. Die drei Punkte werden unter dem Begriff *Analysen* zusammengefaßt, der als *Methoden zur Lösungsfindung und Beurteilung* definiert ist. Da diese Methoden häufig in Form von problemspezifischen oder anwenderspezifischen Programmen implementiert sind, werden an die Software die Forderungen nach maximaler Modularität und Kombinierbarkeit sowie der nutzergerechten Gestaltung gestellt.

Referenzarchitektur

Die Grundstruktur der Referenzarchitektur gliedert sich in die vier Hauptkomponenten Anwendungsteil, Systemteil, Produktmodell und anwendungsspezifisches Wissen (Bild 3.2). Die Komponenten werden in [Abe95] wie folgt definiert:

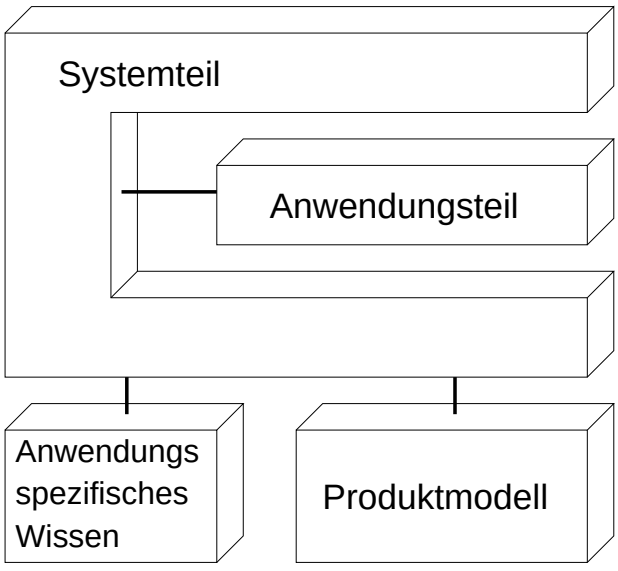


Bild 3.2: Systemstruktur des CAD–Referenzmodells [Abe95]

Anwendungsteil. Menge der in einem CAD–System verfügbaren anwendungsbezogenen Komponenten zur Realisierung konstruktionsspezifischer Funktionalität.

Systemteil. Menge der in einem CAD–System verfügbaren anwendungsunabhängigen Komponenten, die für die Bereitstellung, Konfigurierung, Abarbeitung und Integration von Komponenten des Anwendungsteils benötigt werden.

Produktmodell. Einheit von Produktinformationsmodell und Produktdaten zur Beschreibung einer Klasse von Produkten über den gesamten Produktlebenszyklus.

Anwendungsspezifisches Wissen. Menge von anwendungsspezifischem Wissen zur Lösung von Konstruktionsaufgaben.

Der Anwendungsteil mit den eigentlichen CAx–Applikationen ist dabei vom Systemteil umgeben. Der Systemteil stellt in dieser Symbolik die Schnittstelle zum Benutzer und zu den verbundenen Datenbasen für das anwendungsspezifische Wissen und das Produktmodell dar (vergleiche Bild 3.3).
Die deutliche Trennung der Anwendungs- und Systemkomponenten in der Referenzarchitektur soll die Austauschbarkeit und Erweiterbarkeit sowohl der anwendungsbezogenen als auch der anwendungsunabhängigien Komponenten unterstützen [Abe95]. Die Modularisierung setzt sich innerhalb der einzelnen Komponenten fort. Bild 3.3 [Abe95] zeigt die verschiedenen Subsyssteme des Systemteils, die über eine gemeinsam genutzte Kommunikationspipeline miteinander verbunden sind.

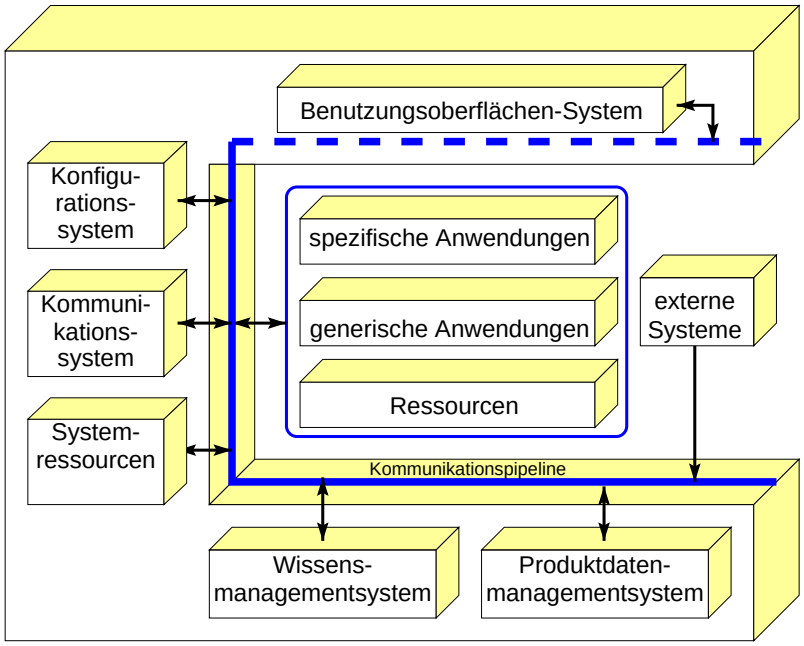


Bild 3.3: Struktur des Systemteils

Anwendung

In einer zweiten Phase des Projektes soll versucht werden, die Ansätze und Konzepte der ersten Phase zu implementieren und deren Effizienz in der industrieellen Anwendung nachzuweisen. Dafür wurden Defizite in den rechnerunterstützten Produktentwicklungsprozessen herausgearbeitet und daraus Innovationfelder für das Referenzmodell abgeleitet. Die drei wesentlichen Innovationsfelder sind nach [Abe97]: *Organisation und durchgängige Unterstützung von Produktentwicklungsprozessen*, *Verteilte kooperative Produktentwicklung* und *Systemtechnische Integration und Optimierung von Produktentwicklungsumgebungen.* In verschiedenen Teilprojekten wurden diese Felder mit unternehmensspezifischen Entwicklungen bearbeitet [Abe97]. Neben der Entwicklung einer durchgängigen Produktdatenhaltung wurden als Ziele hauptsächlich eine Restrukturierung und durchgehende Rechnerunterstützung der Entwicklungsprozesse angestrebt.

Hervorzuheben ist die klar gegliederte und umfassende Systemstruktur des CAD–Referenzmodells. Die Struktur ist für die Integration bestehender Softwaresysteme aber insofern als Referenzstruktur zu sehen, daß die zu integrierenden Systeme jeweils mehrere Komponenten umfassen und daher eine klare Trennung nicht mehr vorgenommen werden kann. Die oben genannten Innovationsfelder können als Randbedingungen für die Integrationsaufgabe gesehen werden.

3.2 Integration von Softwaresystemen

Mit der zunehmenden Rechnerunterstützung des Produktentwicklungsprozesses wächst die Forderung, daß die in den verschiedenen Anwendungsbereichen bereits existierenden Softwaresysteme integriert werden müssen. Die Grundstruktur vernetzter Programme ist eine Client-Server-Beziehung, in der eine Applikation als Server Daten und Methoden zur Verfügung stellt, die von der anderen als Client in Anspruch genommen werden.

In der Informationstechnik existieren für die Realisierung von Client-Server Systemen verschiedene Lösungsansätze. In einem Überblick werden zunächst die wichtigsten Ansätze vorgestellt. Anschließend werden zwei der für die Integrationsaufgabe am geeignetsten erscheinenden Ansätze weiter erläutert. Der erste ist *CORBA*, eine Systemarchitektur, die es ermöglicht, Objekte plattformübergreifend und netzwerktransparent zu verwenden, indem existierende Applikationen über neutrale Schnittstellen gekapselt und an ein standardisiertes Kommunikationssystem angeschlossen werden. Der zweite Ansatz ist die Programmiersprache *JAVA*, die mit dem Ziel entwickelt wurde, eine objektorientierte Sprache zu schaffen, deren übersetzter Code auf allen Rechnerplattformen ohne Neuübersetzung lauffähig ist.

3.2.1 Überblick über Client-Server Umgebungen

Als Basis für die Kommunikation in Netzwerken wurde von der ISO das hierarchisch gegliederte OSI-Schichtenmodell entwickelt (*OSI - Open Systems Interconnection*). In dem Modell werden sieben Transportschichten für unterschiedliche Informationen definiert. Die Schichten reichen vom Verbindungsaufbau über die Sicherung von Transportwegen, die Defintion von Vermittlungsdiensten und die Fehlerkorrektur bis zu Regeln für den Ablauf der Kommunikation, die Form der Informationsdarstellung und die Randbedingungen für Anwenderprogramme [Sch88]. Das Übertragungsmedium und die Anwendungsfunktionen sind nicht Bestandteil der Standardisierung. Bei der Implementierung eines Integrationssystems wird in erster Linie auf die Anwendungsfunktionen eingegangen. Die Schichten des OSI–Modells werden durch die nachfolgend beschriebenen Kopplungsmechanismen auf unterschiedliche Weise bedient.

Eine einfache Client-Server Beziehung kann über *Sockets* realisiert werden. Ein Socket ist ein Verbindungsendpunkt und wird über eine Netzwerkadresse und eine Portnummer identifiziert. Mit Sockets sind Methodenaufrufe über ein Netzwerk hinweg ausführbar. Aus Programmierersicht werden durch ein Socket die Details einer Netzwerkverbindung verdeckt. Allerdings müssen für den Aufbau einer Socketverbindung die Identifizierungsmerkmale des Servers bekannt sein. Die Methodenaufrufe des Client müssen auf die Programmierschnittstelle des Servers abgestimmt sein. Implementierungen von Sockets existieren auf nahezu allen Rechnerplattformen [OH97].

Die Kombination des *Hypertext Transfer Protocoll HTTP* mit dem *Common Gateway Interface CGI* ermöglicht es einem Client, über ein html-Formular ein Programm auf dem Server zu starten und dessen Ausgaben per html-Dokument zu beziehen. Der Datentransport ist an das html-Format gebunden. Für jede Datenübertragung muß eine neue Verbindung aufgebaut werden. Zwischen zwei Programmaufrufen muß der Abfragestatus im Client gespeichert werden, da das Serverprogramm mit jedem Aufruf neu gesartet wird [OH97].

Eine komplette Infrastruktur für verteilte Softwaresysteme wird mit der *Distributed Computing Environment DCE* bereitgestellt [Ope93]. DCE ermöglicht die Trennung von Verwendung und Implementierung eines Softwareprogramms durch die Einfürung einer neutralen Schnittstellenbeschreibungssprache und des Kommunikationsmechanismus RPC [Ope91]. Darüber hinaus stehen in DCE ein Namensdienst, Mechanismen für Zeitsynchronisierung und die Integration von PC–Arbeitsplätzen sowie standardisierte Sicherheitsmechanismen zur Verfügung. Mit Hilfe des *Distributed File System* können Dateisysteme verschiedener Rechner zu einem einzigen, erweiterten Dateisystem zusammengesetzt werden. Die Sicherheitsdienste, der Namensdienst sowie die Technik, Applikationen über neutrale Schnittstellen zu kapseln, wurde bei der Entwicklung von CORBA aufgegriffen. Im Vergleich zu CORBA ist jedoch das Systemkonzept von DCE nicht objektorientiert, sondern prozedural aufgebaut. Das bedeutet, daß die Komponenten

einer DCE–Umgebung und die darin definierten Applikationen nicht durch einfache Vererbung und Erweiterung angepaßt werden können, sondern gegebenenfalls komplett neu implementiert werden müssen [OHE96].
Ein plattformgebundenes Konzept für die Integration verschiedener Softwaresysteme ist das Betriebssystem Windows mit dem Objektmodell *DCOM (Distributed Component Object Model.* Ähnlich wie in CORBA ist mit DCOM die netzwerktransparente Verwendung von Objekten möglich. Allgemeine und spezielle Verzeichnisse und Dienste werden durch das Betriebssystem bereitgestellt [OH97].
Ein umfassende Darstellung und ein Vergleich der Konzepte ist ist in [OHE96] und [OH97] zu finden.
Aus dem Spektrum der möglichen Lösungsansätze scheint CORBA am geeignetsten, da mit CORBA ausdrücklich auch bestehende Softwaresysteme verschiedener Rechnerplattformen und unterschiedlicher Programmiersprachen integriert werden können. Die Technik, bestehende, auch prozedurale Applikationen durch eine objektorientierte Schnittstellenbeschreibung zu kapseln, ermöglicht die Schaffung eines applikationsübergreifend gültigen Modellschemas. Darüber hinaus steht mit einer CORBA–Umgebung eine vollständige Arbeitsumgebung für verteilte Objekte zur Verfügung. Die Verzeichnisse und Dienste dieser Arbeitsumgebung lösen viele der Probleme, die beim Arbeiten mit verteilten Systemen entstehen auf einheitliche Weise.
Wie in Kapitel 2 festgestellt wurde, sind für eine Kopplung der bestehenden Systeme zusätzliche Informationen erforderlich, die in Form von Softwarekomponenten bereitgestellt werden müssen. Da auch hier eine plattformübergreifende Nutzung möglich sein soll, muß für diese Softwarekomponenten eine Programmiersprache gewählt werden, die auf mehreren Plattformen verfügbar ist. JAVA erfüllt diese Forderung, da sie als plattformunabhängige Sprache entwickelt wurde. Darüberhinaus lassen sich mit JAVA auch plattformunabhängige Benutzungsoberflächen generieren. Die beiden Technologien CORBA und JAVA sollen für die Entwicklung eines Integrationssystems verwendet werden und werden in den folgenden Abschnitten näher beschrieben.

3.2.2 CORBA - eine Systemarchitektur für verteilte Objekte

CORBA steht für "*Common Object Request Broker Architecture*" und ist ein Standard für Verwaltung von und die Kommunikation zwischen verteilten Softwareobjekten. Als Bestandteil von der *Object Managing Architecture OMA* ist in *CORBA* spezifiziert, wie Softwareobjekte netzwerk- und plattformübergreifend angesprochen, transportiert und verwendet werden können. Die *OMA* wird von der *Object Managing Group OMG*, einem nicht profitorientierten Zusammenschluß zahlreicher Hardwarehersteller, Softwareentwickler und Netzwerkbetreiber zur Förderung der Objekt–Technologie, entwickelt und festgeschrieben

[Red96]. Dabei werden von der *OMG* lediglich die Inhalte des Standards geliefert. Die Implementierung wird durch die Softwarehersteller vorgenommen.

Struktur

Die verschiedenen in der *OMA* definierten Komponenten sind in Bild 3.4 dargestellt. Zentrale Komponente dieser Architektur ist der *Object Request Broker ORB*, der den Kommunikationskanal zwischen allen beteiligten Ressourcen bildet. In der Gruppe der *Common Object Services COS* sind eine Reihe von Diensten definiert, die allen Objekten und Applikationen im System zur Verfügung stehen und anwendungsunabhängig sind. Die eigentlichen Applikationen sind in den *Application Objects* gekapselt und ebenfalls an den *ORB* angeschlossen. In einem separaten Block sind die für ein Anwendungsgebiet definierten Objekte und deren Dienste in Form der *Common Facilities* definiert. Zusammen mit den *Application Objects* bilden sie eine Arbeitsumgebung (engl.: Framework) für ein bestimmtes Anwendungsgebiet.

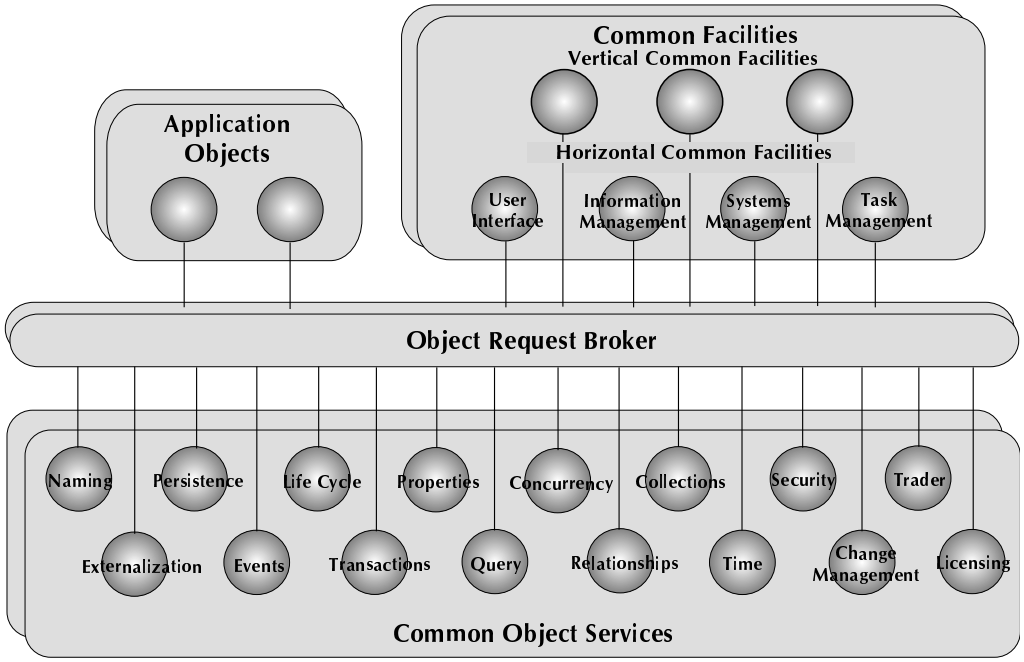


Bild 3.4: Komponenten der *Object Managing Architecture (OMA)* [OHE96]

Der Object Request Broker (ORB)

Der *Object Request Broker (ORB)* kann als Objektbus definiert werden [OHE96]. Objekte können über ihn Methodenaufrufe an andere Objekte senden und die zugehörigen Rückgabewerte beziehen. Client- und Serverobjekte können dabei

sowohl auf verschiedenen Knoten in einem Netzwerk als auch auf unterschiedlichen Rechnerplattformen implementiert sein. Der *ORB* sorgt dafür, daß die Zugriffe auf entfernt liegende Objekte für den Client so erscheinen, als wären sie lokal.
Die Plattformunabhängigkeit wird erreicht, indem sämtliche im Gesamtsystem vorkommenden Objekte, deren Attribute und Methoden in einer neutralen Schnittstellenbeschreibungssprache IDL (*Interface Definition Language*) beschrieben werden. Für die Verbindung zwischen den Applikationen und dem *ORB* werden für alle in der IDL-Datei definierten Objekte bei der Softwareerstellung sogenannte *Object Adapter* generiert, die die über den *ORB* kommenden Funktionsaufrufe umsetzen in Aufrufe im lokalen Adressraum. Das CORBA-Laufzeitsystem verfügt über Verzeichnisse (engl.: Repositories) der im System definierten Objekte und deren Lokalisierung. Die folgende Liste gibt einen Überblick über die Merkmale von CORBA [OHE96].

Statische und Dynamische Methodenaufrufe. In einer CORBA Umgebung sind sowohl statische als auch dynamische Methodenaufrufe möglich. Die statischen Aufrufe werden bereits bei der Softwareherstellung in das Programm eingebunden und sind dadurch schneller, da die Typprüfung beim Übersetzen erfolgt, und die Lokalisierung der Methode feststeht. Anders als andere Kommunikationsmechanismen erlaubt CORBA auch ein dynamisches Binden zur Laufzeit, mit dem Vorteil, daß bei der Erstellung eines Softwareprodukts nicht alle Bibliotheken, mit denen das Produkt später einmal zusammen verwendet werden soll, bekannt sein müssen. Offenbarer Nachteil dieser Methode ist, daß durch die zur Laufzeit durchgeführte Typprüfung und Methodenlokalisierung längere Ausführungszeiten entstehen.

Verknüpfungen zu Objekten in anderen Hochsprachen. Die Zuordnung von den in der Interface Definition Language beschriebenen Strukturen zu den entsprechenden Konstrukten in einer für die Implementierung gewählten Programmiersprache erfolgt in *Language Bindings*. Derzeit sind Umsetzer für die Sprachen C, C++, Java und Smalltalk vorhanden. Durch die strikte Trennung von Schnittstelle und der zugehörigen Implementierung können auch Programme, die in verschiedenen Sprachen implementiert sind, miteinander kommunizieren.

Selbsterklärende Komponenten. Durch die Vorhaltung der Schnittstellenbeschreibungen in den entsprechenden Verzeichnissen können zur Laufzeit Informationen über Eigenschaften von Objekten ermittelt werden. Dies kann beispielsweise benutzt werden, um einem dynamischen Funktionsaufruf zur Laufzeit die richtige Parameterliste zusammenzusetzen.

Transparente Funktionsaufrufe. CORBA-Objekte können im lokalen Adressraum oder auf einem entfernt liegenden Rechner existieren. Der

Benutzer muß sich dabei nicht um die Lokalisierung des Serverobjektes, um Übertragung von Objektdaten, Objektaktivierung oder Byte-Reihenfolge bei der Übertragung von Daten zwischen verschiedenen Plattformen oder Betriebssystemen kümmern. Diese Belange werden im Hintergrund durch die CORBA-Laufzeitumgebung geregelt.

Polymorphie. Die Strukturen einer CORBA-Umgebung sind Objekte im Sinne des objektorienterten Programmierens. Damit gehen alle Funktionsaufrufe auch an Objekte, die in der für sie charakteristischen Weise darauf reagieren.

Komponenten

Softwarekomponenten sind eine besondere Form von objektorientierter Software. Das Ziel der Komponententechnologie ist es, Softwarekonstrukte zu schaffen, die als abgeschlossene Systeme in beliebiger, "nicht vorhersagbarer" Kombination verwendet und zu größeren Gebilden zusammengesetzt werden können. CORBA unterstützt die Entwicklung von Komponenten, indem es die netzwerktransparente und plattformübergreifende Verwendung von Objekten ermöglicht und zugleich Informationen über die Struktur der Objekte bereitstellt, die zur Laufzeit analysiert werden können.
Damit sich verschiedene Objekte zu einem sinnvollen Gebilde zusammensetzen lassen, müssen sie eine gewisse Menge gemeinsamer Eigenschaften aufweisen. Die Definition gemeinsamer Eigenschaften ist auch für die Interoperabilität zwischen Objekten, also den Datenaustausch und das gegenseitige Aufrufen von Methoden, erforderlich.
Die gemeinsamen Eigenschaften von Objekten werden in Form von gemeinsamen Schnittstellen (engl.: *Common Interface*) implementiert. Die Intention eines *Common Interface* liegt darin, Interoperabilität, Austauschbarkeit, Entkopplung und Wiederverwenbarkeit von Softwaremodulen, die unabhängig voneinander entwickelt wurden, zu unterstützen [AJSK98]. Dieser Ansatz setzt voraus, daß sich für die Funktionalität der beteiligten Applikationen ein "gemeinsamer Nenner" finden läßt, der in dem *Common Interface* abgebildet wird. Gegenüber bilateralen Schnittstellen (siehe Bild 3.5) bietet diese Struktur den Vorteil, daß eine neue Applikation nur das *Common Interface* bereitstellen muß und dann automatisch Zugriff auf alle anderen Applikationen hat, während bei einer bilateralen Struktur die Schnittstellen zu jeder anderen Applikation separat implementiert werden müßten.
Die in einer CORBA-Umgebung definierten Objecte werden auch als *CORBA Business Objects (CBO)* bezeichnet. Bei der Definition eines Anwendungsgebiets bilden sie die Dinge ab, mit denen in den Applikationen gearbeitet wird. Das Konzept von CORBA sieht vor, daß diese Objekte *unabhängig* von einer speziellen Applikation verwendbar sein sollen und auch in einer nicht vorherbestimmten Weise zusammenfügbar sein sollen. Mit diesem Ansatz soll eine maximale Flexi-

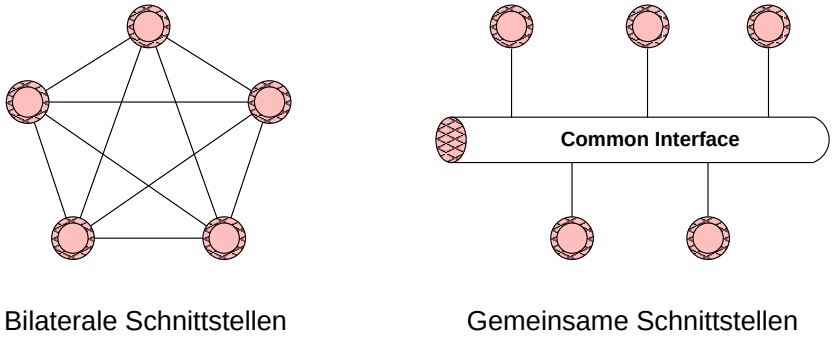


Bild 3.5: Vergleich zwischen bilateralen und gemeinsamen Schittstellen

bilität bei der Verwendung der Objekte erreicht werden.
Die *Business Objects* bestehen aus drei Einzelkomponenten, die unter Nutzung der Basisdienste Kontakt zueinander halten:

Business Object: Das eigentliche *Business Object* überwacht den Speicherbereich, die Metadaten und die Konsistenz des Objekts, steuert gleichzeitigen Zugriff mehrerer Anwender und reagiert auf die Anfragen von Anwendern.

Business Process Objects: Das Verhalten eines Objekts innerhalb eines Prozeßmodells, die darin definierten Zustände und Methoden werden von einem *Business Process Objects* überwacht. Außerdem werden hier die Mechanismen implememtiert, über die sich mehrere Objekte zu einem größeren zusammenfügen lassen.

Business Presentation Objects: Zu einem *Business Object* gehören ein oder mehrere *Business Presentation Objects*, die das Objekt und seine internen Daten in verschiedenen Sichten für den Benutzer sichtbar machen.

Container

Besonderer Bedeutung kommt den Modellen zu, in denen die Komponenten zu komplexen Informationseinheiten zusammengefügt werden. Als Fachbegriff wird hierfür *Container* verwendet. Container müssen die Rahmenbedingungen festlegen, unter denen Objekte zusammengesetzt werden können. Ein typisches Beispiel für einen Container ist ein Dokument, in das verschiedene Objekte, wie Textkomponenten, Bilder, Tabellen und ähnliches eingefügt werden können. Der Container muß in diesem Beispiel dafür sorgen, wieviel Platz den einzelnen Komponenten für die Darstellung zur Verfügung gestellt wird und muß Methoden zum Speichern und Wiederherstellen gemischter Dokumente bereitstellen.

Verteiltes Objektmodell

Während die Komponenten das Zusammenwirken verschiedener Objekte auf Anwendungsebene beschreiben, wird mit CORBA gleichzeitig ein verteiltes Objektmodell auf Systemebene realisiert [Bet94]. In verteilten Objektmodellen ist der Aufruf der Methoden der Objekte nicht meht davon abhängig, ob sie sich auf dem selben Rechner befinden, oder per Netzwerk auf einem fremden Rechner angesprochen werden [Lov97]. Dies wird durch eine strikte Trennung von Schnittstellen und Implementierungen der Objekte realisiert. Die Schnittstellen werden in neutraler Form netzwerkweit zur Verfügung gestellt. Die Laufzeitumgebung von CORBA sorgt für die Weiterleitung von Methodenaufrufen an die zugehörigen Implementierungen. Damit ist die Bindung von Objekten zu einem bestimmten Betriebssystem aufgehoben.

Granularität

Die Granularität einer Schnittstelle ist ein Maß dafür, wie weit die von dieser Schnittstelle angebotenen Objekte in ihre Bestandteile zerlegt werden können. Je nach Clientapplikation können die Anforderungen an die Granularität unterschiedlich sein: ein Stücklistenverwaltungsprogramm benötigt lediglich die Namen und die Anzahl der in einer Baugruppe verwendeten Komponenten, während ein FEM–Programm die genauen Abmessungen der Flächen und Kanten eines Bauteil benötigt. Bei der Integration von Gestaltung und Berechnung kann ein sehr feingranularer Zugriff sowohl auf die Geometrie als auch auf das Rechenmodell erforderlich sein.

Die Common Object Services

Systeme, in denen Objekte aus verteilten Anwendungen verwendet werden, unterscheiden sich von lokalen objektorientierten Systemen. Die lokalen Systeme sind vergleichsweise einfach zu handhaben: die Programme laufen in einem einzelnen Prozeß auf einem einzelnen Rechner und die Schnittstellen der Objekte sind bereits bei der Übersetzung bekannt. Die Betriebssystemumgebung der Objekte ist ebenfalls bei der Übersetzung bekannt und damit fest vorgegeben.
In verteilten Systemen hingegen werden zusätzliche Dienste benötigt, mit deren Hilfe Objekte lokalisiert, verknüpft und transportiert werden können. Die Objekte sind unabhängig von einem bestimmten Prozeß und können auf verschiedenen Betriebssystemen verwendet werden. Es kann vorkommen, daß die Schnittstelle zu einem Objekt erst zur Laufzeit bekannt ist, sodaß Programme dynamisch gebunden werden müssen. Die hierfür erforderlichen Dienste dürfen nicht betriebssystemspezifisch an einen Rechner gebunden sein, sondern müssen als allgemein verfügbare Dienste der objektorientierten Umgebung zur Verfügung stehen [McF98].

Ein großes Spektrum der für verteilte Anwendungen benötigten Methoden ist in den *Common Object Services* definiert. Die Objekte und Methoden dieser Dienste sind per IDL–Definition festgeschrieben und gehören zur Laufzeitbibliothek jeder CORBA-Implementierung. Auf diese Weise wird sichergestellt, daß die Aufrufe für die Erzeugung, Vernichtung, die Benennung, der Transport etc. von Objekten auf allen Rechnerplattformen in gleicher Weise zur Verfügung stehen. Im einzelnen sind folgende Dienste definiert [OHE96]:

Life Cycle Service. In diesem Dienst sind Methoden zum Erzeugen, Kopieren, Verschieben und Löschen von Objekten auf dem *ORB* definiert.

Persistence Service. Hier werden Methoden bereitgestellt, die es erlauben, Objekte dauerhaft in relationalen oder objektorientierten Datenbanken zu speichern.

Naming Service. Mit Hife dieses Dienstes können Objekte mit einem netzwerkübergreifend gültigen Namen verknüpft werden.

Event Service. Über diesen Dienst können Ereignisse, die für die beteiligten Applikationen von Interesse sind, gebündelt und weitergegeben werden. Der *ORB* sorgt für die Weiterleitung der Ereignisinformationen.

Concurrency Control Service. Mit diesem Dienst können gemeinsame Zugriffe auf Objekte überwacht und gesteuert werden. Ein wichtiger Mechanismus beim Simultaneous Engineering.

Transaction Service. Eine einheitliche Schnittstelle für die Übermittlung von Daten zwischen Objekten.

Relationship Service. Dieser Dienst stellt Methoden für die Erzeugung und die Überwachung von zwischen verschiedenen Objekten bestehenden Verknüpfungen bereit.

Externalization Service. Dieser Dienst definiert einen Standard für eine Umwandlung von Objekten in eine serielle Darstellung und umgekehrt.

Query Service. Zugriffe auf verschiedene Datenbanken werden über die in diesem Dienst beschriebenen Schnittstellen gekapselt.

Licensing Service. Die Nutzungshäufigkeit von Objekten kann mit diesem Dienst überwacht werden. Dabei ist die Überwachung für ein einzelnes Objekt, für einen Rechner, für eine Sitzung oder für einen ganzen Bereich möglich.

Properties Service. Mit diesem Dienst können zur Laufzeit Merkmale zu Objekten hinzugefügt werden, indem Merkmalname und ein zugehöriger Wert angegeben werden.

Die allgemeinen Dienste sind zum Teil Bestandteil einer CORBA Laufzeitumgebung und zum Teil Klassen, deren Schnittstelle von den Benutzerobjekten geerbt und implementiert werden muß. Letzteres ist erforderlich, wenn die Implementierung Zugriff auf interne Datenstrukturen einer gekapselten Applikation haben muß. Beispielsweise muß für den Persistence Service ein Algorithmus implementiert werden, der es erlaubt, die gesamte interne Struktur eines Objekts in ein Speichermedium zu schreiben. Dennoch ist die Definition dieses Dienstes sinnvoll, da hiermit eine allgemeine Methode zum Speichern von Objekten geschaffen wurde, deren Implementierung für den Anwender transparent ist.

Common Facilities – Frameworks

Die für einen bestimmten Anwendungsbereich erforderlichen Objekte und deren Methoden sind in den *Common Facilities* definiert. Zusammen mit den eigentlichen Applikationen, den *Application Objects*, bilden sie die Arbeitsumgebung (engl.: Framework), in der Objekte angewendet werden. Typische Anwendungsgebiete sind Textverarbeitung, Bankwesen oder CAx–Anwendungen. Die *Common Facilities* sind noch einmal unterteilt in *Horizontal Common Facilities* und *Vertical Common Facilities*. Dabei bezeichnen die *Horizontal Common Facilities* diejenigen Dienste, die allen Applikationen und Objekten des jeweiligen Anwendungsgebiets gemeinsam zur Verfügung stehen sollen, und die *Vertical Common Facilities* oder auch *Domain Services* [Red96] die Objekte und Methoden einer bestimmten Anwendung.
Die *Common Facilities* sind nicht Bestandteil einer CORBA-Laufzeitumgebung. Vielmehr bilden sie eine Anwendung, die eine funktionierende CORBA–Umgebung voraussetzt. Das zur Zeit am weitesten entwickelte Framework ist das Textverarbeitungssystem *OpenDoc*. Dort sind die in Textdokumenten vorkommenden Objekte, wie Textbausteine, Diagramme, Tabellen oder Bilder als CORBA-Ojekte implementiert und in eine Umgebung eingebettet, die es ermöglicht, auch verteilt vorliegende Dokumente zu bearbeiten, zusammenzufügen, zu speichern oder zu verschieben. Derzeit ist es das Bestreben der OMG, die *Common Facilities* für ausgewählte Anwendungsgebiete ebenfalls mit in den Standard aufzunehmen. Entwürfe existieren unter anderem für den CAx–Bereich und das Gesundheitswesen [OMG95].

Vorteile gegenüber anderen Ansätzen

CORBA bietet gegenüber anderen Ansätzen eine Reihe von Vorteilen. Im einzelnen sind dies folgende Merkmale:

Plattformunabhängigkeit: Diese Eigenschaft von CORBA ermöglicht es, die in der Praxis existierende Vielfalt von Rechnerplattformen zu integrieren. Dies ist ein Vorteil gegenüber plattformspezifischen Ansätzen.

Objektorientiertheit: Durch den objektorientierten Ansatz von CORBA können sowohl die Objekte in den Applikationen, als auch die Dienste und Komponenten der CORBA–Umgebung durch Vererbung zusammengefaßt oder erweitert werden. Von großem Vorteil ist hier die Möglichkeit, mit der Technik der Kapselung auch prozedurale Applikationen in eine objektorientierte Umgebung zu integrieren.

Netzwerkfähigkeit: CORBA ist ein Kommunikationssystem für *verteilte* Objekte. Dies ist ein Vorteil gegenüber lokalen Programmiermodellen. Zudem wird CORBA damit den Forderungen nach verteilten und vernetzten Entwicklungssystemen gerecht.

Framework: Durch die Definition und teilweise Implementierung von Diensten für das Arbeiten mit verteilten Objekten liefert CORBA automatisch ein Infrastruktur für verteilte Systeme. Dies ist ein Vorteil gegenüber rein kommunikationsorientierten Ansätzen.

3.2.3 JAVA als Programmiersprache

Im Zuge der Vernetzung vieler Rechner über das Internet wurde der Bedarf erkannt, eine Programmiersprache zu entwickeln, die ohne Übersetzung auf verschiedenenen Rechnerplattformen lauffähig ist. Von *Sun Microsystems* wurde mit JAVA eine solche Programmiersprache entwickelt [GJS98].

JAVA Virtual Machine

Das Konzept von JAVA besteht darin, daß die Programme nicht direkt ausgeführt werden, sondern von einer Programmumgebung, der *Virtual Machine*, die eine standardisierte Schnittstelle zu den betriebssystemspezifischen Ressourcen wie Speicher, Dateisystem oder Bildschirmdarstellung, zur Verfügung stellt. Die Implementierung der Virtual Machine ist plattformabhängig, wird aber inzwischen oft als Bestandteil des Betriebssystems zusammen mit diesem ausgeliefert. Implementierungen existieren inzwischen für die verschiedenen Windows Versionen, für die gängigen UNIX Versionen und auch für MacOS.

Portabler Programmcode

JAVA ist eine objektorientierte Sprache, die eine vereinfachte Form der Syntax von C++ verwendet. Zusammenhängende Klassen werden in Form von *packages* gebündelt. Der Programmcode wird mit einem Compiler in den JAVA-Bytecode übersetzt. Der Bytecode ist die plattformunabhängige Fassung des Programms und kann über Netzwerke übertragen werden. Während anfänglich der Bytecode von der Virtual Machine interpretiert wurde, wird in den aktuellen Versionen der

Bytecode zur Laufzeit von sogenannten *Just-in-time Compilern* in plattformspezifischen Maschinencode übersetzt.
Zur Sprache gehören ebenfalls zahlreiche Klassen für die Netzwerkprogrammierung, für die Ein- und Ausgabe, für die Programmierung von Nebenläufigkeit sowie eine große Zahl von ebenfalls plattformübergreifend verwendbaren Komponenten einer grafischen Benutzungsoberfläche [Tur99]. Der Zugriff auf Datenbanken ist über eine eigene, zum Standardumfang von JAVA gehörende Schnittstellenbeschreibung (*JDBC Java Database Connectivity*) möglich. Mit den von den Datenbankherstellern bereitgestellten Datenbanktreibern können JAVA Programme auf diese Weise direkt auf netzwerkweit verteilte Datenbanken zugreifen.
JAVA Programme können dabei sowohl als *Applications* als auch in Form von *Applets* erstellt werden. *Applications* sind vergleichbar zu herkömmlichen Programmen: sie haben Zugriff auf alle Systemkomponenten, wie Speicher, Festplatten oder Bildschirm und werden vom Benutzer gestartet und beendet. *Applets* hingegen können Bestandteil eines Internetdokuments sein. Beim Aufruf des Dokuments werden sie in einem Webbrowser ausgeführt. Die Virtual Machine wird in diesem Fall durch den Webbrowser bereitgestellt. Start und Beendigung eines Applets erfolgen durch Aufruf und Verlassen des zugehörigen Dokuments. Die zur Ausführung des Applets benötigten Klassen werden zur Laufzeit vom bereitstellenden Server geladen. Beim Verlassen der Seite oder bei Beendigung des Webbrowsers wird auch das Applet beendet.

JAVA und CORBA

Eine äußerst leistungsfähige Grundlage für die Entwicklung verteilter Appliktionen bildet die Kopplung der Technologien JAVA und CORBA [OH97]. CORBA bietet die Möglichkeit, die Daten und Methoden großer, auch bestehender, Systeme objektorientiert zu kapseln und auch in einem Netzwerk verfügbar zu machen. Mit JAVA können Anwendungsprogramme geschaffen werden, die über CORBA auf diese Daten zugreifen, aber durch Portabilität von JAVA auf nahezu allen Rechnerplattformen lauffähig sind. Auf diese Weise können auch portable Benutzungsoberflächen für Systeme geschaffen werden, die mit Objekten aus den CORBA-Servern arbeiten.
Wie im Abschnitt 3.2.2 beschrieben, müssen bei der Implementierung von Clientanwendungen für CORBA entweder die Schnittstellenbeschreibungen der Serverklassen bei der Übersetzung des Programmcodes vorliegen, oder die Funktionsaufrufe müssen zur Laufzeit über die wesentlichen langsameren dynamischen Schnittstellen erfolgen. Bei der Verwendung von JAVA als Programmiersprache können problemlos Objekte aus verschiedenen Servern zur Laufzeit zusammengefügt werden und trotzdem die schnellen statischen Aufrufe verwendet werden, da die serverspezifischen Zugriffsmethoden als Bestandteil des JAVA Programms mit zum Client übertragen werden. Die Verwendung von CORBA–Objekten ist damit nicht mehr an eine plattformabhängige CORBA–Umgebung gebunden, da

die CORBA–Umgebung gegebenenfalls als JAVA–Applet zusammen mit der eigentlichen Anwendung übertragen werden kann. Bei der Verwendung von Applets wird auf diese Weise ein Zugriff auf die Serverobjekte über das WWW und damit von jedem Rechner im Internet aus möglich.

3.3 Integrierte Systeme

3.3.1 Produktdatenmanagementsysteme

Die Verwaltung von Produktdaten zu bestimmten Konstruktionsständen wird in zunehmendem Maße mit Hilfe von Produktdatenmanagementsystemen (PDM–Systemen) realisiert. Diese Systeme verwalten und archivieren die während der Entwicklung eines Produkts entstehenden Dokumente. Zu den Dokumenten werden Metadaten, wie Bearbeiter, Konstruktionsversion, Freigabestatus sowie Projekt– und Kosteninformationen verwaltet. Zudem können in den Systemen die Abhängigkeiten zwischen verschiedenen Dokumenten abgebildet werden. Auf diese Weise bilden PDM–Systeme die zentrale Datenbasis für größere Entwicklungsprojekte.
Aktuelle Forschungsarbeiten zielen darauf ab, verschiedene PDM–Systeme zu einem firmenübergreifenden Gesamtsystem zu integrieren und ein Produktdatenmanagement auch über räumlich entfernte und verteilte Datenbestände zu realisieren [AGL98].
In Zusammenhang mit PDM–Systemen werden in *Workflow–Systemen* Arbeitsprozesse und die zugehörigen Informationsflüsse abgebildet. In Zusammenarbeit mit einem PDM–System stellt ein Workflow–System je nach Bearbeitungsstand einer Aufgabe die nötigen Informationen und Dokumente zusammen und liefert sie an den Arbeitsplatz des für den nächsten Bearbeitungsschritt zuständigen Sachbearbeiters. Nach Erledigung der Teilaufgabe werden die veränderten oder neu erzeugten Dokumente an das Workflow–System gegeben und dort archiviert oder entsprechend des definierten Prozesses weitergeleitet.
PDM–Systeme und Workflow–Systeme arbeiten dokumentorientiert. Das bedeutet, sie stellen anhand der Metadaten und nach bestimmten Auswahlkriterien die Dokumente zusammen, die der Anwender für die Bearbeitung einer Teilaufgabe benötigt. Die inhaltliche Bearbeitung der Dokumente erfolgt mit den dafür vorgesehenen Programmen, wie CAD–Systemen oder Berechnungssystemen. Dadurch eignen sich diese PDM–Systeme gut für die Realisierung einer losen Kopplung (vergleiche Abschnitt 2.5.1).

3.3.2 Feature Modellier–System FEAMOS

Das System FEAMOS ist ein Modellierungssystem für die rechnerunterstützte Produktentwicklung. FEAMOS steht für **Fea**ture-**Mo**dellier–**S**ystem und ver-

wendet ein eigenes, aus *Features* aufgebautes Datenmodell. Features, werden in diesem Kontext als Konstruktionselemente definiert, die aus Geometrieinformationen, aus semantischer Information oder aus beidem bestehen können [Rie94]. Das System ermöglicht sowohl die Modellierung mit bestehenden Features als auch die Definition neuer Features. Neben den Geometriefeatures sind im System Positionier-, Toleranz- und Normteilfeatures definiert. Die Geometriemodellierung erfolgt unter Verwendung des Modellierkerns ACIS.

Die Erstellung neuer Produkte erfolgt, indem neue, produktspezifische Features zusammen mit bereits bestehenden in einem Modell zusammengefügt werden. Da die Modelle aufgrund der Featuredefinition bereits Semantik enthalten, können diese Informationen direkt für Berechnungen verwendet werden. Die Beschreibung der Modelle erfolgt in einer eigenen, an EXPRESS angelehnten, Sprache PDGL [Rie94].

3.3.3 Objektorientierter Integrationsprozessor OBIP

An der Ruhr–Universität Bochum wurde das Konzept für objektorientierte Integrationsprozessoren für die Integration von Gestaltung und Berechnung entwickelt und implementiert. Ein Integrationsprozessor (kurz: INPRO) ist hier ein Kopplungsbaustein, der zwei verschiedene CA–Systeme miteinander verknüpft. Ein INPRO extrahiert aus den jeweiligen Wirtssystemen die erforderlichen Informationen und fügt diese zu einem eigenen, objektorientierten Datenmodell zusammen, das in einer objektorientierten Datenbank gespeichert wird [WB97]. Dabei muß für jeden zu koppelnden Abschnitt des Produktentstehungsprozesses ein spezifischer INPRO konfiguriert werden. Dies geschieht, indem ein allgemeiner INPRO für die spezifische Kopplungsaufgabe konfiguriert und angepaßt wird. In dem allgemeinen INPRO sind Methoden für die Instanzierung und Speicherung von Konstruktionsobjekten sowie eine Kommunikationsinfrastruktur und eine kontextsensitive Benutzungsoberfläche implementiert.

Eine konkrete Ausprägung eines INPRO's ist ein Integrationsprozessor für die Kopplung von Gestaltung und Berechnung. Diese Komponente enthält spezifische Objekte und Methoden für die Verknüpfung von Geometriemodellierern und Berechnungssystemen. Die Ankopplung der Applikationen geschieht über spezifische Module die das Datenmodell des INPRO's in das Datenmodell der jeweiligen Applikation transformieren. Als Kommunikationssystem wird CORBA verwendet.

Für eine rechnerbasierte Untersützung von Entscheidungsprozessen bei der Modellbildung und für die Bewertung von Berechnungen ist eine Wissensverabeitung in das System integriert [BW98].

3.3.4 Projekt ANICA

Im Projekt ANICA wurde ein System geschaffen, mit dem verschiedene CAD–Systeme direkt miteinander gekoppelt werden können. ANICA steht für **AN**alysis of access **I**nterfaces of various **CA**x systems. Der Bedarf für eine direkte Kopplung entstand aus der Tatsache, daß die bisher praktizierten Kopplungen über Dateien in standardisierten Formaten mit numerischen Ungenauigkeiten und Informationsverlusten durch nicht umkehrbare Konvertierungen behaftet sind [AJSK98]. Der Lösungsansatz besteht im wesentlichen aus der Wahl eines Kommunikationssystems und aus der Definition einer standardisierten Zugriffsschnittstelle.

Kommunikationssystem

Als Kommunikationsinfrastruktur für ANICA wurde CORBA gewählt, weil CORBA das Konzept einer einheitlichen, gemeinsamen Schnittstelle der beteiligten Applikationen unterstützt. Gleichzeitig ist mit CORBA eine direkte Kopplung von Applikationen möglich.

Definition einer Zugriffsschnittstelle

Die Objekte der gemeinsamen Zugriffsschnittstelle wurden aus STEP [ISO94] abgeleitet. In einem als *Architecture Mining* bezeichneten Prozeß wurde untersucht, welche Methoden zu diesen Objekten in verschiedenen CAD–Systemen existieren und welche Parameter diese Methoden benötigen. Im nächsten Schritt wurden die gefundenen Methoden generalisiert und daraus eine allgemeine Schnittstellendefinition für den Zugriff auf Geometriemodelle abgeleitet. Anschließend wurden die definierten Funktionen durch Abbilden der Parameter und Methodennamen auf die CAD–System spezifische Notation implementiert [AJSK98].

Anwendung

Im Teilprojekt ProDMU wurde die Anwendbarkeit des Ansatzes nachgewiesen. Exemplarisch wurden die CAD–Systeme Pro/ENGINEER und CATIA mit den definierten Schnittstellen in eine CORBA Umgebung integriert. Für eine Bauraumuntersuchung wurden dann ein in Pro/ENGINEER und ein in CATIA erzeugtes Bauteil in einer CATIA Sitzung zu einer Baugruppe zusammengefügt. Dabei wurden beide Modelle als Objekt zur Laufzeit aus den jeweiligen Applikationen zusammengefügt. Durch die direkte Kopplung wurden zum einen die oben genannten Nachteile der Dateikonvertierung aufgehoben und zum anderen eine Zeitersparnis von ca. 20 % erreicht [SAKJ98]. Gleichzeitig konnten die bei der Kollisionsprüfung als kritisch erkannten Bereiche in den jeweiligen Originalmodellen markiert werden.

Die Probleme dieser Anwendung lagen in der Erzeugung von Geometrien in den jeweiligen CAD–Systemen. Während der Export von Flächenmodellen aus Pro/ENGINEER heraus unproblematisch ist, können keine reinen Flächenmodelle zu einem bestehenden Parametrikmodell zugefügt werden [SAKJ98].

Datenmanagement

Ein zentrales Problem bei der Verwendung von verteilten Objekten ist die persistente Speicherung. Solange die Speicherung in den systemspezifischen Dateiformaten erfolgt, ist die Verwendung von Objekten immer noch an die Verfügbarkeit des zugehörigen CAD–Systems gebunden. Um ein wirklich portables Objekt zu erhalten, muß entweder die Applikation, in der das Objekt erstellt wurde, zusammen mit dem Objekt gespeichert werden, oder ein einheitliches, für alle Applikationen lesbares Format verwendet werden. Da die in ANICA verwendeten Objekte auf STEP beruhen, wurde ein zentraler Datenserver eingerichtet, der alle Objekte in einer objektorientierten Datenbank speichert. Das konzeptionelle Modell der Datenbank ist durch die Objektklassen von STEP gegeben. Ein der Datenbank vorgelagerter *Object Manager* wandelt die Datenbankeinträge in die von den Applikationen benötigten Objekte und überwacht gleichzeitig die Konsistenz bei parallelen Zugriffen.
Hauptinteressent der Entwicklungen ist die Automobilindustrie, in der zur Zeit noch in verschiedenen Fachabteilungen unterschiedliche CAD–Systeme eingesetzt werden. Für Bauraumuntersuchungen und im Rahmen der Entwicklung des *Digital Mockup (DMU)* sollen die Modelle der verschiedenen Systeme zu einem Gesamtmodell zusammengefügt werden können.

3.3.5 Integrationsumgebung CORSICA

Ein integriertes Ingenieursystem auf der Basis eines verteilten Produktmodells ist in der Integrationsumgebung CORSICA realisiert. CORSICA steht für *CORBA- und STEP- basierende Integrationsumgebung für CAx-Verfahren* und wurde an der Universität-GH Paderborn entwickelt [Hah98]. Das System verwendet ein durchgängiges, STEP-basiertes Modellschema das auf alle beteiligten Applikationen abgebildet wird. Die Systemarchitektur von CORSICA beruht auf im wesentlichen auf die drei folgenden Objektarten:

Intelligente Objekte. Die einzelnen CAx–Systeme werden durch die *intelligente Objekte* gekapselt. Die Schnittstellenbeschreibung der intelligenten Objekte umfaßt Mechanismen für die modelltechnische, die prozeßtechnische und die systemtechnische Kopplung der CAx–Systeme. Das bedeutet, daß die intelligenten Objekte die im integrierten Ingenieursystem existierenden CAE–Werkzeuge mitsamt einer Integrationsschicht repräsentieren, die als Koppelelement die proprietäre Kommunikationsschnittstelle an die standardi-

sierte Schnittstellenspezifikation des integrierten Ingenieuresystems anpaßt [Hah98].

Offene Objekte. Diese Objekte repräsentieren die Bestandteile des Produktmodells, die sich innerhalb der der integrierten CAx–Werkzeuge befinden. Diese Objekte führen das Produktmodell und die in den Werkzeugen auf das Produktmodell definierten Operationen zu einem einheitlichen objektorientierten Modell zusammen [Hah98].

Logische Objekte. Die Verbindung zwischen den verschiedenen Instanzen eines Objektes in den einzelnen CAx–Systemen wird über die *logischen Objekte* hergestellt. Damit werden die auf die intelligenten Objekte verteilten Partialmodelle zu einem verteilten Produktmodell zusammengefügt.

Das System wird um zahlreiche Basisdienste, wie Identifikationsdienste, Systemmanagementdienste und einen Constraintmanager zur Überwachung systemübergreifender Zwänge ergänzt. Mit dem System wurde die Kopplung des CAD–Systems *Euclid3* und eine EMV–System (ein Programm zur Berechnung Elektromagnetischer Verträglichkeit) realisiert.

3.3.6 Konstruktionsanalyse- und Leitsystem KALEIT

Im Rahmen des von der Deutschen Forschungsgemeinschaft geförderten Sonderforschungsbereiches 203 “Rechnerunterstützte Konstruktionsmodelle im Maschinenwesen” wurde ein Konzept für ein System zur durchgängigen Unterstützung des Konstruktionsprozesses erarbeitet, das am Institut für Maschinenkonstruktion -Konstruktionstechnik der TU Berlin in dem System KALEIT umgesetzt wurde. Das System gliedert sich in die vier Hauptmodule *Aufgabenanalyseprozessor*, *Lösungskoordinationsprozessor*, *Konstruktionsleitsystem* und *Speicherbereich* [Fel89]. Für die Speicherung der produktbezogenen Daten wird ein eigenes Datenmodell verwendet. Bei der Durchführung von Entwicklungsaufgaben baut das Modell eines Entwicklungsschrittes auf den im vorherigen Schritt erzeugten Daten auf. Auf diese Weise wird eine durchgängige Nutzung der Daten über alle Entwicklungsschritte erreicht [Bei90].
Das Klären und Präzisieren der Anforderungen wird durch den *Aufgabenanalyseprozessor* unterstützt. Diese Komponente verwendet eine vorgegebene Liste von Hauptmerkmalen und stellt für die verschiedenen Merkmale zahlreiche Assoziationshilfen bereit. In einem Analyseteil wird anhand der Anforderungen überprüft, ob sich in der Datenbasis bereits eine Lösung für diese Aufgabe befindet, die gegebenenfalls herangezogen werden kann [Stü90].
Der *Lösungskoordinationsprozessor* soll den Anwender bei Planung der einzelnen Konstruktionsschritte unterstützen, indem er die verfügbaren Rechnerwerkzeuge, deren erforderliche Eingabedaten und ihre Ergebnisse bereitstellt. Er stellt

damit die Verbindung zwischen den Arbeitsschritten beim Konstruieren und den verfügbaren Rechnerwerkzeugen her.
Der eigentliche Entwicklungsprozeß wird durch das *Konstruktionsleitsystem* unterstützt. Dabei werden die vom Aufgabenanalyseprozessor ermittelten Aufgaben in der erforderlichen Reihenfolge ausgeführt. Das Zusammenwirken von Methoden, Modellen und Daten wird als *Operationsstruktur* bezeichnet. Dabei existieren für verschiedene Branchen unterschiedliche Grundmuster von Operationsstrukturen.
Die im Entwicklungsprozeß benötigten Methoden, Modelle und Prozeßdaten werden im Speicherbereich verwaltet. Damit war in KALEIT eine Form eines integrierten Produktmodells verwirklicht. Neben den Modellinformationen waren in der Operationsstruktur auch die erforderlichen Metainformationen für eine zielgerichtete Anwendung von Methoden enthalten. Die Methoden waren dabei auf das Datenmodell abgestimmt.

3.3.7 Das System mfk

Am Lehrstuhl für Konstruktionstechnik der Universität Erlangen Nürnberg wurde das System **mfk** als Assistenzsystem für die Beschreibung und Analyse von Bauteilen und Baugruppen entwickelt. Das System ist als Aufsatz zu einem kommerziellen CAD–System konzipiert, verwendet aber ein eigenes, semantisches, relationsbasiertes Produktmodell. Für die Visualisierung, für mathematisch geometrische Berechnung und für die Repräsentation der geometrischen Elemente wird die Funktionalität des CAD–Systems verwendet. Ebenso erfolgt die Benutzerinteraktion über eine Erweiterung der Benutzungsschnittstelle des CAD–Systems und damit in der für den Konstrukteur vertrauten Arbeitsumgebung [Löf97].

Produktmodell

Das Produktmodell von mfk besteht aus den drei Grundelementen *Parameter*, *Formelement* und *Beschreibungsmerkmal*, die über *Relationen* miteinander verknüpft sind. Die Relationen dokumentieren dabei die während des Konstruktionsprozesses dynamisch wachsenden Zusammenhänge zwischen den Informationseinheiten [Web92]. Das Produktmodell ist die gemeinsame Datenbasis für den Analyseteil und den Syntheseteil des Systems.

Integration von Berechnungsmethoden

In das System sind verschiedene Berechnungsmethoden eingebunden, die direkt auf der Datenbasis arbeiten. Unter anderem wurde für die Implementierung dieser Methoden ein wissensbasiertes System integriert. Externe Methoden können entweder in dem wissensbasierten System implementiert werden und arbeiten

dann ebenfalls auf der Datenbasis oder sie können Eingangsparameter in Form einer Datei oder aus der Programmierschnittstelle des Systems beziehen.

3.3.8 iViP–Projekt

Das Projekt *Innovative Technologien und Systeme für die integrierte Virtuelle Produktentstehung* (kurz: iViP) ist ein von Industrie und Forschungseinrichtungen unterstütztes Projekt mit dem Ziel verbesserte technische Rahmenbedingungen zur Schaffung technischer Produkte zu erlangen. Im Vordergrund des Projekts steht die Entwicklung und industrielle Einführung innovativer Werkzeuge für die Produktentstehung auf der Basis einer offenen Systemarchitektur [KTA98]. Mit dieser Architektur soll eine Produktentstehungsumgebung geschaffen werden, die eine integrierte Funktionaliät für eine phasenübergreifende Aufgabenbearbeitung zur Verfügung stellt. Im Mittelpunkt der Entwicklung steht der "Digitale Master", der als verbindlicher Informationsträger sämtliche für die Produktentstehung und alle Folgephasen relevanten Daten eines Produkts enthalten soll. Die dem Projekt zugrundeliegende iViP–Architektur sieht die Schaffung einer offenen Integrationsplattform vor, die auch die Einbindung bestehender heterogener Systemwelten unterstützt [iVi99]. Das Projekt ist in die sechs Cluster: Infrastruktur Prozeßmanagement, Infrastruktur Datenmanagement, Innovative Gestaltungswerkzeuge, Aufbau und Validierung virtueller Produkte, Werkzeuge zur virtuellen Fertigungserprobung und Referenzmodelle zur Fertigungserprobung gegliedert.

Systemarchitektur

In den Projekten der iViP–Cluster "Infrastruktur – Prozeßmanagement" und "Infrastruktur – Datenmanagement" wird die softwaretechnische Basis für das Gesamtprojekt erarbeitet. Die Basis ist eine CORBA–basierte Integrationsplattform, die ein Framework für die in den anderen Teilprojekten entstehenden Softwaremodule darstellt (vergleiche Bild 3.6).
Grundlage bilden die Systemdienste für die Anwenderverwaltung, das Management von Anwendersitzungen und die Verwaltung der iViP–Werkzeuge (Komponenten). Die eigentlichen Applikationen sind über objektorientierte Schnittstellenbeschreibungen gekapselt und in das Gesamtsytem integriert. Bei der Implementierung wurde weitgehend auf international etablierte Standards zurückgegriffen. Für die Bedienung des Systems wurde mit dem iViP–Client eine eigene Benutzungsoberfläche geschaffen. Der iViP–Client stellt einen definierten Umfang an Grundfuktionalität zur Verfügung, der durch Ergänzung von anwendungsspezifischen Plug–ins erweitert werden kann [iVi00].
Für die Nutzung von Softwarekomponenten ist das Konzept *Software on demand* vorgesehen. Hiermit können Nutzern Dienste zur Verfügung gestellt werden, die beispielsweise nicht an ihren lokalen iViP–Servern installiert sind. Dabei können

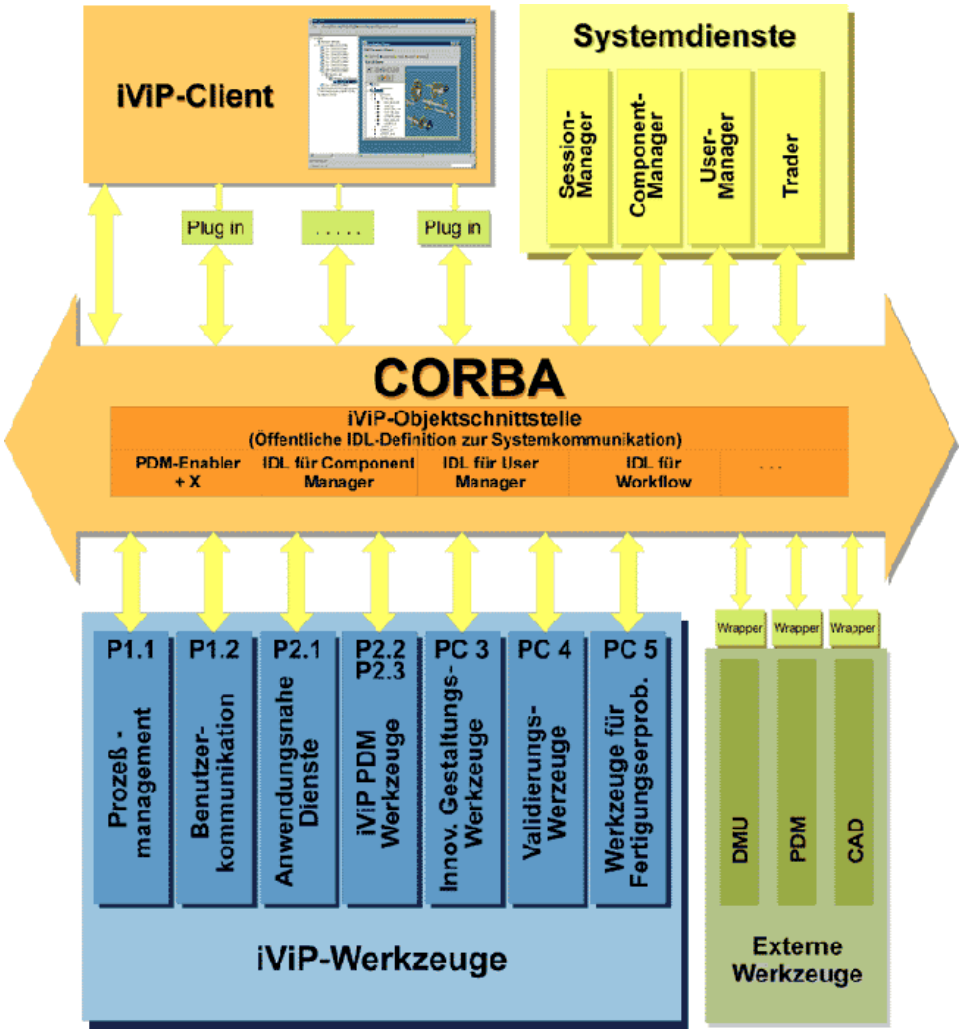


Bild 3.6: iViP–Softwarearchitektur [iVi00]

die Methoden entweder zum Anwender transportiert oder über Remote–Aufrufe auf der Servermaschine verwendet werden.

3.3.9 Virtuelle Berechnungskompetenzzentren

Eine Architektur "virtueller Berechnungskompetenzzentren für die Integration von Gestaltung und Berechnung" wird in [Wöl98] beschrieben. Dort ist ein Berechnungs-Kompetenz-Zentrum definiert als ein *Knotenpunkt im Internet, an dem Berechnungsdienstleistungen einer bestimmten Fachdisziplin mit hoher Fach- und Problemlösungskompetenz* angeboten werden. Die Leistungen eines solchen Kompetenz–Zentrums umfassen dabei die Beratung, die Erstellung differenzierter Berechnungsangebote, sowie die Durchführung und Dokumentation von Berech-

nungen. Die Architektur besteht aus den Komponenten *Broker*, *Berechnungsmanager* und einer *objektorientierten Methodenbank*. Die Kommunikation dieser Komponenten erfolgt über das Internet unter Nutzung standardisierter Protokolle. Die vom Kompetenzzentrum angebotenen Berechnungen können mit Geometriemodellen verknüpft werden, indem eine Verknüpfung der einzelnen Parameter hergestellt wird. Um die Abbildung der Objekte im Rechenmodell zu denen im Geometriemodell zu ermöglichen, wurde ein Engineering–Meta–Modell EMM definiert, das sich an der Baustruktur orientiert. Das Zusammenspiel der einzelnen Komponenten wird im folgenden erläutert:

Broker. Der Broker ist für die Koordinierung der Arbeitsabläufe in der verteilten Umgebung zuständig. Er überwacht und protokolliert die Zugriffe in der verteilten Umgebung. Zudem sorgt er für eine gezielte Informationsbereitstellung bezüglich der von einzelnen Kompetenzzentren angebotenen Dienstleistungen und vermittelt Berechnungsanfragen mit einer bestimmten Thematik an ein geeignetes Kompetenzzentrum weiter.

Berechnungsmanager. Der Zugriff auf die Methoden eines Kompetenzzentrums erfolgt über den Berechnungsmanager. Diese Komponente stellt die Informationen über die angebotenen Methoden zur Verfügung und unterstützt den Anwender bei der Auswahl einer geeigneten Methode mit Hilfe eines wissensbasierten Systems. Je nach Art und Umfang der Methode unterstützt sie den Anwender auch bei der Durchführung der Berechnung, indem sie die Berechnung startet, den Benutzer zur Eingabe zusätzlicher Informationen oder von Geometrieverknüpfungen auffordert und die Ergebnisse dokumentiert.

Objektorientierte Methodenbank. Um eine einheitliche Beschreibung der von einem Kompetenzzentrum angebotenen Methoden zu erhalten, werden Methoden in einer objektorientierten Methodenbank gehalten. Darin werden neben den möglichen Geometrieverknüpfungen einer Methode auch äußere Lasten, Werkstoffe, zusätzliche Bedingungen und assoziierte Bewertungsverfahren verwaltet.

3.3.10 Vergleich der Systeme

In der folgenden Übersichtstabelle werden die Datenmodelle, der Kommunikationsmechanismus und die Offenheit der betrachteten Systeme miteinander verglichen. Ein Maß für die Offenheit ist der Aufwand zur Einbindung neuer Methoden. Die Tabelle zeigt, daß in den meisten Fällen die Berechnungssoftware an die Modellstruktur und an die Plattform des CAD–Systems gebunden ist. Bei einem Wechsel einer der beiden Komponenten muß die Kopplung neu implementiert werden. Eine Integration neuer Methoden würde vereinfacht, wenn sowohl für

System	Datenmodell	Kommunikation	Einbindung neuer Methoden
FEAMOS	eigenes (STEP angelehnt)	—	müssen im System formuliert sein
OBIP	eigenes (objektorient.)	CORBA	Anpassung and das OBIP Datenmodell
ANICA	STEP	CORBA	—
OBIP	eigenes (objektorient.)	CORBA	Anpassung and das OBIP Datenmodell
CORSICA	STEP	CORBA	Implementierung der CORSICA Metaklassen
KALEIT	eigenes	—	Implementierung in CATIA API
mfk	semant. relation. (Pro/E angelehnt)	Socket	auf das Modell abgestimmte API
iViP	eigenes	CORBA	Implementierung der iViP Metaklassen
Virtuelle Ber. CC	Eng. Meta Modell (Pro/E angelehnt)	Socket	auf das Modell abgestimmte API lose Verkn. möglich

Tabelle 3.1: Gegenüberstellung der betrachteten Integrationssysteme

die Zugriffe auf die Geometriedaten, als auch auf die Rechenmodelle einheitliche Schnittstellen existierten. Durch eine Trennung von Schnittstellendefinition und Implementierung, wie sie durch CORBA realisiert werden kann, könnte die Bindung an eine Rechnerplattform aufgehoben werden.
Die Bindung an ein spezielles CAD–System wird durch die Verwendung von STEP aufgelöst, wie sie im Projekt ANICA realisiert wurde. Jedoch beschränkt sich ANICA auf den Austausch von Geometriedaten. Für eine Integration von Berechnungen müssen aber auch nichtgeometrische Daten verarbeitet werden können. Darüberhinaus sind in STEP nur Modelle, aber keine Methoden definiert. Die Anwendung von Berechnungsmethoden als Werkzeug erfordert den programmgesteuerten Zugriff über definierte Funktionen für die Geometrieanalyse, die Geometrieerzeugung und die Geometrievariation.

3.4 Integrationsstrategien

Aus den oben vorgestellten Integrationssystemen und den im vorigen Kapitel beschriebenen Merkmalen lassen sich für die Integration von Berechnungsmethoden in den Gestaltungsprozeß im wesentlichen drei Strategien ableiten:

Integriertes Produktmodell. Bei dieser Strategie wird versucht, bestehende oder neu geschaffene Produktmodelle so zu erweitern, daß alle für eine Berechnung benötigten Daten in dem Modell enthalten sind. Beispiele hierfür sind das applikationsneutrale Produktmodell STEP und die auf bestimmte CAD-Systeme aufbauenden Produktmodelle der Systeme *mfk* oder *FEAMOS*.

Standardisierte Zugriffsmethoden. Ein anderer Schwerpunkt liegt in der Definition standardisierter Zugriffsmethoden auf die bestehenden Modelle. Die Zugriffsmethoden sind dabei häufig an die zugrundeliegenden Modelle gebunden. In einigen Systemen geht daher die Vereinheitlichung der Zugriffsschnittstelle mit einer Erweiterung oder Modifikation des Produktmodells einher. Das Spektrum der verwendeten Zugriffsmethoden reicht dabei von Kommunikationsstandards auf Systemebene, wie IPC oder TCP/IP, über CAD-System spezifische statische oder dynamische Bibliotheken bis hin zur objektorientierten Kapselung und Integration in eine verteilte Arbeitsumgebung. Das Ziel aller Zugriffsmethoden ist es, einen einheitlichen Zugriff auf die Geometriedaten zu erhalten, oft in Zusammenhang mit einer Einbettung in komplexe Systemumgebungen und Schaffung eines Netzwerkzugriffs.

Modelltransformation. Eine weitere Technik ist die Modelltransformation, bei der die Modelle des einen Systems in die Modelle des anderen Systems überführt werden. Modelltransformationen sind in der Regel mit Informationsverlust verbunden und daher nicht umkehrbar. Die Transformationen

können in beide Richtungen erfolgen: ein Geometriemodell kann für die Weiterverarbeitung in einem Berechnungssystem aus der CAD-System internen Darstellung in ein neutrales Austauschformat übersetzt werden und aus dem Ergebnis einer Berechnung kann durch Übersetzen und Zuordnen von Merkmalen ein Geometriemodell erzeugt werden.

In Tabelle 3.2 sind die Hauptmerkmale der drei Strategien dargestellt. Allen Strategien ist gemein, daß die Geometriemodelle die Kernmodelle sind, zu deren Schaffung oder Variation Berechnungsmethoden zur Hilfe genommen werden.

Strategie	**Vorteile**	**Nachteile**
Integriertes Produktmodell	Redundanzfreie, konsistente Datenbasis für alle Applikationen	Einbindung bestehender Systeme aufwendig Komplexes monolithisches Datenmodell
Standardisierte Methoden	Anpassung weiterer Applikationen möglich Flexibler Zugriff auf interne Modelle.	Zusätzlich Modell- und Prozeßkopplung nötig Zusätzlich Konsistenzüberwachung erforderlich
Modelltransformation	Einfache Anbindung bestehender Applikationen	Mehrfache Iterationen führen zu Datenverlust

Tabelle 3.2: Hauptmerkmale der Integrationsstrategien

Im Hinblick auf eine Modellkonsistenz und eine prozeßübergreifende Nutzung von Daten wäre die Anwendung eines integrierten Produktmodells die beste Lösung. Allerdings ist ein solches Modell mit den bestehenden Applikationen nicht handhabbar. Eine Modelltransformation scheidet aus, da wegen des auftretenden Datenverlusts mehrere Iteration einer Operation nicht durchgeführt werden können. Für das weitere Vorgehen werden daher standardisierte Zugriffsmethoden gewählt. Anzustreben ist dabei die Bildung von Applikationsgruppen, von Systemen die für ähnliche Funktionalität gleiche Methoden verwenden.

3.5 Zusammenfassung

Die beschriebenen Integrationslösungen bieten alle die Möglichkeit, in den Gestaltungsprozeß integrierte Berechnungen durchzuführen. Dabei sind alle Lösungen entweder an ein CAD–System oder an eine eigenes Produktmodell gebunden. Die Berechnungsmethoden sind damit nicht ohne Anpassungsaufwand in einer Umgebung aus verschiedenen CAD–Systemen anwendbar. Neue Methoden müssen

immer für das vorhandene System entwickelt oder an die Systemumgebung angepaßt werden. Die Forderung nach Offenheit der Systeme ist damit nicht erfüllt. Durch die Entwicklung von CORBA und JAVA ist inzwischen die plattform- und netzwerkübergreifende Verwendung von Softwarekomponenten möglich geworden. Diese Standards sind aber allgemeine Ansätze, in denen noch keine Modellstrukturen definiert sind. In diesem Bereich steht mit STEP ein umfassender Ansatz zur Verfügung, der aber parallel zu den aktuellen CAD–Systemen entwickelt wird und derzeit nicht in der Lage ist, die Modelle der heutigen CAD–Systeme abzubilden.
Im Projekt ANICA wird eine plattformübergreifende Kopplung zweier CAD–Systeme realisiert, allerdings werden hier nur Geometriedaten ausgetauscht. Für die Integration von Berechnungen müssen die Modelle um die nichtgeometrischen Daten erweitert werden. In der weiteren Arbeit wird daher das Konzept einer objektorientierten Kapselung von Geometriemodellierern für die Integration von Berechnungssoftware erweitert.

4 Anforderungen an das Integrationssystem

Aus den in Kapitel 2 vorgestellten Merkmalen von Gestaltung und Berechnung und denn in Kapitel 3 festgestellten Defiziten lassen sich Anforderungen an ein Integrationssystem ableiten. Generelle Zielsetzung ist dabei, den Entwicklungsprozeß zu verkürzen, indem das rechnerunterstützte Gestalten und Berechnen in einem *integrierten System* zusammengefaßt werden. In dem Integrationssystem sollen *bestehende* Gestaltungs- und Berechnungssysteme miteinander gekoppelt werden können. Die Integration muß dabei auf eine verbesserte Funktionalität des Gesamtsystems abzielen [WN99]. In diesem Kapitel wird eine Liste mit den Anforderungen an das Integrationssystem aufgestellt. Die Gliederung der Anforderungen orientiert sich an den Integrationsebenen des KOMFORCE Arbeitskreises, wonach eine Integration in den drei Ebenen *Modellebene*, *Systemebene* und *Prozeßebene* erfolgen muß [KOM96]. Die vierte Ebene dieses Konzepts, die *Methodenebene* entfällt in diesem Fall, da die zu integrierenden Berechnungsprogramme Gegenstand der Integration sind, und somit durch die anderen drei Ebenen beschrieben werden.

4.1 Anforderungen der Teilbereiche

Allgemeine Problembeschreibung

Aus den allgemeinen Systemen der früheren Kapitel werden Anforderungen für ein konkretes Integrationssystem zusammengestellt. In dem System sollen das parametrische CAD–System Pro/ENGINEER und das Berechnungsprogramm für Schraubenverbindungen BOLT [BOL94] integriert werden. Diese Systeme stehen stellvertretend für aktuelle verwendete Geometriemodellierer und für eine große Zahl von Berechnungsprogrammen. Der Integrationsansatz soll daher so gewählt werden, daß er auf ähnliche Konfigurationen übertragbar ist.

Modelle

Im CAD–System wird ein parametrisches Geometriemodell für die Abbildung der geometrischen Daten verwendet. Geometrische Eigenschaften und Topologiein-

formationen sind über die Programmierschnittstelle des Systems zugänglich.
Das Berechnungprogramm verwendet für die Modelldarstellung benannte Parameter. Andere Objekte als Parameter sind in dem Modell nicht enthalten. Über Steuerparameter können unterschiedliche Modellkonfigurationen und bestimmte Rechenoperationen, wie Auslegung, Nachrechnung oder Optimierung ausgewählt werden.

Programme

Eine Integrationslösung sollte sich in eine vorhandene Hardware- und Softwareumgebung einfügen [Abe97]. Die gewählten Programme existieren zwar beide auf der selben Rechnerplattform und wären daher auch auf einem einzelnen Rechner koppelbar, der Lösungsansatz soll aber auf verschiedene Plattformen übertragbar sein und dabei auch Systeme verschiedener Plattformen zusammenführen können. Dies impliziert, daß der Integrationsansatz auch netzwerkübergreifend arbeiten muß.
Die Programmierschnittstelle des CAD–Systems ist in der Sprache *C* implementiert. Das Berechnungsprogramm ist in *FORTRAN* implementiert.

Prozesse

Die prozeßtechnische Integration muß dafür sorgen, daß sich das Berechnungprogramm an verschiedenen Stellen des Gestaltungsprozesses als Werkzeug einsetzen läßt. Das bedeutet, daß die in Bild 2.6 auf Seite 29 dargestellten Systemwechsel zu einem beliebigen Zeitpunkt möglich sind und sich die beiden Teilprozesse “Gestalten” und “Berechnen” zu einem durchgängigen Gesamtprozeß zusammenfügen lassen. Dabei muß insbesondere der systemübergreifende Informationsaustausch für die Schritte ermöglicht werden, an denen beide Systeme beteiligt sind. Weiterhin muß der Integrationsansatz sicherstellen, daß die innerhalb eines Systems bestehenden Teilprozesse und Iterationen durchgeführt werden können. Voraussetzung hierfür sind neben den oben bereits genannten Anforderungen bezüglich der modelltechnischen und der prozeßtechnischen Integration vereinheitlichte Zugriffsmethoden auf die Programme und deren Modelle. Dazu gehören die Lokalisierung und der Start der Programme sowie die Mechanismen zur Übergabe von Eingangsgrößen und Rückgabewerten. Das Integrationssystem muß es außerdem ermöglichen, die für den aktuell durchgeführten Prozeßschritt erforderliche Sicht auf die Daten des anderen Modells darzustellen. Um dem Benutzer eine einfache Handhabung des Systems zu ermöglichen, sollte die Benutzungsoberfläche für alle Systemkomponenten einheitlich sein.
Allgemein werden an Methoden folgende Forderungen gestellt [FR99] [Abe97]:

- einfache Bedienung
- maximale Flexibiltät

- kurze Zugriffzeit
- hohe Verfügbarkeit
- hoher Integrationsgrad

Verschiedene Berechnungsverfahren sollten auch miteinander kombinierbar sein. Standardberechnungen sollten vom Konstrukteur an seinem Arbeitsplatz, in seiner gewohnten Arbeitsumgebung durchführbar sein [KMM99]. Für eine qualifizierte Methodenauswahl müssen daher Informationen über den Anwendungsbereich, die Gültigkeit und die Aussagegüte einer Methode bereitgestellt werden. In vielen Fällen muß aus einer Berechnung auch eine Dokumentation abgeleitet werden, die gegenüber Kunden als Nachweis dienen kann [Küm99]. Eine automatische Archivierung durchgeführter Berechnungen wird von Abeln in [Abe97] gefordert.

Sicherheit

Bei einem Datenaustausch über ein Netzwerk ergeben sich besondere Anforderungen bezüglich der Sicherheit der Datenübertragung. Hier sind verschiedene Ebenen zu unterscheiden:

Sicherheit gegen unautorisierten Zugriff: Sobald urheberrechtlich geschützte Modelle oder Methoden über das Internet zugänglich gemacht werden, müssen diese durch geeignete Schutzmechanismen vor unberechtigtem Zugriff geschützt werden.

Sicherheit der Datenübertragung: Die bei einem berechtigten Zugriff übertragenen Modelle und Daten dürfen nicht durch dritte verändert werden oder einsehbar sein.

Zugriffskontrolle bei autorisiertem Zugriff: Selbst bei einem berechtigten Zugriff müssen die Zugriffsoperationen durch ein Kontrollsystem überwacht werden. Dabei müssen Zugriffsparameter wie Schreib- und Leseberechtigungen und übertragene Informationstiefe, sowie gleichzeitige Zugriffe mehrerer Benutzer überwacht werden.

Besonderer Bedeutung kommt der Verwendung von urheberrechtlich geschützten Dokumenten dritter zu, beispielsweise den für eine Berechnung verwendeten Normdaten. Diese können zwar in der Berechnung verwendet werden, dürfen aber nicht als Dokument über das Internet zugänglich gemacht werden.

Anforderungen an das Integrationssystem	
F/W	**Anforderungen**
	Modelle
F	Kopplung eines parametrischen Geometriemodellierers mit einem parameterorientierten Berechnungsprogramm.
F	Im CAD–System sind geometrische Eigenschaften und Topologiemerkmale abrufbar.
F	Im CAD–System können Norm– und Standardteile durch Namen identifiziert werden.
F	Das Berechnungsmodell wird durch benannte Parameter beschrieben.
F	Die Auswahl von Norm– und Standardteilen im Berechnungssystem erfolgt über Parameter.
F	Vom Berechnungssystem sind verschiedene Modellkonfigurationen berechenbar. Die Auswahl von Konfigurationen erfolgt über Steuerparameter.
F	Das Geometriemodell ist das Arbeitsmodell des Anwenders. Die Berechnung dient dazu, ein Geometriemodell zu erzeugen oder zu verändern.
	Programme
F	Für eine synchrone Anwendung der Systeme soll eine direkte Kopplung der Programme realisiert werden.
F	Die Kopplung soll unter Nutzung der vorhandenen Programmierschnittstellen erfolgen.
F	Das Integrationssystem muß auf dem Betriebssystem AIX (UNIX) lauffähig sein.
W	Mit dem Integrationssystem sollen weitere verbreitete Betriebsysteme (andere UNIX–Versionen, Windows) angebunden werden können.
F	Die Schnittstelle des CAD–Systems ist in C implementiert.
F	Die Schnittstelle des Berechnungssystems ist in FORTRAN implementiert.
W	Die Integration von in anderen Sprachen implementierten Systemen soll möglich sein.

Tabelle 4.1: Anforderungsliste für das Integrationssystem

Anforderungen an das Integrationssystem	
F/W	**Anforderungen**
	Prozeß
F	Berechnungsmethoden sollen als Werkzeuge beim Gestalten einsetzbar sein.
F	Die Berechnung erfolgt synchron zur Gestaltung. Die Methoden haben eine kurze Laufzeit.
F	Der Geometriemodelierer ist am Arbeitsplatz verfügbar. Das Berechnungsprogramm kann auf einem anderern Rechner im Intranet implementiert sein.
W	Das Berechnungsprogramm kann auf einem Rechner im Internet implementiert sein.
F	Als Werkzeug müssen bei einer Berechnung Geometrieanalysen, Geometrievariation und die Erzeugung von Geometrie möglich sein.
F	Die Arbeitsmodelle müssen in den Originalsystemen sichtbar gemacht werden können.
F	Verknüpfungen zwischen den Modellen müssen visualisiert werden können.
W	Sicherheit gegen unberechtigen Zugriff.
W	Sicherheit der Datenübertragung.
W	Zugriffskontrolle bei berechtigetem Zugriff.
F	Benutzung durch einen Anwender.
W	Erweiterbar auf mehrere Anwender.
F	Die Benutzung darf kein zusätzliches Wissen im Bereich Informationstechnik beim Anwender erfordern.
F	Mit dem System sollen Auslegungsrechnungen, Nachweisrechnungen und Optimierungen möglich sein.

Tabelle 4.2: Anforderungsliste für das Integrationssystem (Fortsetzung)

4.2 Anforderungsliste

Die Anforderungen sind in einer Anforderungsliste (in Anlehnung an [PB97]) auf den Seiten 78– 79 dargestellt. In der Liste werden Forderungen (F) und Wünsche (W) unterschieden.

4.3 Zusammenfassung

Der Integrationsansatz muß eine Integration in drei Hauptbereichen ermöglichen (vergleiche Bild 4.1):

Modell: Die in den Programmen existierenden Datenmodelle müssen in einem integrierten Modell zusammengeführt werden. Da die Modelle eine unterschiedliche Semantik aufweisen, müssen über zusätzliche Informationen Verknüpfungen zwischen den Modellen hergestellt werden.

System: Da die bestehenden Programme auf verschiedenen Rechnerplattformen und Betriebssystemen existieren können, muß der Integrationsansatz plattformübergreifend und netzwerkfähig sein. Um die durchgeführten Berechnungen reproduzierbar zu machen, müssen die Modelle beider Systeme und die Verknüpfungen zwischen den Modellen speicherbar sein.

Prozeß: Um eine Methode als Werkzeug in einem Gestaltungssystem nutzen zu können, muß sich die Anwendung einer Berechnungsmethode in die bestehenden Anwenderprozeduren des Gestaltungssystems einfügen lassen. Da die Applikationen über unterschiedliche Anwenderprozeduren verfügen, müssen die Abläufe der beiden Systeme durch zusätzliche Softwarekomponenten aufeinander abgestimmt werden.

Die Hauptmerkmale der zu verbindenden Systeme sind in Bild 4.1 dargestellt.

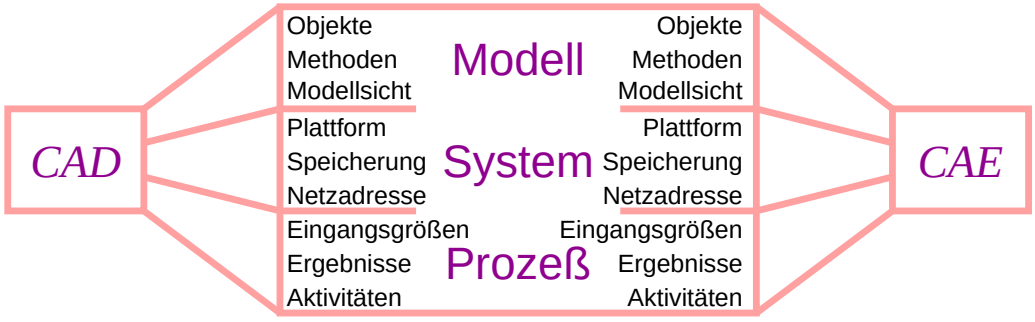


Bild 4.1: Merkmale der zu integrierenden Systeme

5 Integrationskonzept für CAx–Anwendungen auf der Basis von CORBA

In diesem Kapitel werden mögliche Bestandteile und Strukturen eines CORBA–basierten Netzwerkes für CAx–Anwendungen vorgestellt. In der Einführung werden die Merkmale einer CORBA–Umgebung zusammengefaßt und die Situation beschrieben, die sich bei der Anwendung von CORBA für die Integration von Gestaltung und Berechnung ergibt. In einem Überblick werden die Aktivitäten der *OMG* auf diesem Gebiet dargestellt. Anschließend werden mögliche Strukturen für die modelltechnische und die systemtechnische Kopplung bestehender CAx–Systeme vorgestellt. Abschließend werden zwei Integrationansätze präsentiert.

5.1 Ausgangssituation

Mit der durch die *OMG* definierten Architektur (vgl. Bild 3.4 auf Seite 52) ist die Grundstruktur einer CORBA–basierten Arbeitsumgebung bereits vorgegeben: verschiedene Applikationen sind über einen gemeinsamen Kommunikationskanal miteinander verbunden und haben dabei Zugriff auf eine Reihe allgemeiner und anwendungsgebietspezifischer Dienste. CORBA ermöglicht die netzwerktransparente und plattformübergreifende Verwendung von Objekten, die über eine neutrale Schnittstellenbeschreibung bekannt gegeben werden und durch die Applikationen zur Verfügung gestellt werden müssen.

Die Integrationsaufgabe besteht darin, bestehende Softwaresysteme zu einem Gesamtsystem zu kombinieren. Sind die Applikationen durch Spezifikation und Implementierung einer Schnittstelle in ein CORBA–System integriert, wird die Interoperabilität auf Systemebene durch CORBA sichergestellt. Die Interoperabilität auf einer semantischen Ebene ist aber durch CORBA nicht vorgegeben. Dafür müssen die betrachteten Applikationen, ihr Verhältnis zueinander, die von ihnen bereitgestellten Objekte und die durchführbaren Operationen definiert und implementiert werden.

5.2 Stand der Standardisierung

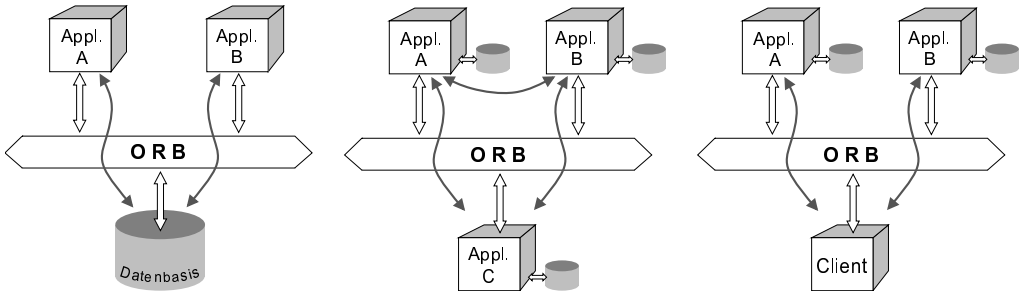
Zusammen mit der *Object Management Architecture* werden von der OMG für verschiedene Anwendungsgebiete Frameworks, also vordefinierte Systemumgebungen, zusammengestellt und standardisiert. Der Stand der Entwicklungen ist für die einzelnen Anwendungsgebiete sehr unterschiedlich. Während für das Anwendungsgebiet *Textverarbeitung* die Entwicklung sehr weit fortgeschritten ist und mit OpenDoc [OHE96] bereits eine Implementiertung vorliegt, sind die Standards für weitere Anwendungsgebiete noch in Bearbeitung. Eine Beschreibung der zu einem Anwendungsgebiet gehörenden *Common Facilities*, also der für dieses Gebiet typischen Objekte und Methoden, wird von der OMG im Dokument *Common Facilities Architecture* [OMG95] veröffentlicht.

Der Bereich Konstruktion und Fertigung (engl. Titel: *Manufacturing Facility*) beschreibt die rechnergestützte Integration von Fertigungsaufgaben und Einrichtungen und deren Zusammenwirken mit weiteren Unternehmensbereichen. Als Teilbereiche werden Produktentwicklung, Prozeßsteuerung, Qualitätsmanagement, CAD, Vertrieb, Sicherheit und Versuchsfeld genannt. Das Ziel der Integration ist es, die bisherigen starren, monolithischen Systeme durch modulare, flexibel an Prozeßänderungen anpaßbare Komponenten zu ersetzen. Von besonderer Bedeutung sind dabei die in der Praxis vorliegende geographische Verteilung der Systemkomponenten und das weite Feld der am Entwicklungsprozeß beteiligten Applikationen. Daraus resultieren weitere Anforderungen an die applikationsübergreifenden Dienste, die insbesondere die dynamische Konfigurierbarkeit von Komponenten, ein Dokumentationssystem zur nachvollziehbaren Protokollierung sämtlicher Aktivitäten im Zusammenhang mit den Produktdaten und den Zugriff auf die Produktdaten betreffen [OMG95]. Der Ansatz ist damit weit umfassender, als es für die Integration von Gestaltung und Berechnung erforderlich wäre.

Die Arbeitsgruppe hat ihre Tätigkeit erst vor kurzem aufgenommen, sodaß noch keine Ergebnisse vorliegen. Lediglich im Bereich PDM-Integration liegen schon Ergebnisse vor. Mit dem Dokument *PDM Enablers* [OMG98] liegt bereits die überarbeitete Fassung eines Vorschlags für eine Spezifikation der Funktionalität eines CORBA basierten PDM-Systems vor. Die Spezifikation sieht eine entwicklungsprozeßbezogene Verwaltung von in Baustrukturen geordneten Bauteilen und Baugruppen vor. Die Funktionalität umfaßt die für PDM-Systeme üblichen Operationen, wie Komponenten erzeugen und finden, Versionsionierung, Änderungsmanagement und Konfigurationsmanagement. Das System arbeitet mit in *Dokumenten* gespeicherten Modellen, denen über *Attribute* zusätzliche Eigenschaften zugeordnet werden können. Die inhaltliche Beschreibung der Objekte soll mit dem durch STEP [ISO94] definierten Format erfolgen.

5.3 Client–Server–Strukturen

Das Zusammenwirken verschiedener Applikationen in einer verteilten Umgebung muß durch klare Zuständigkeiten und Abhängigkeiten organisiert werden. Üblich sind hier Client–Server– Strukturen, in denen eine Applikation Daten und Methoden als Server zur Verfügung stellt und die andere als Client mit diesen Daten arbeitet oder die Methoden in Anspruch nimmt. Im nachfolgenden Text werden mögliche Anordnungen von Client- und Serverapplikationen für einen integrierten Entwicklungsarbeitsplatz vorgestellt.



Zentraler Datenserver | Applikationen als Client und Server | Externer Client

Bild 5.1: Client–Server–Strukturen für die Integration von Gestaltung und Berechnung

5.3.1 Zentraler Datenserver

Sämtliche Produktdaten werden in einer zentralen Datenbank verwaltet (vergleiche Bild 5.1 links). Ein Datenbankserver stellt den als Client arbeitenden Applikationen das gesamte Datenmodell oder Auszüge des Modells zur Verfügung. Die Applikationen arbeiten mit den bereitgestellten Daten und übertragen die geänderten Modelle zurück auf den Server. Diese Struktur bietet zwei wichtige Vorteile: Die strukturierte Speicherung und Bereitstellung von Produktinformationen und die Verfügbarkeit von Verfahren zur Steuerung des Zugriffs auf Produktinformationen [GAP93]. Damit können Inkonsistenzen leichter überwacht werden und parallele Zugriffe, wie sie beim Simultaneous Engineering erfolgen, gesteuert werden. Die Daten des zentralen Servers müssen von den Applikationen in die internen Darstellungen umgewandelt werden. Das Produktmodell des Datenbankservers muß in der Lage sein, sämtliche produktbezogenen Daten abzubilden.
Eine mögliche Implementierung dieser Struktur wird in [AK98] beschrieben (vergleiche Abschnitt 3.3.4).

5.3.2 Applikationen als Server

Eine andere Struktur ergibt sich, wenn die beteiligten Applikationen als Server für die in ihnen bearbeiteten Modelle betrachtet werden. Danach würden alle Geometriedaten in einem "Geometrieserver" gehalten und entsprechend Anforderungen in einem "Anforderungsserver" usw. Ein Geometrieserver stellt alle geometrischen Daten zur Verfügung und bietet Methoden zum Analysieren und Variieren der Geometrie an. Bei dieser Struktur entsteht ein verteiltes Datenmodell mit den entsprechenden Konsequenzen für die Konsistenzüberwachung und Datenhaltung.
Die Verknüpfung der Modelle ist auf zwei Weisen möglich:

Applikationen sind gleichzeitig Client. Ein Server kann gleichzeitig Client eines anderen Server sein. Beispielsweise kann ein Geometrieserver Daten aus der Anforderungsliste beziehen. Bei mehreren Servern entstehen schnell komplexe Abhängigkeiten zwischen den verschiedenen Partialmodellen mit hohen Anforderungen bezüglich der Interoperabilität der Objekte und der Konsistenzüberwachung.

Externe Clients. Eine klare Trennung der Partialmodelle bleibt erhalten, wenn die Applikationen nur als Server für externe Clientprogramme arbeiten. Ein externes Clientprogramm bezieht Daten aus verschiedenen Partialmodellen und fügt diese zu einem integrierten Modell zusammen. Änderungen in einem der Partialmodelle werden auf Anforderung des Clients vorgenommen und im Datenmodell der zugehörigen Applikation gespeichert. Die Anforderungen an die Interoperabilität der Objekte sind bei dieser Konfiguration geringer, da die Verknüpfung über den Client erfolgt. Demzufolge ist der Aufwand für die Implementierung des Client wesentlich höher, da durch den Client zusätzlich die Konsistenz der Modelle überwacht werden muß.

5.3.3 Client–Server–Strukturen mit CORBA

Aus den herkömmlichen 2-Ebenen Client-Server Strukturen, bei denen ein Client direkt mit dem zugehörigen Server in Verbindung steht, wird bei der Verwendung von CORBA eine drei-Ebenen Client-Server Struktur (vgl. Bild 5.2). Die Client-Anwendung, in der die Objekte verwendet und zusammengefügt werden, bildet die erste Ebene. Üblicherweise werden in dieser Ebene auch die sichtbaren Bestandteile der Objekte präsentiert. In der zweiten Ebene liegen die Implementierungen der Objekte, eingebettet in eine CORBA-Laufzeitumgebung. In dieser Ebene werden die vom Client kommenden Aufrufe an die entsprechenden Server weitergeleitet und dort ausgeführt. Die CORBA-Laufzeitumgebung stellt die Kommunikationsinfrastruktur zur Verfügung und verdeckt die internen Datenstrukturen der Datenserver, indem sie nur die in der Schnittstellendefinition festgelegten Objekte und Methoden bekannt gibt. Die dritte Ebene bilden die

Datenbasen für die Objekte der zweiten Ebene. Dabei können eigens dafür eingerichtete Datenbanken genauso zum Einsatz kommen, wie existierende Programme (engl.: legacy code), die über IDL gekapselt wurden, oder CORBA-konforme Serveranwendungen [OH97].

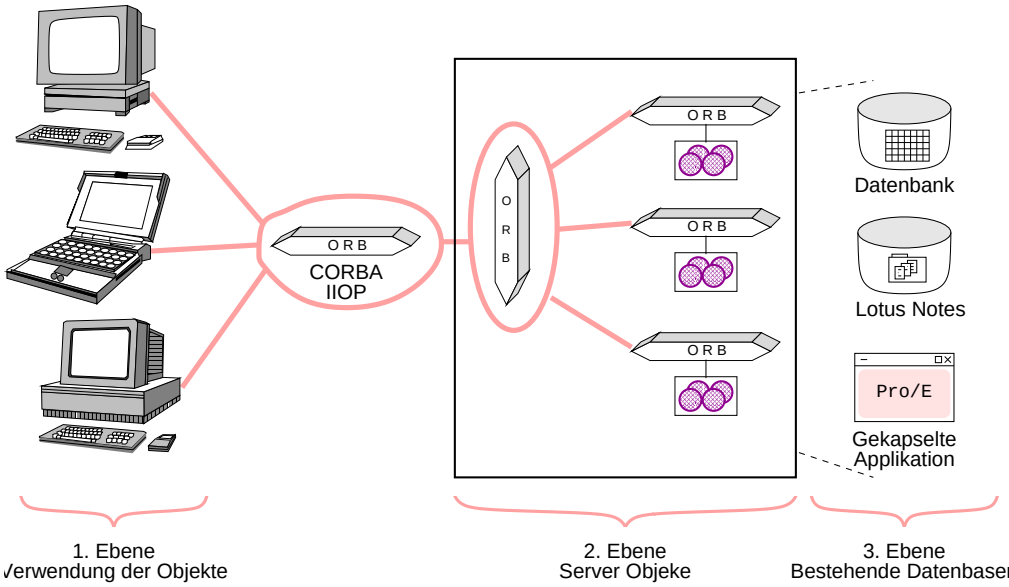


Bild 5.2: 3-Ebenen Client-Server Struktur mit CORBA (in Anlehnung an [OH97])

Mit CORBA ist es möglich, die drei Bereiche Präsentationslogik, Anwendungslogik und Datenhaltung auf verschiedene Rechner, plattformübergreifend, zu verteilen [Tur99]. Da CORBA in der Mitte zwischen den Datenbasen und den Client-Anwendungen steht, spricht man auch von *Middleware*.
Die Integration von bestehenden CAD–Systemen in eine Client-Server Struktur bringt einige Probleme mit sich, da die existierenden Systeme weder als Client- noch als Server ausgelegt sind (vgl. Abschnitt 2.5). Die Systeme vereinen alle drei Ebenen in einer einzigen Anwendung.

5.3.4 Bewertung der Strukturen

Für eine Integration von Gestaltungs- und Berechnungssoftware kommen grundsätzlich alle vorgestellten Strukturen in Frage. Die Ansätze erfordern aber unterschiedlich viel Aufwand bei der Implementierung der Clientanwendungen und der Serverkomponenten. Sie unterscheiden sich ebenfalls darin, wie sich die Datenmodelle bestehender Softwaresysteme integrieren lassen und wie die Konsistenz zwischen den verschiedenen Datenbeständen überwacht werden kann. Tabelle 5.1 zeigt einen Überblick über die Merkmale der vorgestellten Strukturen.

Merkmal	Zentraler Datenserver	Applikationen. als Client + Server	externer Client
Aufwand serverseitig	hoch	hoch	niedrig
Aufwand clientseitig	mittel	hoch	hoch
Konsistenz-überwachung	einfach	aufwendig	mittel
Integration bestehender Datenstrukt.	schwierig	schwierig	einfach
Erweiterbarkeit	einfach	einfach	schwierig

Tabelle 5.1: Vergleich der verschiedenen Client-Server Strukturen

Die Verwendung eines zentralen Datenservers ermöglicht eine einfache Überwachung der Konsistenz und ist leicht erweiterbar. Allerdings muß für die zentrale Speicherung ein Datenmodell bestehen, das in der Lage ist alle entstehenden Informationen abzubilden und das in die Datenformate in den Applikationen umgewandelt werden kann. Dies ist insbesondere für die Integration bestehender Softwaresysteme eine kaum lösbare Aufgabe. Eine sinnvolle Variante, bei der die Modelle der bestehenden Applikationen als Basis genommen werden, ist daher die Zusammenführung der Modelle in einem externen Client. Entsprechend ist Implementierung des Clients wesentlich aufwendiger und, da in der Anwendung zwei spezielle Server zusammengeführt werden, schwieriger erweiterbar. Die Struktur, in der die Applikationen gleichzeitig als Client und Server arbeiten, erfordert, daß, wie beim zentralen Datenserver, allen Anwendungen ein einheitliches Datenmodell zugrunde liegt. Darüberhinaus erfordert sie erheblichen Aufwand in der Konsistenzüberwachung.

5.4 Produktmodellstrukturen

Für eine durchgängige Rechnerunterstützung des Entwicklungsprozesses muß ein System geschaffen werden, das in der Lage ist sämtliche anfallenden Daten zu speichern und den nachfolgenden Prozeßschritten zugänglich zu machen. Im Sinne der objektorientierten Programmierung muß ein *konzeptionelles Modell* für die Abbildung der in den verschiedenen Phasen benötigten Daten geschaffen werden [Boo94]. Die konzeptionellen Modelle werden in diesem Kontext als *Produktinformationsmodelle* bezeichnet.

5.4.1 Partialmodelle in den Applikationen

In den zu koppelnden Applikationen existieren bereits Modelle für die rechnerinterne Abbildung der Objekte. Dies sind seitens der CAD–Systeme geometrieorientierte Modelle und seitens Berechnungssysteme parameterorientierte Modelle, in denen geometrische und nichtgeometrische Informationen enthalten sind (vgl. Kapitel 2). Bei der Integration müssen die Objekte aus den verschiedenen Applikationen zusammengeführt und miteinander verknüpft werden. Grundsätzlich sind folgende Kombinationen von Objekten denkbar:

Gleiche Objekte in allen Applikationen. Die einfachste Form der Kopplung ergibt sich, wenn in beiden Applikationen die gleichen Objekte mit den selben Methoden implementiert sind. In einer solchen Umgebung können die Objekte der einen Applikation direkt in der anderen verwendet werden, ohne daß die Objekte über Abbildungsvorschriften miteinander verknüpft werden müssen. Ein Beispiel hierfür ist die in [AJSK98] beschriebene Kopplung der CAD–Systeme Pro/ENGINEER und CATIA.

Verschiedene Objekte. Der typische Fall bei der Integration bestehender Systeme ist der, daß von den Applikationen unterschiedliche Objekte zur Verfügung gestellt werden. Die verschiedenen Modelle ergeben von sich aus noch kein zusammenhängendes Produktmodell. Vielmehr müssen die Objekte aus den verschiedenen Modellen mit Hilfe von zusätzlichen Informationen zu einem Produktmodell zusammengefügt werden.

Interoperiernde Objekte. Anzustreben ist eine Systemumgebung, in der Objekte einer Applikation in einer anderen verwendet werden können. Dieser Ansatz spiegelt das natürliche Vorgehen bei der Entwicklung wieder, wonach für die Bearbeitung einer Aufgabe Informationen verschiedener Teilbereiche zusammengeführt werden. Als Beispiel sei die in Kapitel 2 beschriebene Beziehung zwischen Ersatzmodellen für die Berechnung und Geometriemodellen genannt, wonach die Geometrie Bestandteil des Ersatzmodells ist. In diesem Fall müssen die Objekte so definiert werden, daß sie untereinander Daten austauschen können und sich zu einem Gesamtmodell ergänzen.

Zentrales Modell und Methodenserver. Denkbar ist auch die Verwendung eines zentralen Modells, aus dem spezielle Sichten für die Weiterverarbeitung in den Applikationen abgeleitet werden. In den Applikationen werden die Modelle analysiert oder verändert und anschließend in das zentrale Modell zurückgeschrieben. In einer solchen Umgebung könnten unabhängige Softwareanbieter (engl.: *Independent Software Vendors ISV*) universell einsetzbare Methoden entwickeln und auf Methodenservern zur Verfügung stellen. Die Abbildung von Daten eines zentralen Modells auf die rechnerinternen Modelle der Applikationen bereitet dabei noch erhebliche Schwierigkeiten,

da dieser Schritt mit den zur Zeit existierenden neutralen Modellen noch mit einer verlustbehafteten und damit nicht umkehrbaren Modelltransformation verbunden ist [Har97].

Für die bestehende Integrationsaufgabe müssen in jedem Fall verschiedene Objekte zusammengeführt werden. Die Art und Weise, wie die internen Modelle der Applikationen über die Schnittstellen zugänglich gemacht werden, bestimmt die Struktur des Gesamtmodells und die Art der Speicherung.

5.4.2 Semantische Verknüpfung von Objekten

Die Semantik von Softwareobjekten wird durch ihre Attribute und die von von ihnen bereitgestellten Methoden erzeugt [OHE96]. Da die Attribute und Methoden von CORBA–Objekten über die Schnittstellenbeschreibungen festgelegt werden, kann durch eine entsprechende Formulierung der Schnittstellenbeschreibung die Semantik einer Applikation verändert werden.
Damit sich Objekte verknüpfen lassen, müssen sie einer gewissen gemeinsamen Semantik unterliegen. Die minimale gemeinsame Semantik von Objekten in CORBA–Umgebung ist dadurch gegeben, daß alle Objekte Eigenschaften von einem allgemeinen *CORBA-Objekt* erben, und damit innerhalb der Umgebung ansprechbar und identifizierbar sind. Zusätzliche Semantik kann den Objekten durch die Attribute und Methoden gegeben werden, die in der Schnittstellenbeschreibung definiert und im zugehörigen Server implementiert sind. Die Semantik der Objekte ist für die modelltechnische und die prozeßtechnische Kopplung von großer Bedeutung, da über sie das Verhalten der Objekte und die Zugriffsmöglichkeiten definiert werden.
Das Problem bei der Kopplung bestehender Applikationen ist, daß jedes System seine eigene Semantik für die Darstellung der internen Modelle verwendet. Im einfachsten Fall kann ein bestehendes System direkt an eine vorgegebene Semantik angepaßt werden, wenn die Objekte und Methoden der Schnittstellenbeschreibung sinngemäß mit denen des Systems übereinstimmen. In diesem Fall können die Objekte und Methoden direkt aufeinander abgebildet werden. Für die Implementierung einer gewünschten Semantik müssen dann gegebenenfalls Datenstrukturen oder Funktionsaufrufe neu zusammengestellt werden. Einen erheblichen Aufwand bedeutet die Schaffung einer Semantik in Systemen, die keine oder nur eine einfache Semantik beherrschen. In diesem Fall muß die Semantik durch zusätzliche Informationen in der Schnittstellenimplementierung erzeugt werden.

5.4.3 Definition eines Strukturmodells

Aus den in Kapitel 2 beschriebenen Merkmalen von Geometriemodellen und Rechenmodellen wird ein Strukturmodell für integrierte Berechnungen definiert. Das

Modell ist als Referenzmodell zu verstehen, das neben den Informationen aus den beiden Teilmodellen die nötigen Verknüpfungsinformationen enthält. Im folgenden Text werden die Bestandteile eines Strukturmodells und die möglichen Relationen zwischen den Bestandteilen definiert. Die Gliederung des Modells orientiert sich an der Baustruktur. Eigenschaften von Baustrukturelementen werden mit Hilfe von benannten Parametern definiert. Über Bedingungen und Relationen werden die Beziehungen der Elemente untereinander formuliert. Über die Bedingungen können die Objekte in den Applikationen identifiziert werden.
Je nachdem, wie die Applikationen gekapselt werden, muß das Strukturmodell in einem verbindenden Client oder in allen Applikationen implementiert werden. Im folgenden Text werden die Bestandteile eines Strukturmodells erläutert, wie es für die Berechnung von Schraubenverbindungen erforderlich ist.

Baustruktur

Das Strukturmodell orientiert sich an der Baustruktur der zu berechnenden Verbindung. Eine Baustruktur wird hier definiert als eine Reihe von Bauteilen und Baugruppen, die über geometrische Relationen miteinander verknüpft sind und an geometrischen Orten, den *Bezugselementen*, ausgerichtet werden. Zu einem Baustrukturelement können verschiedene Repräsentationen existieren. Mögliche Kombinationen von Baustrukturelementen werden durch *erforderliche*, *alternative* oder *optionale* Elemente gebildet. Die einzelnen Elemente haben Merkmale, die über benannte Parameter beschrieben werden. Die Zuordnung von Parametern zu Elementen für verschiedene Formen von Berechnungen werden über *Bedingungen* und *Relationen* formuliert. Eine symbolische Darstellung des Strukturmodells enthält Bild 5.3. Die Beziehungen zwischen den Elementen des Strukturmodells sind in Bild 5.4 dargestellt.

Bezugselemente

Für die Identifizierung bestimmter Wirkorte werden im Modell geometrische *Bezugselemente*, wie Achsen, Ebenen oder Punkte definiert. An einem Grundgerüst, bestehend aus in allen Verbindungstypen vorkommenden Bezugselementen *Schraubenachse* und *Trennfugenfläche*, werden die möglichen Baustrukturelemente (Komponenten) und weitere Bezugselemente mit alternativen und optionalen Elementen ausgerichtet. Merkmale von Komponenten werden durch benannte Parameter definiert. Zu jeder Komponente werden Listen der von ihr definierten Parameter und zusätzlichen Bezugselemente gehalten.

Parameter

Die produktbeschreibenden Merkmale werden durch benannte Parameter definiert. Die Parameter sind entweder Baustrukturelementen oder Bezugselementen zugeordnet. Zur einfachen Lokalisierung sind die Parameter nochmals in die

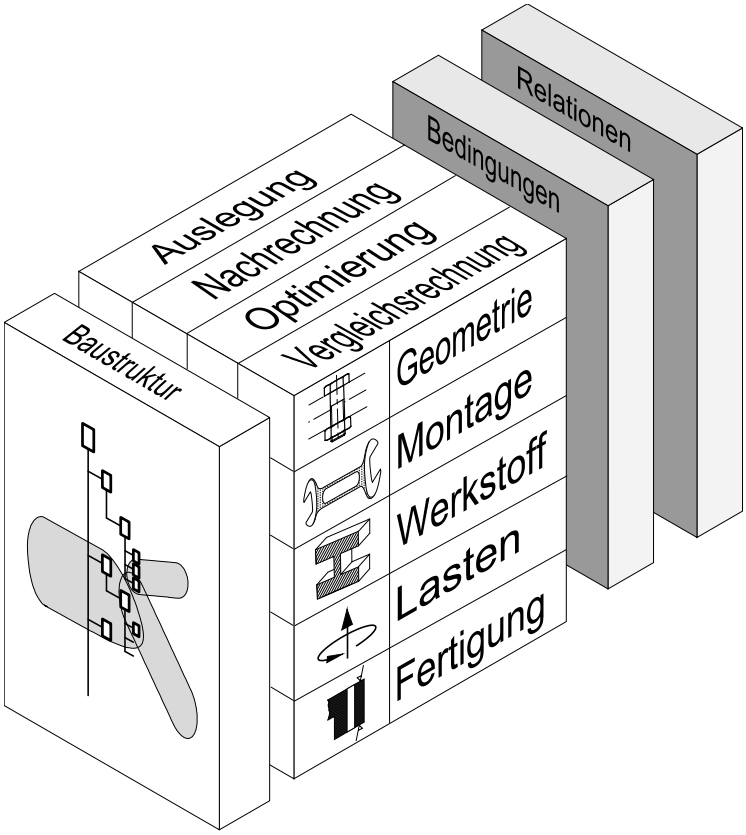


Bild 5.3: Bestandteile des Strukturmodells für integrierte Berechnungen

Merkmalsgruppen *Geometrie*, *Montage*, *Werkstoff*, *Lasten* und *Sicherheit* unterteilt.

Bedingungen und Relationen

Mit Hilfe von Bedingungen und Relationen können Verknüpfungen zwischen verschiedenen Bestandteilen des Modells hergestellt werden. Relationen bestehen zwischen zwei Elementen und können für die Identifizierung von Objekten und die Abbildung der Struktur verwendet werden. Über Bedingungen können Einschränkungen der Modellkonfiguration und Regeln für die Konsistenzüberwachung formuliert werden.

Prozeßinformationen

Parallel zu der Baustruktur und zu den Parametern werden im Modell Informationen zur Einordnung des Modells in den Entwicklungsprozeß gespeichert. In einer Liste sind die möglichen Verwendungsarten einer Methode, wie *Dimensionieren*, *Festigkeitsnachweis* oder *Optimierung* angegeben. In einem Prozeßobjekt

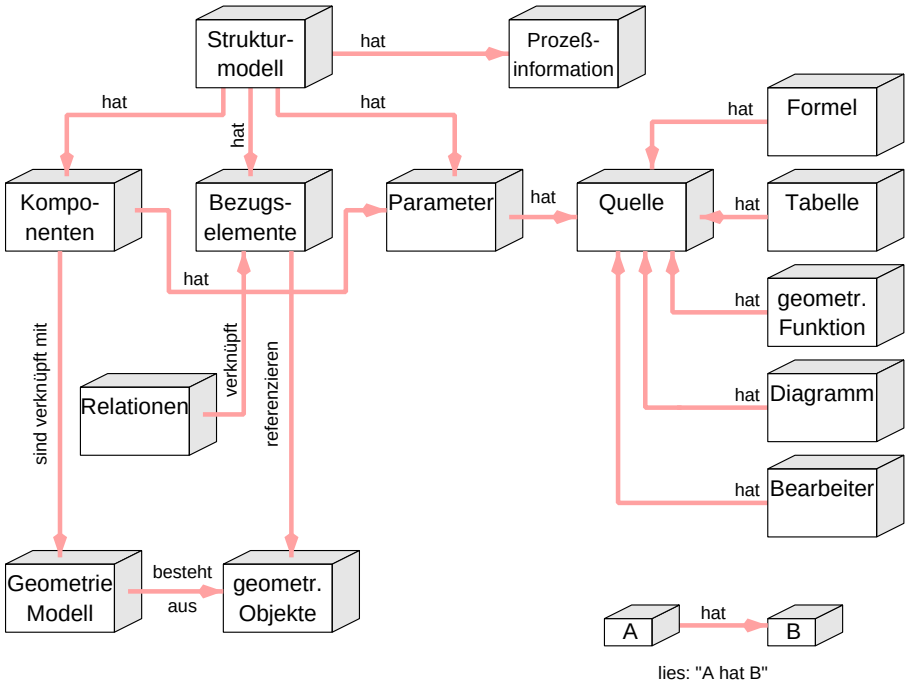


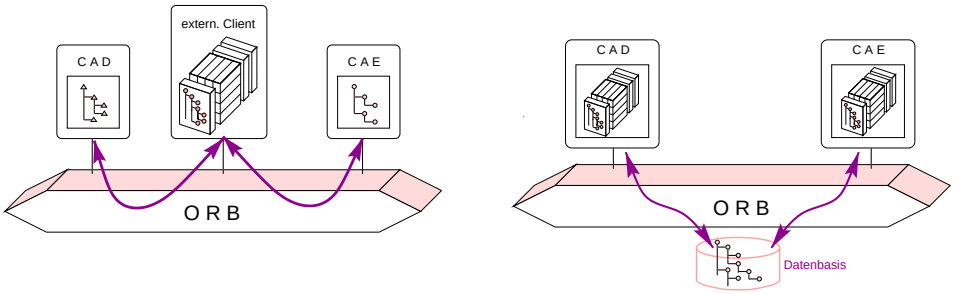
Bild 5.4: Entity-Relationship Darstellung des Strukturmodells

wird angegeben, bei welchem Prozeßschritt ein Parameter Eingangsgröße und bei welchem er Ergebnis ist.

5.5 Kapselung der Applikationen

Mit der Formulierung einer neutralen Schnittstelle wird die interne Datenstruktur einer CORBA–Applikation durch eine objektorientierte Beschreibung der nach außen sichtbaren Objekte und Methoden verdeckt oder *gekapselt*. Dadurch ist es möglich, beinahe jeden beliebigen Programmcode über einen standardisierten Zugriffsmechanismus anderen Systemen zugänglich zu machen.
Für die Implementierung der Schnittstelle wird aus der Schnittstellenbeschreibung (der IDL-Datei) mit einem Precompiler Programmcode mit den Methodenköpfen in der Programmiersprache der jeweiligen Applikation generiert. Die Methoden müssen dann in der Entwicklungsumgebung der Applikation implementiert und übersetzt werden. Die CORBA Laufzeitumgebung wandelt die von einem externen Client kommenden Funktionsaufrufe in Methodenaufrufe der Schnittstelle um. Auf diese Weise können auch bereits übersetzte Programmbibliotheken in eine CORBA Umgebung eingebunden werden, wenn zu jedem Aufruf eine entsprechende Definition in der Schnittstellenbeschreibung vorgesehen ist. Man spricht dann von einer *delegate*-Technik, da die Aufrufe an die

bestehende Bibliothek *delegiert* werden [Red96].
Je nachdem, wie weit die Objekte der Schnittstellenbeschreibung den Strukturen des internen Modells einer Applikation entsprechen, ist der Implementierungsaufwand unterschiedlich groß. Eine durchgängige Schnittstellenstruktur in allen beteiligten Applikationen vereinfacht das Zusammenfügen von Objekten zu einem integrierten Modell.
Im folgenden Text werden zwei Konzepte für eine Integration bestehender Systeme vorgestellt: eine einfache Kapselung, bei der die vorhandenen Schnittstellen der Systeme durch eine Formulierung in IDL direkt über CORBA ansprechbar gemacht werden, und die Definition eines Frameworks, also einer Arbeitsumgebung, in der von den Applikationen semantisch gleichartige Strukturen zur Verfügung gestellt werden.



Einfache Kapselung

Implementierung des Strukturmodells in allen Applikationen

Bild 5.5: Mögliche Anbindungen der Applikationen

Einfache Kapselung der Applikationen

Bei dieser Kopplung werden die Objekte, die in den Programmierschnittstellen der Applikationen definiert sind, in einer IDL-Datei beschrieben und über eine einfache Implementierung gekapselt. Durch diese Kapselung kann auf die Applikationen netzwerktransparent und plattformunabhängig zugegriffen werden. Eine semantische Verknüpfung auf Modellebene und die Abstimmung der Aktivitäten zur Schaffung eines durchgängigen Prozesses muß durch eine zusätzliche Software erfolgen (vgl. Bild 5.5 links). Die Merkmale eines solchen Systems werden in Abschnitt 5.6 beschrieben. In Kapitel 6 wird die Implementierung einer einfachen Kopplung des CAD–Systems Pro/ENGINEER und des Schraubenberechnungsprogramms BOLT vorgestellt.

CAx-Framework

Als Alternative zu der einfachen Kapselung wird in Abschnitt 5.7 ein Konzept für eine CAx-Umgebung vorgestellt. In dieser Umgebung wird für alle Applika-

tionen eine gemeinsame Schnittstelle der bereitgestellten Objekte definiert, die es ermöglicht, mehrere Repräsentationen eines Objektes in den verschiedenen Applikationen zu verwalten (vergleiche Bild 5.5 rechts). Die Schnittstellen sind so gestaltet, daß von allen Systemen Objekte mit gleichen Strukturen und Methoden für die Analyse und die Variation angeboten werden. Auf diese Weise ergeben sich interoperierende Objekte. Die Schaffung einer zusätzlichen Softwarekomponente für die Verknüpfung von Modellen und für die Realisierung eines durchgängigen Prozeßmodells entfällt. Die Applikationen sind eingebettet in eine Umgebung standardisierter Dienste.

5.6 Systemkonzept für einfach gekapselte Applikationen

5.6.1 Systemstruktur

In der Schnittstelle des Geometriemodellierers sind geometrische Objekte, sowie Methoden für die Analyse und die Variation dieser Objekte definiert. Die Schnittstelle des Berechnungssystems enthält benannte Parameter. Über die Methoden können Parameter gesetzt und abgefragt und eine Neuberechnung veranlaßt werden.
Da bei dieser Kopplungsform die zu verknüpfenden Objekte eine unterschiedliche Semantik haben, muß die Verknüpfung in einer zusätzlichen Softwarekomponente erfolgen. Dies kann durch einen separaten Client für beide Systeme realisiert werden. Die geometrischen Objekte des CAD–Systems und die Parameter des Berechnungssystems werden im Client zu einem Strukturmodell zusammengesetzt. Das Strukturmodell ist das Arbeitsmodell des Bearbeiters und wird aus den Modellen der Applikationen abgeleitet. Durch gezielte Interaktionen des Bearbeiters können Variationen am Strukturmodell vollzogen und in den Applikationen nachgeführt werden. Dabei sind für alle möglichen Veränderungen am Strukturmodell Anweisungen und Bedingungen zu formulieren, wie die Variation in den Applikationen umgesetzt werden muß. Beispielsweise müssen bei Veränderung des Schraubendurchmessers im Geometriemodell das Modell der Schraube ersetzt und im Rechenmodell der entsprechende Parameter variiert werden.

5.6.2 Anwendungsszenarien

Diese Kopplung läßt sich für sehr viele bestehende Softwaresysteme realisieren, da sich für die meisten Systeme eine Beschreibung der Schnittstelle in IDL formulieren läßt und sich viele Programmiersprachen direkt oder indirekt in eine CORBA Umgebung integrieren lassen. Die Formulierung in IDL ermöglicht dabei eine gleichzeitige Verwendung von Objekten aus in unterschiedlichen Sprachen implementierten Servern und von verschiedenen Orten eines Netzwerkes.

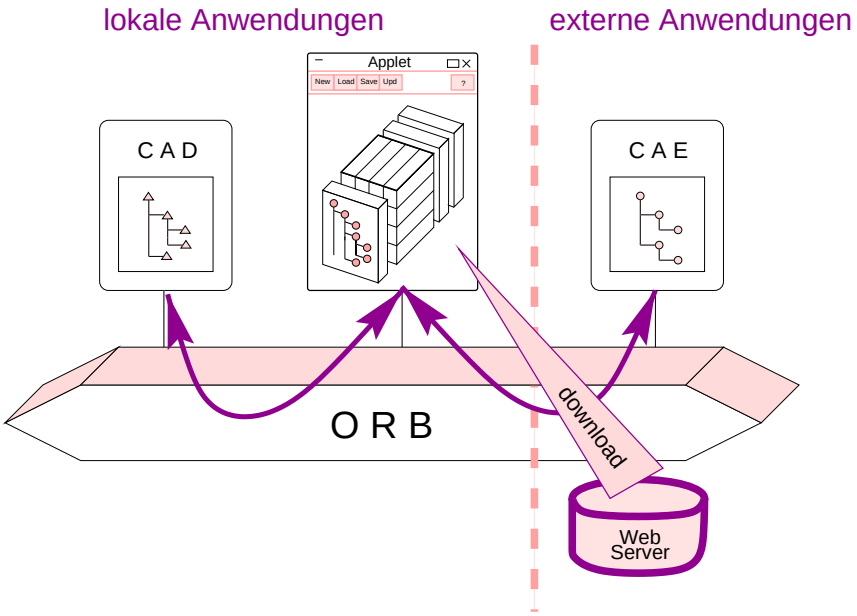


Bild 5.6: Integrationssystem mit einfach gekapselten Applikationen

Sie ist damit besonders geeignet, um Berechnungsprogramme die nur auf einer bestimmten Plattform existieren und nicht portiert werden können, in eine andere Hardeware- oder Softwareumgebung zu integrieren, sofern für beide Plattformen eine CORBA–Laufzeitumgebung existiert.
Für die Programmierung der zusätzlichen Informationen zur Verknüpfung der Modelle und Prozesse bietet sich die Programmiersprache JAVA an. Durch die Plattformunabhängigkeit von JAVA kann diese Softwarekomponente immernoch eingesetzt werden, wenn sich die Plattform einer Applikation ändert. JAVA–Programme haben vollen Zugriff auf die in einer CORBA–Umgebung definierten Strukturen. Darüberhinaus läßt sich mit JAVA eine portable Benutzungsoberfläche für beide Systeme gestalten. Dies ist insbesondere für die Bereitstellung von Methoden im Internet interessant.
Eine Implementierung dieses Ansatzes sowie Variationsmöglichkeiten zur Übertragung des Strukturmodells werden in Kapitel 6 vorgestellt.

5.7 Systemkonzept für ein CAx-Framework

Mit der oben beschriebenen einfachen Kopplung können zwar die bestehenden Programme in einem integrierten System zusammenarbeiten, die Verwendung von Objekten aus den Programmen erfordert aber immer einen auf genau diese Server abgestimmten Client. Der Grundgedanke der Komponententechnologie, nämlich portable, austauschbare Objekte zu schaffen, ist damit nicht erfüllt.

Voraussetzung hierfür wäre die Definition einer Arbeitsumgebung, in der Objekte verschiedener Applikationen als, bis zu einem gewissen Maß, gleichwertige Objekte verwendet werden können und die Objekte von sich aus untereinander Informationen austauschen könnten. In einer solchen Arbeitsumgebung würden die allgemeinen Objekte als Metaklassen in der Arbeitsumgebung definiert und in den Applikationen als spezifische Klassen implementiert. Die Eigenschaften der Objekte wären über standardisierte Zugriffsmethoden zugänglich. Ein durchgängiges Modellschema sorgt für eine eindeutige und konsistente Ablage von Informationen. Das Zusammenwirken der Applikationen müßte durch eine Reihe von allgemeinen und speziellen Diensten unterstützt werden.
In diesem Abschnitt wird ein Konzept für ein CAx-Framework zur Integration von Gestaltung und Berechnung vorgestellt.

5.7.1 Produktmodell und Partialmodelle

In dem CAx-Framework werden die Partialmodelle in den Applikationen als spezifische Ausschnitte eines Gesamtproduktmodells gesehen. In einer neutralen Datenbasis wird in Form einer *Referenzstruktur* ein Verzeichnis der in einem Produkt vorkommenden Baustrukturelemente angelegt. Zu jedem Baustrukturelement der Referenzstruktur können in den Applikationen verschiedene Sichten existieren. Die Referenzstruktur bildet die zentrale Datenbasis, in der alle Verknüpfungen zu den Modellen in verschiedenen Applikationen verwaltet werden.
Die Applikationen bilden spezielle Informationen zu einem Ausschnitt oder zur gesamten Baustruktur ab. Beispielsweise bildet das CAD-System Geometrieinformationen ab und das Rechenmodell ein aus dem Geometriemodell abgeleitetes Ersatzmodell mit Berechnungsverfahren-spezifischen Erweiterungen ab. Dabei sind in allen Partialmodellen einzelne Baustrukturkomponenten identifizierbar (vergleiche Bild 5.7). Die Eigenschaften der Modelle werden durch die im Abschnitt 5.4.3 definierten, benannten *Parameter* beschrieben. Die Parameter sind einem Baustrukturelement zugeordnet und gehören einer der Gruppen *Geometrie*, *Werkstoff*, *Montage*, *Fertigung* oder *Lasten* an.

Referenzstruktur

In einer Referenzstruktur werden alle in einem Produkt vorkommenden Baustrukturelemente und deren Verknüpfungen gespeichert. Für die Abbildung der Verknüpfungen werden auf den Komponenten Bezugsorte definiert, zu denen in den Partialmodellen unterschiedliche Repräsentationen existieren können. Die Referenzstruktur wird in einer von den Applikationen unabhängigen Datenbasis gespeichert.
Die Referenzstruktur ist vergleichbar mit dem in [GAP93] beschriebenen *General Model*, das für die Beschreibung von Teilebibliotheken entwickelt wurde. Zu einem General Model können verschiedenen Repräsentationsformen, beispielsweise

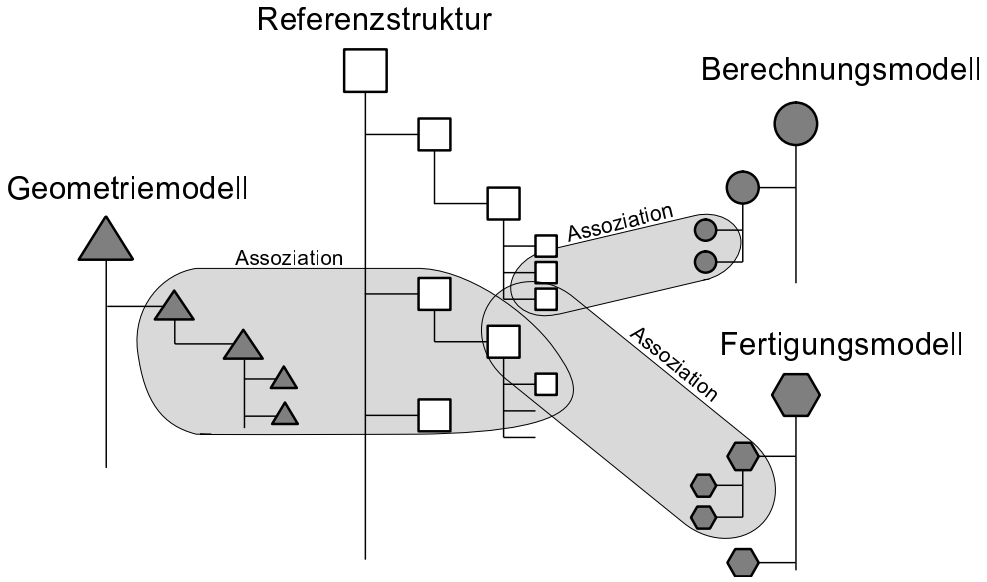


Bild 5.7: Produktmodell aus Referenzstruktur mit assoziierten Partialmodellen (in Anlehnung an [And96])

2D Darstellungen, 3D Darstellungen, FEM–Modelle oder Bauteilkennlinien existieren.

Assoziationen

Zu den Baustrukturelementen der Referenzstruktur werden Informationen gespeichert, in welchen Applikationen Repräsentationen eines Elements existieren und welcher Art die Informationen sind. Dies wird insbesondere für die Konsistenzüberwachung benötigt, wenn bei Änderungen an einem Modell andere Modelle nachgeführt werden müssen. Die Informationen werden auch benötigt, um bei der Definition eines neuen Partialmodells herauszufinden, welche Partialmodelle und Modellinhalte bereits existieren.
Für die Abbildung der Abhängigkeiten zwischen den Partialmodellen kann der Relationship Service aus den *Common Object Services* der CORBA Umgebung verwendet werden. Dieser Dienst erlaubt es, Beziehungen zwischen zwei Objekten zu definieren, indem den Objekten die Rolle, die sie in dieser Beziehung darstellen, zugeordnet wird. Typische Beziehungen sind *Erzeuger* und *Verarbeiter* von Information. Beispielsweise kann durch eine Berechnung eine Schraube dimensioniert werden. Das Rechenmodell ist damit *Erzeuger* der Modellinformation zum Baustrukturelement *Schraube*. Durch Analyse der Relationen kann zurückverfolgt werden, in welchem Partialmodell eine Baustrukturkomponente definiert wurde.

CAD–Komponenten

Die geometrische Repräsentation der Baustrukturelemente ist in den CAD–Komponenten enthalten. Über die Schnittstelle werden Methoden für die Durchführung von geometrischen Analysen und für die Modellierung definiert. Neben den Repräsentationen der Baustrukturelemente sind in diesem Modell auch Repräsentationen der in der Referenzstruktur gehaltenen Bezugselemente enthalten. Auf diese Weise können auch geometrische Eigenschaften und Relationen zwischen Bezugselementen ermittelt werden.
Um den Anwendern Zugriff auf geometrische Eigenschaften der Modelle zu ermöglichen, muß die Schnittstelle eine sehr feingranulare Aufteilung der von ihr angebotenen Objekte aufweisen. Konkret müssen in einem Geometriemodell einzelne geometrische Objekte, wie Punkte, Körperkanten oder Flächen ansprechbar sein. Die Zugriffsmethoden werden sowohl als Funktionsaufrufe als auch in Form von scriptgesteuerten Anfragen ausgeführt.
Da die existierenden CAD-Systeme überwiegend baustrukturorientierte Modelle abbilden, ist die Implementierung dieser Schnittstelle vergleichsweise einfach. Die meisten der Objekte sind in einer ähnlichen Form bereits in der Programmierschnittstelle der gängigen CAD-Systeme enthalten. Unbefriedigend ist die doppelte Datenhaltung der Geometrieobjekte, die sich aus der Kapselung der existierenden Schnittstelle ergibt. Ungelöst ist zur Zeit noch die Übertragung von parametrischen Modellen zwischen verschiedenen CAD-Systemen, insbesondere weil die persistente Speicherung noch in einem systemspezifischen Dateiformat erfolgt. Eine Schnittstelle zu implementieren, mit der parametrische Geometriemodelle fremder Systeme importiert werden können ist das bisher nicht erreichte Ziel einer Integration.

Berechnungskomponenten

In den Berechnungskomponenten werden die Baustrukturelemente um die für die Berechnungsaufgabe benötigten Informationen ergänzt. Dabei können die Parameter des Rechenmodells aus anderen assoziierten Modellen ermittelt werden. Für diesen Zweck werden zu jedem Parameter Bedingungen formuliert, anhand derer der Wert eines Parameters ermittelt werden kann. Beispielsweise kann ein Parameter als Abstand zweier Bezugsflächen definiert sein, und, sofern ein Geometriemodell verknüpft ist, über eine Analysefunktion direkt aus dem Geometriemodell ermittelt werden.
Da von einer Berechnungsmethode üblicherweise nur bestimmte Strukturen berechnet werden können, muß zu jeder Methode ein Modell der berechenbaren Strukturen verfügbar sein. Im Strukturmodell eines Berechnungssystems müssen daher die berechenbaren Verbindungen und die dafür erforderlichen Bedingungen enthalten sein. Darüber hinaus müssen Verknüpfungsanweisungen enthalten sein, mit denen ein geometrisches Ersatzmodell aus einem gegebenen Geometrie-

modell abgeleitet werden kann. Für die Ableitung des Ersatzmodells kann die Funktionalität eines CAD-Systems erforderlich sein.
Je nach Aufgabenstellung kann das Ergebnis einer Berechnung unterschiedlich sein:

Modell. Im Falle einer Auslegungsrechnung wird durch eine Berechnung eine Kombination von Baustrukturelementen und deren Merkmalsausprägungen festgelegt. Üblicherweise werden zu diesen Elementen Modelle in den Norm- und Standardteilbibliotheken existieren.

Dokumente. Als Ergebnis einer Nachweisrechnung werden Dokumente erzeugt, die die Eingabewerte und die erzielten Ergebnisse enthalten. Die Dokumente können zusammen mit dem Geometriemodell archiviert werden.

Änderungsanforderungen. Optimierungsrechnungen können zu Änderungsanforderungen an das Geometriemodell führen. Sofern ein Geometriemodell verknüpft ist, können die Änderungen durch das System, oder falls das nicht möglich ist, durch den Benutzer vollzogen werden.

Durchgängiges Prozeßmodell

Für die Realisierung einer durchgängigen Rechnerunterstützung müssen Aktivitäten mit Eingangsgrößen und Ergebnissen definiert und in allen Applikationen implementiert werden. Die Aktivitäten beschreiben das Zusammenwirken der verschiedenen Softwarekomponenten und sind damit wesentliche Voraussetzung für eine Interoperabilität der Objekte.

5.7.2 Allgemeine und spezielle Dienste

Zu einer Arbeitsumgebung gehören neben den eigentlichen Applikationen Hilfsprogramme und Dienste, die in CORBA unter dem Namen *Common Facilities* zusammengefaßt werden. Wie im Abschnitt 3.2.2 beschrieben, werden die Dienste aufgeteilt in anwendungsgebiet-spezifische (*vertical*) und anwendungsgebietübergreifende (*horizontal*) Dienste. Eine Auswahl der für einen integrierten Entwicklungsarbeitsplatz sinnvollen Dienste wird in den folgenden Abschnitten vorgestellt.

Bereitstellung von Normen

Ein großer Teil der beim Gestalten und beim Berechnen benötigten Daten kann aus Normen oder Bibliotheken bezogen werden. Die Art der benötigten Informationen ist dabei für die Applikationsarten unterschiedlich. Während ein Berechnungsprogramm nur die Parameterwerte benötigt, werden von einem CAD-System auch die zugehörigen Geometriemodelle benötigt. Ein Dienst für die Be-

reitstellung von Normen kann daher aufgeteilt werden, in einen allgemeinen, für alle Applikationen verfügbaren Dienst und einen applikationsspezifischen Dienst.

Applikationsübergreifender Normdienst. Die Liste der verfügbaren Normen und Bibliotheken, sowie die in den Normen enthaltenen Tabellen können als allgemeiner Dienst angeboten werden. Darüberhinaus können die in Normen enthaltenen erläuternden Texte und Bilder als Auswahlhilfe für den Benutzer bereitgestellt werden.

Applikationsspezifische Normdienste. Die von dem allgemeinen Normdienst bereitgestellten Normen und deren Inhalte müssen von den Applikationen in applikationsinterne Darstellungen umgewandelt werden. Bei CAD-Systemen sind dies üblicherweise in Teilebibliotheken abgelegte Geometriemodelle. Zur Wahrung der Konsistenz sollten die Geometriemodelle als parametrisierte Modelle abgelegt werden, deren Parameter durch die vom allgemeinen Normdienst bereitgestellten Werte gesetzt werden. Der applikationspezifische Normdienst für ein CAD-System muß für die richtige Auswahl des zu einer Norm gehörenden Geometriemodells und für Zuordnung der Parameter sorgen.

Für ein Berechnungssystem müssen die vom allgemeinen Normdienst bezogenen Parameter den programmspezifischen Parametern zugeordnet werden.

Für beide Systeme gilt, daß manchmal für die Ermittlung von Parametern weitere Normen herangzogen werden müssen. Beispielsweise müssen nach der Auswahl einer Schraube mit metrischem Gewinde die für die Berechnung benötigten Gewindeabmessungen und Toleranzen aus der Norm für metrische Gewinde bezogen werden.

Hilfesystem

Die Bereitstellung von Hilfeinformationen kann ebenfalls in Form eines allgemeinen Dienstes erfolgen. Auch hier kann in allgemeine und systemspezifische Komponenten unterteilt werden.

Allgemeines Hilfesystem. Diese applikationsübergreifende Komponente bietet Informationen über die Arbeitsumgebung und Metadaten über die bearbeiteten Modelle an. Für die Arbeitsumgebung sind dies Informationen über verfügbare Programme, Daten und Dienste. Zu den Metadaten der Modelle gehören Verknüpfungen mit anderen Partialmodellen, Änderungsdatum, Bearbeiter und ähnliches.

Denkbar ist auch, daß weitere Informationssysteme, zum Beispiel für Gestaltungsregeln, oder recyclinggerechte Produktentwicklung in das System eingebunden werden. Unter dem Oberbegriff Hilfesystem können so die

Informationssysteme und die Navigatoren für die anderen Systemkomponenten Normdatenbank, Dokumentationssystem und PDM-System zusammengefaßt werden.

Applikationsspezifische Hilfesysteme. Dies sind die Onlinehilfesysteme der beteiligten Applikationen.

Protokollierung und Dokumentation

Ein wichtiger Arbeitsschritt beim Arbeiten mit verteilten Modellen ist die nachvollziehbare Protokollierung der durchgeführten Arbeitsschritte (vergleiche Kapitel 4). Der *Event Service* von CORBA bietet die Möglichkeit bestimmte Ereignisse zu definieren die allen oder bestimmten Applikationen mitgeteilt werden. Durch Definition charakteristischer Ereignisse für die Erstellung oder Veränderung von Modellen können die Aktivitäten überwacht und protokolliert werden.

5.7.3 Speicherung der Modelle

Beim Arbeiten mit im Netzwerk verteilten Applikationen ist der Speicherung der Modelle aufwendiger als bei rein lokalen Anwendungen. Aus Gründen der Datensicherung und der Überwachung von Änderungen ist eine Speicherung in einem zentralen Speichermedium anzustreben [AK98]. Voraussetzung dafür ist, daß sich alle Modelle unabhängig von der zugehörigen Applikation in einem entsprechenden Medium speichern lassen. Der CORBA Dienst *Persistence* definiert zwar standardisierte Methoden zur Speicherung und Wiederherstellung von Objekten, die Implementierung dieses Dienstes liegt aber in der Hand des Programmierers, da für die Speicherung ein Abbild der internen Datenstruktur der gekapselten Applikation erstellt werden muß. Der *Persistence Service* verwendet für die Speicherung die im *Serialization Service* definierten Methoden zur Abbildung von Objekten in einen kontinuierlichen Speicherbereich. Bei der Implementierung könnte die entsprechende systemspezifische Datei als Datenstrom übertragen und in ein lokales Dateisystem abgelegt werden.
Für die Verwaltung der gespeicherten Daten bietet sich die Verwendung der in [OMG98] definierten Methoden eines Produktdatenmanagementsystems an.

5.7.4 Systemstruktur

In Bild 5.8 auf Seite 101 ist ein Überblick über die im Framework vorkommenden Komponenten dargestellt:
Neben den Applikationen, die jeweils ein eigenes Strukturmodell halten, sind im System die allgemeinen und die speziellen Dienste vorhanden. Die speziellen Dienste, Normen, PDM-System und Hilfesystem stellen die wesentlichen Methoden für die Arbeit mit CAx-Komponenten zur Verfügung. Sie verwenden dafür

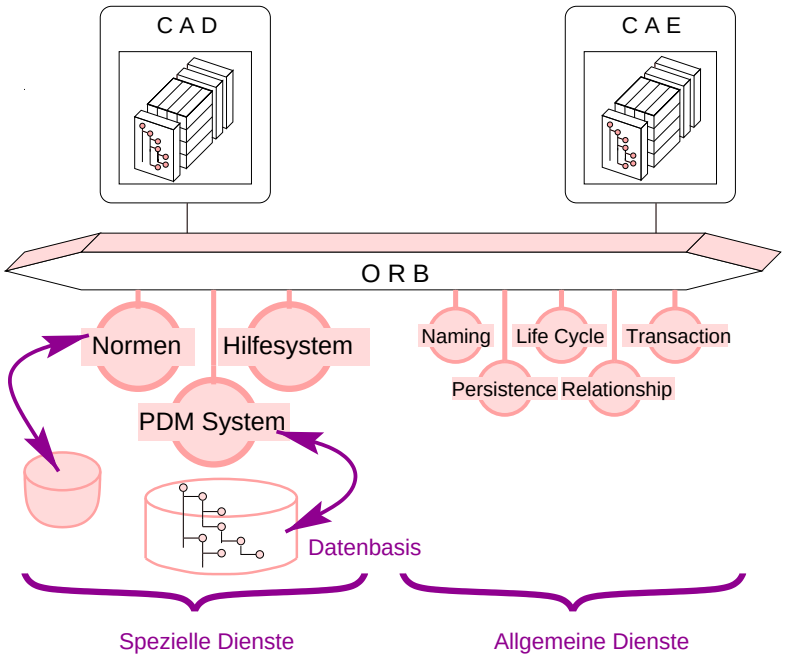


Bild 5.8: Systemstruktur des CAx-Frameworks

die Methoden der allgemeinen Dienste *Naming*, *Life Cycle*, *Transaction*, *Persistence* und *Relationship*.

5.7.5 Anwendung

In einem Beispiel soll ein möglicher Ablauf einer integrierten Berechnung beschrieben werden. Die Aufgabenstellung lautet: in einer lokalen CAD-Sitzung soll für eine vorhandene Verbindungsumgebung eine Schraube ausgewählt werden. Die Betriebskräfte und weiteren Randbedingungen bezüglich Fertigung und Montage seien bekannt und können vom Benutzer eingegeben werden. Der Ablauf sieht dann wie folgt aus:

1. Über das **Informationssystem** erhält der Benutzer eine Liste der verfügbaren Berechnungssoftware und der damit durchführbaren Berechnungen. Der Benutzer wählt eine Methode aus und sendet eine Anfrage an die CORBA-Umgebung, daß eine Berechnung mit dem aktuellen Geometriemodell durchgeführt werden soll.

2. Mit Hilfe des **Implementation Repository** der CORBA-Umgebung wird der zur gewählten Methode gehörende Server lokalisiert und gegebenenfalls automatisch gestartet. Dem Berechnungsserver wird das Geometriemodell als Arbeitsmodell übergeben.

3. Anhand der Bedingungen und Relationen im Rechenmodell wird das Geometriemodell analysiert. Dabei werden die Analysefunktionen durch das CAD-System ausgeführt. Anschließend wird durch das Berechnungssystem eine Schraube ausgewählt. Im Rechenmodell ist diese Schraube durch ihre Normbezeichnung spezifiziert.

4. Aus dem Normdienst des CAD-Systems wird auf Anforderung des Benutzers mit den Daten des allgemeinen Normdienstes und der vom Berechnungssystem erstellten Normbezeichnung ein Geometriemodell der gewählten Schraube erzeugt. Das Rechenmodell stellt die für den Einbau benötigten Bedingungen zur Verfügung.

5. Bei Abschluß der Sitzung werden alle betroffenen Datenbasen aktualisiert. Dem Geometriemodell und der Referenzstruktur wurden jeweils eine Komponente hinzugefügt. In der Datenbasis existieren jetzt zwei Verknüpfungen zum Produkt: eine zum Geometriemodell und eine zum Rechenmodell.

Aus der Systemstruktur ergeben sich darüber hinaus noch weitere Nutzungsmöglichkeiten des Systems:

Nutzung der Geometriekomponenten: Die vom CAD–System bereitgestellten Komponenten können von allen geometrieverabeitenden Systemen genutzt werden. Beispiele hierfür sind FEM–Systeme oder Simulationssoftware.

Nutzung der Berechnungskomponenten: Die bei der Berechnung festgelegten Größen haben Einfluß auf viele weitere Teilbereiche der Produktentwicklung. Beispielsweise werden Oberflächenbeschaffenheiten auch bei Festlegung des Fertigungsablaufs benötigt. Die für alle Bereiche gleiche Struktur der Modelle ermöglicht es dem Anwender, die gesuchte Information schnell aufzufinden. Die standardisierten Zugriffsmethoden erlauben es, auch direkte Verknüpfungen zwischen den Daten verschiedener Modelle herzustellen.

Methodensuche: Die Auswahl der Berechnungsmethoden erfolgt mit Hilfe des Informationssystems durch den Anwender. Die tatsächliche Lokalisierung und Aktivierung der zugehörigen Server wird mit Hilfe der entsprechenden Verzeichnisse durch die CORBA-Lauftzeitumgebung erledigt.

Berechnungen durch externe Anbieter: Mit dem System können auch Berechnungsmethoden externer Anbieter integriert werden. Dabei können alle der in Abschnitt 2.5.1 genannten Szenarien, Modellübertragung, Methodenübertragung oder direkte Verknüpfung realisiert werden. Die Sicherheitmechanismen von CORBA können für eine entsprechende Einschränkung der Zugriffsrechte und Verschlüsselung der Übertragung benutzt werden.

5.7.6 Implementierung

Die Realisierung eines CAx-Frameworks ist wesentlich aufwendiger als die einfache Kapselung bestehender Applikationen, da neben der Implementierung der Schnittstellen auch zahlreiche Dienste entworfen und implementiert werden müssen. Der folgende Abschnitt gibt einen Überblick über die erforderlichen Arbeitsschritte und die Erweiterungsmöglichkeiten.

Arbeitsschritte

- Das in Abschnitt 5.4.3 vorgestellte Strukturmodell muß in allen beteiligten Applikationen implementiert werden. Dazu gehören auch eine sinnvolle Verteilung der Informationen auf die verschiedenen Modelle, sowie Methoden zur Datenabfrage, Archivierung und für die Konsistenzüberwachung. Grundlage ist hierfür eine zu schaffende gemeinsame Schnittstelle aller Applikationen.

- Zur Speicherung der Referenzstruktur muß ein geeignetes Speichermedium ausgewählt und für die Abbildung von Baustrukturen und Verknüpfungen konfiguriert werden.

- Der Normdienst muß konzipiert und implementiert werden. Dazu gehört die Einrichtung einer Datenbank für die allgemeinen Tabellen der Normen sowie die Bereitstellung der applikationsspezifischen Sichten der in der Datenbank gehaltenen Modelle.

- Die allgemeinen Dienste, die nicht Bestandteil der CORBA-Laufzeitumgebung sind, müssen spezifiziert und implementiert werden. Dazu gehören die Definition von Beziehungen und Rollen im *Relationship Service* und die Implementierung der Methoden des *Persistetence Service* und des *Serialization Service*.

Erweiterungsmöglichkeiten

Eine bestehende Implementierung eines CAx-Frameworks läßt sich erweitern, indem weitere Applikationen entsprechend gekapselt und eingebunden werden. Voraussetzung für eine Interoperabilität mit den vorhandenen Systemen ist, daß die hinzukommenden Applikationen ebenfalls in der Lage, sind mit Baustrukturkomponenten zu arbeiten und die für die Speicherung und Verknüpfung der Modelle benötigten Methoden bereitstellen. Ist dies der Fall, können die Applikationen auf einer hohen semantischen Ebene Daten austauschen. Die bereits bestehende Infrastruktur steht allen Applikationen gleichermaßen zur Verfügung.

Realisierbarkeit

Die praktische Umsetzung dieses Konzeptes ist mit erheblichem Aufwand verbunden. Die Schaffung eines Strukturmodells, in dem sich die Geometrieinformationen und verschiedenen Berechnugsmodelle abbilden lassen, erfordert ein sehr umfangreiches Konzept. Die Implementierung dieses Konzeptes in bestehenden Softwareumgebungen ist eine weitere schwierige Aufgabe, da allein die Kopplung zweier geometrieverarbeitender Systeme zu erheblichem Aufwand führt und nur mit eingeschränkter Funktionalität realisierbar ist [AJSK98]. Im Vergleich dazu ist die Implementierung der allgemeinen und speziellen Dienste einfach zu realisieren, insbesondere weil durch die standardisierten Dienste von CORBA ein großer Teil schon implementiert ist.

5.8 Vergleich der Systemkonzepte

Abschließend werden die beiden aufgestellten Systemkonzepte hinsichtlich des Aufwands und des erzielbaren Nutzens verglichen. Wegen der noch relativ groben Struktur kann die Bewertung nur Anhaltswerte liefern.

Aufwand

Der Aufwand kann aufgeteilt werden in Aufwand zur Modellierung des Systems und in Implementierungsaufwand. Zur Modellierung gehören die die Formulierung von Modellinhalten und die Schaffung eines Prozeßmodells. Die Implementierung kann in Client- und Serverkomponenten aufgeteilt werden.
Der Modellierungsaufwand für ein CAx-Framework ist dabei wesentlich höher als bei einer einfachen Kopplung, da hier ein Objektmodell und ein Prozeßmodell geschaffen werden müssen, die sich in allen Applikationen abbilden lassen. Dementsprechend ist auch der Implementierungsaufwand in den Servern höher, da das neue Objektmodell auf die internen Modelle der Applikationen abgebildet werden muß. Bei der einfachen Kopplung besteht der Aufwand für die Kopplung darin, die Modelle und Prozesse zweier konkreter Systeme aufeinander abzustimmen. Der serverseitige Implementierungsaufwand ist bei der einfachen Kopplung gering, da die CORBA–Schnittstellen Abbilder der vorhandenen Schnittstellen sind.

Nutzen

Als Nutzen werden die Erweiterbarkeit, die Anwenderfreundlichkeit und die Wiederverwendbarkeit der geschaffenen Strukturen betrachtet.
Die Erweiterbarkeit der einfachen Kopplung ist schwierig, da die implementierten Schnittstellen und die Integration im Client speziell auf die Applikationen abgestimmt sind. Gleiches gilt für die Wiederverwendbarkeit, wobei sich durch die

Formulierung in IDL die Objekte zumindest auch in anderen Systemumgebungen verwenden lassen. Eine Anpassung der Modelle ist aber immer erforderlich.
Die Erweiterbarkeit und Wiederverwendbarkeit der Objekte im CAx–Framework ist gut, da diese Objekte als portable, interoperierende Objekte mit ähnlicher Struktur definiert wurden. Ein klares und umfassendes Objektmodell erleichtert dabei die Eingliederung neuer Systeme. Darüberhinaus bietet der objektorientierte Ansatz von CORBA die Möglichkeit, bestehende Objekte durch Vererbung und Erweiterung mit zusätzlichen Eigenschaften und Methoden zu ergänzen. Aus einem gegebenen Basissystem können so problemspezifische Lösungen entwickelt werden. Die Anwenderfreundlichkeit eines CAx–Frameworks ist größer, da durch die ähnliche Struktur aller Objekte Informationen leichter identifiziert werden können. Bei der einfachen Kopplung hingegen besteht das Integrationssystem aus einer applikationsspezifischen Anpassung und ist daher für jede Kopplung unterschiedlich.
Bild 5.9 zeigt eine Darstellung der Bewertung. Man erkennt, daß die Vorteile des CAx–Frameworks einen wesentlich höheren Aufwand nach sich ziehen. Außerdem wird deutlich, daß bei der einfachen Kopplung der Hauptanteil der Implementierung im Client liegt.

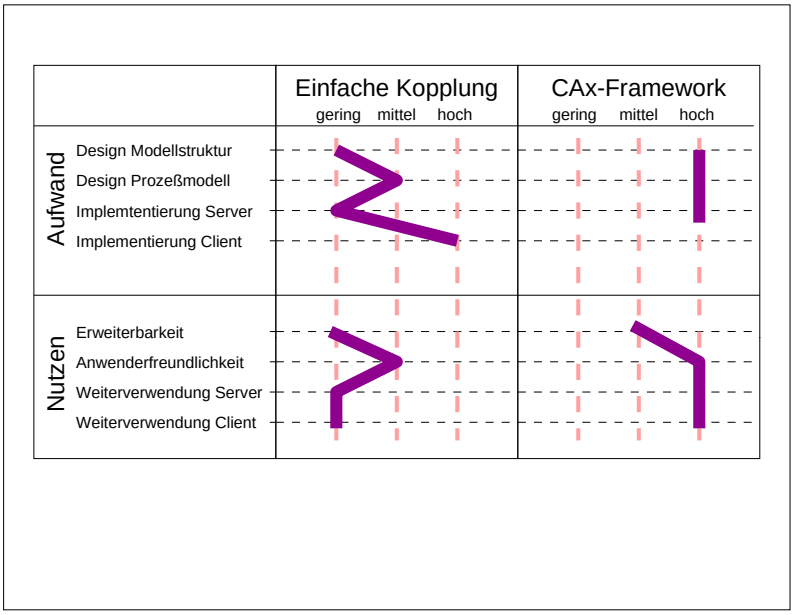


Bild 5.9: Vergleich der Integrationskonzepte

Fazit

Der Aufwand für die Definition und Implementierung eines CAx–Frameworks ist mit dem heutigen Stand der Technik im Vergleich zum erzielbaren Nutzen erheb-

lich zu groß. Insbesondere die Definition einer Datenstruktur, die in der Lage ist, die Modelle bestehender Applikationen vollständig abzubilden, ist kaum realisierbar. Dennoch sollte das CAx–Framework als Vision für zukünftige Entwicklungen gesehen werden, da mit diesem System verschiedene Softwaresysteme in verteilter Umgebung sinnvoll zusammenarbeiten können. Wegen des Umfangs dieser Aufgabe sollte die Entwicklung von den Systemherstellern selber vorangetrieben werden. Die Entwicklungen im Bereich CAD–Referenzmodell und STEP, sowie die Arbeiten der OMG selber sind hierfür eine gute Basis.
Für die Lösung der Integrationsaufgabe ist die vorgestellte einfache Kopplung eine realisierbare Variante. Mit relativ geringem Aufwand kann hiermit eine plattformübergreifende und netzwerkfähige Integration realisiert werden. Dieses Konzept wurde daher aufgegriffen und exemplarisch implementiert.

6 Implementierung eines Integrationssystems

In diesem Kapitel wird eine Implementierung der in Abschnitt 5.6 beschriebenen, einfachen Kopplung zweier bestehender Programme vorgestellt. Ziel der Integration war es, in den CAD–Gestalungsprozeß integrierte Berechnungen von Schraubenverbindungen durchzuführen. Für das System wurden das CAD–System Pro/ENGINEER und das Berechnungsprogramm für Schraubenverbindungen BOLT [BOL94] in die CORBA-Umgebung SOM [IBM96] der Firma IBM integriert. Pro/ENGINEER ist ein verbreitetes, parametrisches 3-D CAD–Sytem und repräsentiert damit den Stand der Technik auf dem Gebiet Geometriemodellierung. BOLT ist am Institut für Maschinenkonstruktion - Konstruktionstechnik der TU Berlin entwickelt worden und steht beispielhaft für ein Rechenverfahren, bei dem standardisierte Konstruktionselemente, hier Schrauben, in einer individuell gestalteten Umgebung eingesetzt und berechnet werden müssen. Dabei ist die Steuerung über Eingabedateien mit benannten Parametern und die Implementierung in FORTRAN typisch für eine große Zahl in der Industrie existierender Anwendungen [VDI99].

6.1 Übersicht

6.1.1 Systemstruktur

Das System gliedert sich in die beiden Applikationen, Pro/ENGINEER und BOLT, die über entsprechende Schnittstellen mit dem *Object Request Broker* verbunden sind. Die Schnittstellendefinitionen beschreiben die Objekte und Methoden der in den Programmierschnittstellen definierten Objekte. Dies sind auf der Seite des CAD-Systems geometrische Objekte und für das Berechnungsprogramm benannte Parameter.
Beide Applikationen arbeiten als Server. Die Verknüpfung der beiden Applikationen auf semantischer Ebene erfolgt durch einen in Java implementierten Client, den Modellmanager. Der Modellmanager bildet somit das Strukturmodell ab (vgl. Bild 5.3 auf Seite 90).
Eine abstrakte Schnittstellenbeschreibung definiert allgemeine, von allen Appli-

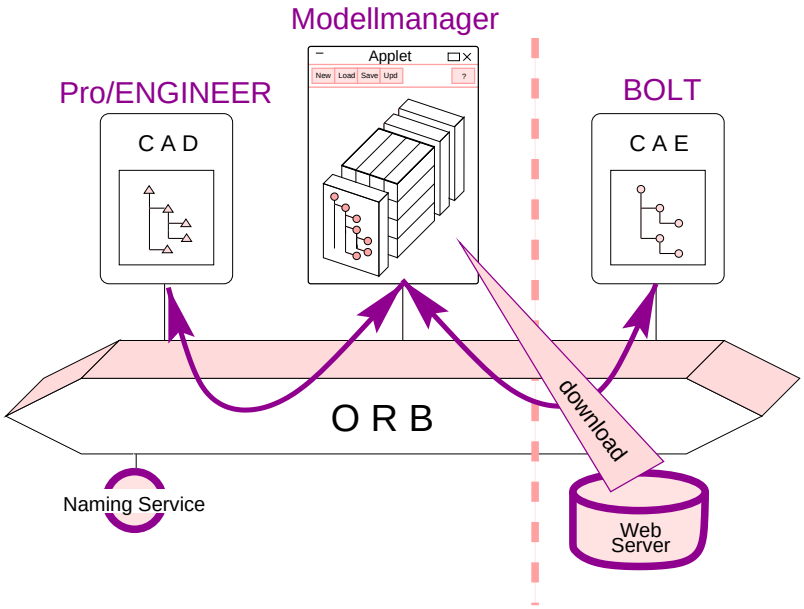


Bild 6.1: Systemstruktur des implementierten Integrationssystems

kationen benötigte Objekte, wie einen Basistyp für Modelle und einfache Objekte sowie die zugehörigen Methoden zum Identifizieren und Vergleichen von Objekten. Außerdem werden in Form eines *Common Object Service* verschiedene Containerklassen zum Transport von Objekten bereitgestellt.
Das System ist so aufgebaut, daß die Geometriemodellierung auf dem lokalen Rechner erfolgt, während das Berechnungprogramm von einem entfernten Server bezogen wird. Diese Aufteilung wurde aus folgenden Gründen vorgenommen:

- Die für die Darstellung von Geometriedaten erforderliche zu übertragende Datenmenge ist zur Zeit noch sehr groß, so daß sich bei einem entfernt laufenden CAD-System eine starke Netzbelastung ergeben würde.
- Die verfügbaren CAD-Systeme sind für einen lokalen Betrieb optimiert.
- Das Geometriemodell ist der eigentliche "Modellierungsgegenstand" und ist Eigentum des Anwenders, während das verwendete Berechnungsprogramm auch von einem externen Anbieter bezogen werden könnte.

Eingebettet in eine Unternehmensstruktur würde die Funktionalität des CAD-Systems am Konstruktionsarbeitsplatz durch von der Berechnungsabteilung oder von externen Anbietern bereitgestellte Berechnungsmethoden erweitert. Das Geometriemodell wird mit den vorgenommenen Änderungen und den zusätzlichen Informationen in der lokalen Datenbasis gespeichert. Die verwendeten Methoden bleiben "Eigentum" des Anbieters, werden nicht mit gespeichert und behalten auch keine Modellinformationen in ihrer Datenbasis.

6.1.2 SOM als CORBA–Laufzeitumgebung

Als CORBA-Umgebung wurde SOMobjects 3.0 der Firma IBM gewählt. SOMobjects ist aus dem *System Object Model (SOM)* entstanden, das Grundlage für das Betriebssystem OS/2 der Firma IBM war. Mit der Entwicklung von CORBA wurde das System durch Erweiterungen und Anpassungen konform zu dem neu definierten Standard gemacht. Die Version SOMobjects 3.0 ist konform zum Standard CORBA 2.0.

Die zu SOMobjects gehörende Software ermöglicht es, Arbeitsumgebungen zu entwickeln, die sich aus verteilten, interoperierenden Objekten zusammensetzen. Die Software besteht aus folgenden Komponenten [IBM96]:

IDL Compiler. Die neutralen Schnittstellenbeschreibungen werden mit einem IDL-Compiler in entsprechende sprachabhängige Konstrukte umgewandelt. Serverseitig sind dies Methodenköpfe und die zugehörigen Deklarationen. Für die Verwendung im Client werden die Deklarationen erzeugt, die die neu definierten Datentypen im Clientprogramm bekannt machen und für die Weiterleitung der Methodenaufrufe an die Laufzeitumgebung sorgen. Servercode kann in den Programmiersprachen *C* und *C++* und Clientdeklarationen können für *C*, *C++* und *Java* erstellt werden.

SOM Laufzeitumgebung. Das verteilte Objektmodell von CORBA wird durch die SOM Laufzeitumgebung realisiert. Die Softwarekomponente verwaltet die Klassen und die dazugehörigen Objekte und leitet die Methodenaufrufe an die Implementierungen der Objekte weiter.

Verteilte Arbeitsumgebung. Die in CORBA festgelegte Infrastruktur für ein CORBA-basiertes System wird durch die verteilte Arbeitsumgebung (engl.: *Distributed Framework*) bereitgestellt. Dazu gehören der *Object Request Broker* und die verschiedenen Verzeichnisse über die im System definierten Schnittstellen und Implementierungen.

Object Services. Zusätzlich zur Arbeitsumgebung werden die implementierten *Object Services* zur Verfügung gestellt. Dabei wird in SOM 3.0 nur ein Teil der in CORBA definierten Dienste angeboten. Die implementierten Dienste sind: *Life Cycle*, *Security*, *Transaction* und *Naming*.

Implementierungen von SOMObjects existieren für die Betriebssysteme *AIX*, *OS/2* und *Windows95*.

6.2 Geometriemodellierer

6.2.1 Pro/ENGINEER

Das CAD–System Pro/ENGINEER ist ein durchgängig parametrischer 3-D Geometriemodellierer. Die Modellierung erfolgt durch Hinzufügen von Konstruktionselementen (engl.: *Features*) zu einem als Basiselement bezeichneten Ausgangskörper. Als Konstruktionselemente können die systeminternen Grundelemente, wie Bohrung, Fase, Schlitz oder Massivkörper, oder aus diesen Grundelementen zusammengesetzte Geometriemakros verwendet werden. Die Konstruktionselemente werden mit Hilfe von Bedingungen und benannten Parametern an bestehenden Objekten ausgerichtet.
Mehrere Bauteile können zu einer Baugruppe zusammengesetzt werden. Dabei kann die Positionierung der Komponenten sowohl anhand von Koordinaten für Lage und Ausrichtung erfolgen als auch durch die Verknüpfung von geometrischen Objekten auf beiden Komponenten über einfache geometrische Relationen.
Die Parameter eines Bauteils können zueinander in Beziehung gesetzt oder aus einer gespeicherten Tabelle bezogen werden. In verschiedenen Modulen, zum Beispiel für die Ableitung von Fertigungszeichnungen oder für die Erzeugung von NC-Datensätzen, können die erzeugten Modelle weiterverarbeitet werden. Pro/ENGINEER verwendet dabei für alle Darstellungen dieselbe Datenbasis, sodaß bei Änderungen in einer Darstellung alle anderen Sichten automatisch nachgeführt werden.
Für den programmgesteuerten Zugriff auf die Geometriemodelle steht mit Pro/-TOOLKIT eine leistungsfähige Programmierschnittstelle zur Verfügung. Mit der Schnittstelle sind alle Operationen möglich, die auch über die Benutzungsoberfläche ausführbar sind. Darüber hinaus lassen sich die Benutzungsoberflächen um weitere Menüpunkte und die internen Datenstrukturen um zusätzliche Merkmale erweitern.
Die Implementierung von Pro/TOOLKIT Applikationen erfolgt entweder als dynamisch gebundene Bibliothek oder als eigenständiges Programm, das über Interprozeßkommunikation auf den Pro/ENGINEER Prozeß zugreift. Die letztere Variante bietet eine größere Flexibilität. Auf diese Weise kann Pro/ENGINEER als Teil einer übergeordneten Anwendung gestartet werden [Par97].

6.2.2 Schnittstellendefinition des Geometriemodellierers

In der Geometrieschnittstelle sind zwei Partialmodelltypen, *Part* und *Assembly*, für Bauteile und Baugruppen definiert. Partialmodelle können als eigenständige Modelle in die Applikation geladen werden. Die Methoden zum Aufrufen und Schließen von Baugruppen und Bauteilen sind in der abstrakten Metaklasse für Partialmodelle definiert. Eine Baugruppe besteht aus einem oder mehreren Bauteilen und den Verknüpfungen zwischen diesen Bauteilen. Ein Bauteil kann in

seine geometrischen Bestandteile Flächen, Konturen, Kanten, Punkte und Konstruktionselemente zerlegt werden. Zusätzlich sind in beiden Modelltypen Bezugsebenen und -achsen definiert, an denen beim Modellieren die Konstruktionselemente positioniert werden. Bild 6.2 zeigt die Klassenhierarchie der geometrischen Objekte. Die Klassen wurden für das CAD-System Pro/ENGINEER implementiert, könnten aber auch für andere CAD–Systeme, beispielsweise CATIA implementiert werden. Die interne Repräsentation der Objekte wäre dann unterschiedlich, aber trotzdem könnte in gleicher Weise auf sie zugegriffen werden.

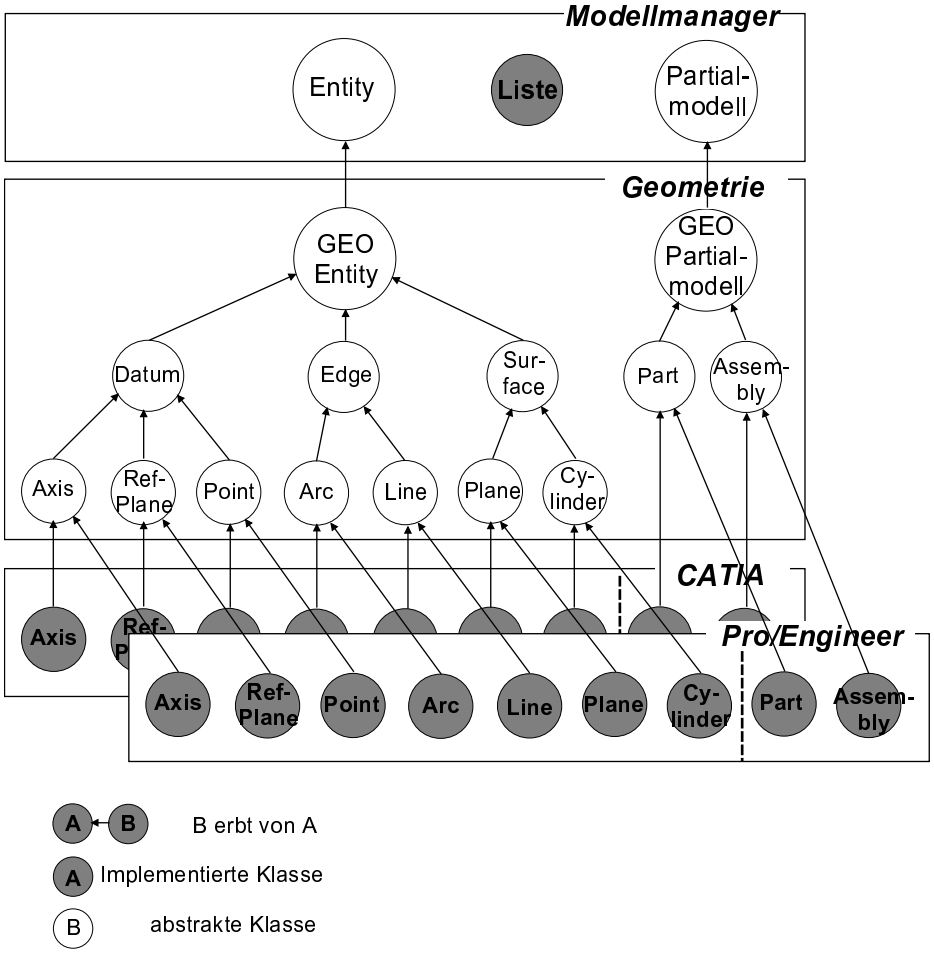


Bild 6.2: Struktur der Schnittstelle zum Geometriemodellierer

Für die Analyse von Geometriemodellen stehen Funktionen zum Auffinden von Objekten mit einem bekannten Namen und von Objekten, die in einer bestimmten Relation zu einem gegebenen Objekt stehen, zur Verfügung. Durch Namen können Norm- und Standardteile sowie Konstruktionselemente identifiziert werden, die aus einer Bibliothek entnommen sind und deren Name daher bekannt ist. Die Identifizierung von Objekten anhand von Relationen wird für Struktu-

ren verwendet die sich nicht, oder nicht in jedem Fall aus Bibliothekselementen zusammensetzen. Die gesuchten Elemente und die zwischen ihnen bestehenden Relationen werden in Form eines Relationsgraphen übergeben. Bild 6.3 zeigt einen Relationsgraphen für eine einfache Bohrung. Die Analysefunktion zerlegt den Graphen und versucht eine eindeutige Zuordnung von Objekten im Graphen zu Objekten des Geometriemodells herzustellen. Ist die Abbildung nicht eindeutig möglich, wird vom Benutzer eine Auswahl per Mausklick angefordert. Das Ergebnis des Aufrufs wird als Liste der gefundenen Elemente an den Client zurückgegeben. Wegen der sehr schnell sehr komplex werdenenden Relationsgraphen ist es auch möglich, die gesuchten Bezugselemente direkt per Mausklick auszuwählen.

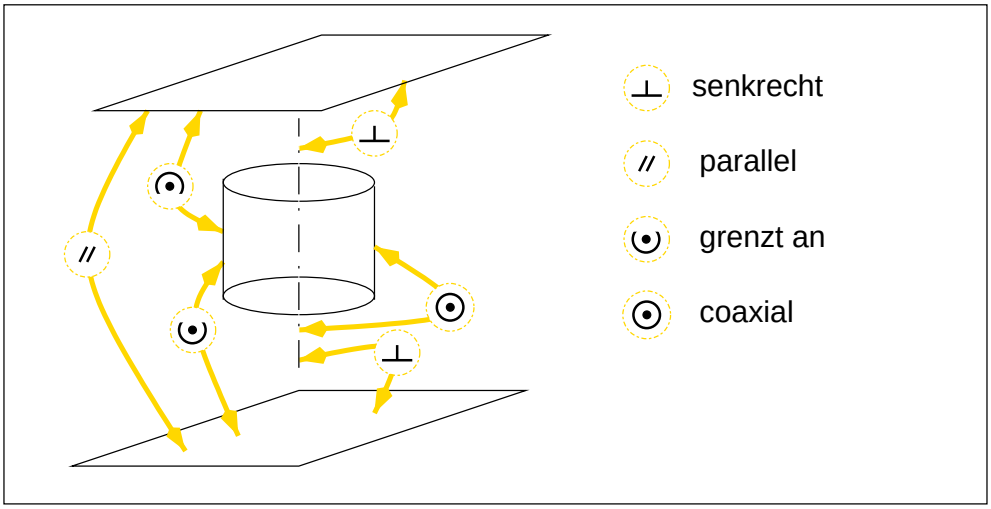


Bild 6.3: Relationsgraph für eine einfache Bohrung

Zur Ermittlung von geometrischen Eigenschaften stehen Methoden zur Verfügung, die sowohl Eigenschaften eines einzelnen Objektes wiedergeben (Länge einer Kante, Flächeninhalt, usw.) als auch Methoden, die aus der Relation zweier Objekte entstehen (Abstand zweier Achsen, Winkel zwischen zwei Ebenen usw.). Bei Eigenschaften, denen im Parametrikmodell ein Parameter zugeordnet ist, kann auch der Parameter direkt verknüpft werden. In diesem Fall sind die Eigenschaften des Modells durch den Client veränderbar.

Eine automatische Variation des Geometriemodells ist außerdem durch Hinzufügen oder Entfernen von Bauteilen oder Konstruktionselementen möglich. Norm- und Standardteile können, sofern sie aus einer Teilefamilie stammen, über die Schnittstelle durch andere Elemente der selben Familie ersetzt werden.

6.2.3 Implementierungsdetails

Programmodule

Die Implementierung der CAD-Schnittstelle besteht aus dem eigentlichen Serverprogramm, der Bibliothek mit den Objektimplementierungen und dem Programm Pro/ENGINEER. Das Serverporgramm wird als Kindprozeß durch die als Dämon laufende CORBA Umgebung gestartet, sobald eine Anfrage von einem Client nach einem Geometrieobjekt gestellt wird. Das Serverprogramm ist eine Standardimplementierung aus der CORBA–Umgebung und bildet das Hauptprogramm für die als dynamisch gebundene Bibliothek realisierte Objektimplementierung. In der Objektimplementierung sind die in der Schnittstellenbeschreibung enthaltenen Konstrukte definiert und mit Hilfe der Pro/TOOLKIT Bibliotheken implementiert. Zu Beginn einer Sitzung wird durch die Objektimplementierung der Pro/ENGINEER Prozeß im asynchronen Modus gestartet. In diesem Modus ist das Serverprogramm das Hauptprogramm, das Funktionen im Pro/ENGINEER Prozeß aufruft, während im synchronen Modus der Pro/ENGINEER Prozeß das Hauptprogramm ist [Par97]. Der Datenaustausch zwischen Serverprogramm und Pro/ENGINEER erfolgt über Interprozeßkommunikation. Die Kommunikation zwischen dem Client und den Objektimplementierungen erfolgt über *Remote Procedure Calls*. Eine schematische Darstellung der Programmodule zeigt Bild 6.4.

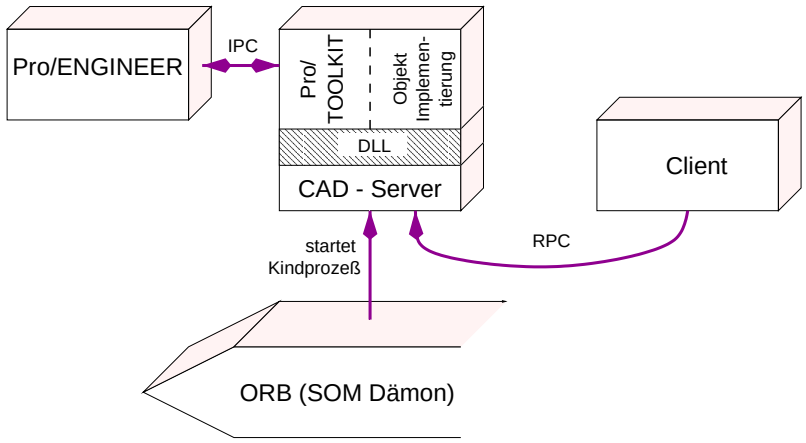


Bild 6.4: Programmodule des Geometrie–Servers

Struktur der Objekte

Die Objektimplementierung ist für die Speicherverwaltung aller durch Client–Programme erzeugte Objekte verantwortlich. Als Attribute sind in diesen Objekten Identifizierungsmerkmale und Verwaltungsdaten gespeichert. Geometrische

Merkmale werden immer aktuell aus dem CAD–System ermittelt. Zur Identifizierung werden die Speicheradresse und die Identifizierungsnummer (Id) der entsprechenden Struktur im CAD–System gespeichert. Mit Hilfe der nur während einer Sitzung gültigen Speicheradresse können Funktionsaufrufe direkt an die Struktur im CAD–System weitergeleitet werden. Die Identifizierungsnummer kann für die dauerhafte Speicherung von Objektreferenzen verwendet werden, da sie über mehrere Sitzungen unverändert bestehen bleibt. Dies kann bei der Archivierung von integrierten Berechnungen genutzt werden. Außerdem sind in der Objektimplementierung Topologieinformationen gespeichert. Dies sind die übergeordnete Struktur die das Objekt enthält und die weiteren Strukturen, die diesem Objekt zugehören. Am Beispiel einer Fläche (*Plane*) ist das übergeordnete Objekt das enthaltende Bauteil und die zugehörigen Objekte sind die begrenzenden Kanten. Bild 6.5 zeigt eine symbolische Darstellung der Klasse *Plane*.

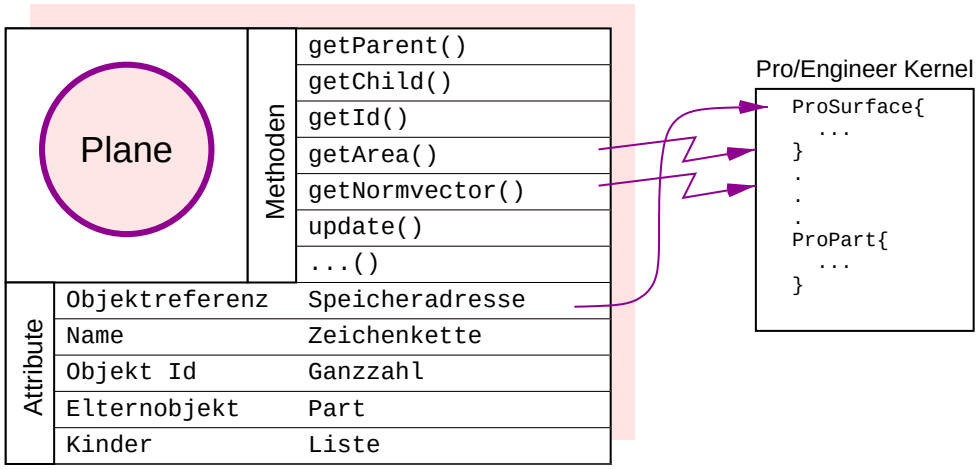


Bild 6.5: Struktur eines Geometrieobjektes in der Objektimplementierung

Um den Aufwand für die Konsistenzüberwachung zu minimieren, werden geometrische Eigenschaften nicht in der Objektimplementierung gespeichert, sondern bei Bedarf als Funktionsaufruf aus dem CAD–System ermittelt.
Eine Inkonsistenz der Toplogie in der Schnittstellenimplementierung und im CAD–System kann entstehen, wenn das CAD–Modell verändert wird und dadurch in der Schnittstelle Verweise auf nicht mehr existierende CAD–Strukturen gespeichert sind. Eine Variation des Geometriemodells kann während einer integrierten Berechnung auf zwei Weisen erfolgen: durch Austausch eines Teils einer Teilefamilie und durch Verändern von Parametern. Beim Austausch eines Teils wird als Ergebnis des Funktionsaufrufs die Referenz des neuen Teils zurückgegben und anstelle der bestehenden eingesetzt. Damit ist die Konsistenz für die übergeordneten Objekte und für das ersetzte Objekt gewahrt. Lediglich die untergeordneten Objekte müssen neu verknüpft werden. Dies kann manuell oder, sofern

die Strukturen benannt wurden, anhand des Namens durch das System erfolgen. Bei der Variation von Parametern werden die neuen Parameterwerte zunächst nur in der Schnittstellenimplementierung gespeichert. Erst nach einem entsprechenden Funktionsaufruf (*update()*) werden die Parameter in das CAD–System übertragen und das CAD–Modell wird regeneriert. Die Implementierung der update–Funktion überprüft nach erfolgreicher Regenerierung alle Referenzen des betroffenen Bauteils. Sollte ein Objekt auf ein nicht mehr gültige Strukter im CAD–System verweisen, muß dieses Objekt neu verknüpft werden.

6.3 Berechnungssystem

6.3.1 Schraubenberechnungsprogramm BOLT

BOLT wurde am Institut für Maschinenkonstruktion - Konstruktionstechnik für die Berechnung von Schraubenverbindungen entwickelt. Mit sieben Modulen können Zylinderverbindungen, Balkenverbindungen, Apparateflansche, verschraubte Kreisplatten, rotationssymmetrische Mehrschraubenverbindungen, Rechteckflansche und Kragflansche berechnet werden. Allgemeine, für alle Module geltende Daten, werden als *Werkstoffdaten*, *Standardwerte* und *Berechnungswerte* in Dateiform zur Verfügung gestellt. Spezielle modulspezifische Daten werden in einer Norm- und Hilfswertedatei gespeichert.
Das Programm arbeitet parameterorientiert im Batchbetrieb. Die Eingangsparameter werden aus der Eingabedatei gelesen, auf syntaktische Richtigkeit und Konsistenz überprüft und anschließend an das aufgerufene Modul übergeben. Bei einer fehlerhaften Eingabedatei wird ein Hinweis in die Ergebnisdatei geschrieben und die Berechnung abgebrochen. Bei vollständigen und richtigen Eingabedaten wird die Berechnung durchgeführt und die Ergebnisse werden in einer lesbar formatierten Textdatei gespeichert (siehe Bild 6.6).
Die Berechnungsmodule sind in FORTRAN77 implementiert und dadurch auf verschiedenen Plattformen übersetzbar. Für das Betriebssystem Windows 95 wurde inzwischen eine komfortable Benutzungsoberfläche für die Erstellung der Eingabedateien und die Visualisierung der Ergebnisse erstellt.

6.3.2 Struktur der Schnittstelle

Die Schnittstelle zum Schraubenberechnungsprogramm BOLT arbeitet parameterorientiert. Es werden *Inputparameter* für Eingabewerte, *Outputparameter* für Ergebniswerte und *Temporary Parameters* für Zwischenergebnisse unterschieden. Die Zwischenergebnisse haben nur innerhalb des Rechenschemas eine Bedeutung, können aber bei der Durchführung von Berechnungen von Interesse sein, um das Zustandekommen eines Ergebnisses zu verdeutlichen. Die von BOLT bereitgestellten Partialmodelle sind die einzelnen Module für die verschiedenen berechen-

```
**********************************************************************
*                                                                    *
*         AUSLEGUNG EINER SCHRAUBENVERBINDUNG - HAUPTBELEG           *
*         ************************************************           *
*                                                                    *
* 1. BENUTZTER BERECHNUNGSMODUL ---- VDI 2230 ----       BOLT 4.2    *
*                                                                    *
* 2. AUSGABEDATEN                                                    *
* ------------------------------------------------------------------ *
* I SCHRAUBE           D    = I M  10.00    I STEIGUNG  P =   1.50 I *
* I SCHRAUBENLAENGE         I    69.00 MM I GEWINDEL. B =  28.00 I *
* I FESTIGKEITSKLASSE GUETE = I    12.90    I                      I *
* I--------------------------+-------------+----------------------I *
* I MUTTER                  I M  10.00    I STEIGUNG  P =   1.50 I *
* I SCHEIBE           DSCH = I    21.00 MM I        HSCH =   2.00 I *
* I--------------------------+-------------+----------------------I *
* I BOHRLOCHDURCHMESSER  DB = I    11.00 MM I                      I *
* I KOPFKREISDURCHMESSER DK = I    16.00 MM I                      I *
* I--------------------------+-------------+----------------------I *
* I ANZIEHFAKTOR      ALPHA = I     1.60    I NACH VDI 2230        I *
* I--------------------------+-------------+----------------------I *
* I ANZIEHMOMENT,THEORETISCH  I    84.72 NM I                      I *
* I EINSTELLMOMENT-10.PROZENT I    76.25 NM I                      I *
* I--------------------------+-----------------------------------I *
* I ANZIEHVERFAHREN :         I DREHMOMENTENGESTEUERT MIT         I *
* I                           I DREHMOMENTENSCHLUESSEL            I *
* I--------------------------+-----------------------------------I *
* I SICHERHEITEN GEGEN  -     I             I                     I *
* I - DAUERBRUCH              I     0.83    I VORH. SPAN. =  63.22 I *
* I - FLAECHENPRESS. UEBERS.  I     4.07    I FLAECHENPRES. 221.33 I *
* I - LOCK.D.VERB.(FVMIN/FPA) I     5.21    I FVMIN IN KN =  27.80 I *
* I - QUERKRAFTUEBERS.        I     6.43    I REIBFAKTOR  =   0.20 I *
* ------------------------------------------------------------------ *
*                                                                    *
* 3. SCHRAUBENKRAEFTE UND SICHERHEITEN                               *
*     X     - X-KOORDINATE DER SCHRAUBE                              *
*     P1    - AUSLASTUNG DER SCHRAUBE NACH DEM VORSPANNEN            *
*     P2    - AUSLASTUNG DER SCHRAUBE BEI BETRIEBSLAST               *
* ------------------------------------------------------------------ *
* I    X     I   FKRES  I   FSA    I  FSMAX   I   P1    I   P2    I  *
* I    MM    I    N     I    N     I    N     I  PROZ.  I  PROZ.  I  *
* I----------+----------+----------+----------+---------+---------I  *
* I     0.00 I   22468. I    2709. I   48500. I   90.00 I   94.33 I  *
* ------------------------------------------------------------------ *
**********************************************************************
```

Bild 6.6: Beispiel eines BOLT Ergebnisausdrucks

baren Verbindungstypen.
Von der CORBA–Schnittstelle werden die Werte aller Parameter gespeichert. Bei Durchführung einer Berechnung mit dem Aurruf *proceed()* werden die Werte der *Inputparameter* in die Eingabedatei von BOLT geschrieben und das Programm gestartet. Nicht gesetzte, aber benötigte Werte werden dabei aus einer Liste von Standardwerten bezogen. Anschließend werden mit den Ergebnissen der Berechnung die Werte der *Resultparameter* und der *Temporary Parameters* überschrieben. Als Variation kann dem Aufruf *proceed()* eine Liste von Parametern übergeben werden, die für die aktuelle Berechnung verwendet werden soll.

6.3.3 Implementierungsdetails

Programmodule

Ähnlich wie beim Geometrie–Server besteht das Berechnungssystem aus einem Standard–Serverprogramm und einer dynamisch gebundenen Bibliothek mit der eigentlichen Objektimplementierung. Anders als beim CAD–System wird hier die in C implementierte Schnittstellenimplementierung direkt mit den in FORTRAN implementierten Moduldateien von BOLT zusammengebunden. Dies ist möglich, weil der C–Compiler und der FORTRAN–Compiler gleichartige Moduldateien erzeugen. Auf diese Weise sind direkte Funktionsaufrufe zwischen den beiden Bibliotheken möglich. Werden bei solchen Funktionsaufrufen Speicheradressen übergeben, können auch Speicherbereiche, die von BOLT reserviert wurden, in der Schnittstellenimplementierung ausgewertet werden. Bild 6.7 zeigt schematisch die einzelnen Programmodule und ihre Verknüpfungen.

Funktionsweise

Die Parameter und ihre aktuellen Werte werden von der Schnittstellenimplementierung zwischengespeichert. Eine Berechnung wird durch den Funktionsaufruf *proceed()* gestartet. Da BOLT die Eingabedaten in Form einer Datei benötigt, werden für die Berechnung sämtliche Parameter entsprechend formatiert in eine Eingabedatei geschrieben. Anschließend wird die Hauptroutine von BOLT aufgerufen. Diese analysiert die Eingabedatei und legt die Parameter in mehreren Feldern im Speicher ab. Die eigentliche Berechnung erfolgt mit den im Speicher befindlichen Werten und auch die Ergebnisse werden im Speicher abgelegt, bevor sie in die Ergebnisdatei geschrieben werden. Bevor die Hauptroutine von BOLT beendet wird, erfolgt ein Sprung in die Objektimplementierung. Bei diesem Funktionsaufruf werden die Speicheradressen der zuvor angelegten Parameterfelder übergeben. Für die zuvor definierten Ergebnisparameter werden die Speicherbereiche ausgewertet und die Ergebnisse in die entsprechenden Objekte übertragen. Ist dies erfolgt, wird die Hauproutine von BOLT beendet und der Speicherbereich für die Felder wieder freigegeben. Nach Beendigung des Funktionsaufrufs

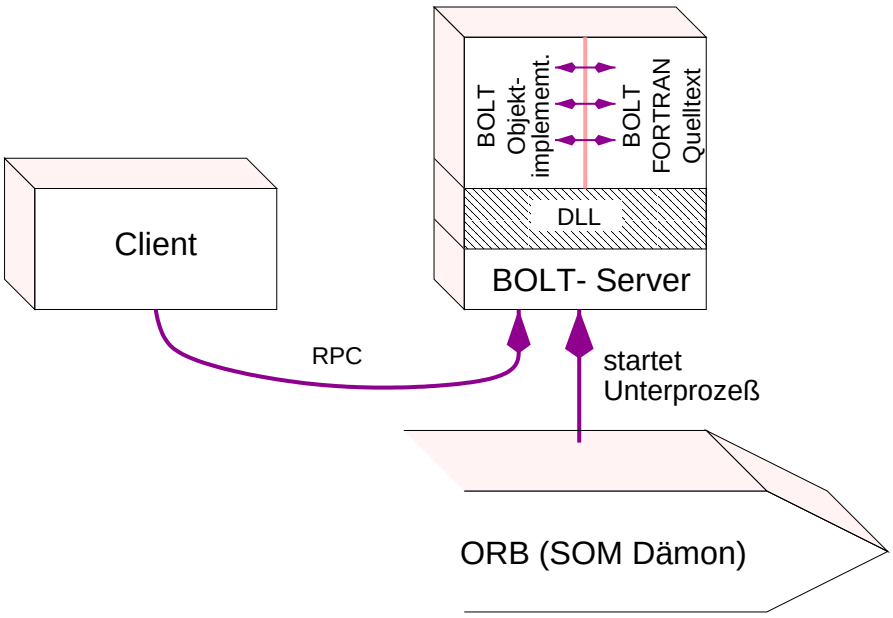


Bild 6.7: Programmodule des Berechnungs–Servers

proceed() stehen in den Parameterobjekten der Schnittstellenimplementierung die aktuellen Ergebniswerte der Berechnung.
Als schwierig erwies sich die Behandlung von Fehlern bei unvollständigen oder ungültigen Eingabeparametern, da diese Ereignisse von BOLT zwar erkannt werden, aber nur durch eine vorzeitige Programmbeendigung und einen entsprechenden Kommentar im Ergebnisausdruck behandelt werden. Im Ergebnis äußerten sich solche Programmabbrüche meist dadurch, daß ein großer Teil der Ergebnisparameter den Wert null hatte. Um dem Anwender die Möglichkeit zu geben, bei zweifelhaften Ergebnissen die Ergebnisdatei direkt einzusehen, kann der Inhalt der Ergebnisdatei mit einer zusätzlichen Methode direkt abgerufen werden. Eine ausführliche Fehlerbehandlung könnte über eine Anpassung der Fehlerbehandlungroutinen von BOLT erfolgen. Erfolg oder Mißerfolg einer Berechnung könnten als Ergebnis des *proceed()*-Aufrufs übertragen werden.

6.4 Modellmanager

Im Modellmanager werden die Objekte aus den Applikationen zusammen mit Datenbankinformationen auf semantischer Ebene zu einem *Strukturmodell* verknüpft. Inhaltlich setzt sich das Strukturmodell aus den in Bild 5.3 auf Seite 90 gezeigten Informationen zusammen. Dabei werden die geometrischen Objekte aus dem CAD–Modell, die Rechenergebnisse aus BOLT und die übrigen Daten aus einer Datenbank oder vom Benutzer bezogen. Der Modellmanager ist damit der

Client für die beiden Applikationen und die Datenbank. Das im Modellmanager gehaltene Modell stellt das aktuelle Arbeitsmodell für den Benutzer dar und ist nicht zu jeder Zeit konsistent mit beiden Partialmodellen. Die Konsistenz kann durch gezielte Interaktion des Benutzers wiederhergestellt werden. Durch diesen Mechanismus kann eines der Modelle variiert und optimiert werden, ohne daß das andere Modell ständig nachgeführt werden muß. Beispielsweise kann das Rechenmodell bezüglich des Schraubendurchmessers optimiert werden und anschließend die geometrische Realisierbarkeit im Geometriemodell überprüft werden.
Bei Aufruf des Modellmanagers wird der Graph des gewählten Modells im Speicher angelegt und mit den ausgewählten Partialmodellen verknüpft. Dazu wird entweder versucht, das Modell anhand der im Relationsgraphen gegebenen Bedingungen zu identifizieren, oder die Zuordnung der Komponenten, Bezugselemente und Parameter erfolgt durch den Benutzer. Nach vollständiger Verknüpfung der Modelle können durch Variation einzelner Parameter die Partialmodelle variiert werden. Auch hier ist eine vollständig automatisches Nachführen der Partialmodelle nicht immer möglich, sodaß eine Benutzerinteraktion erforderlich werden kann.

6.5 Ablauf einer integrierten Berechnung

Dieser Abschnitt beschreibt den prinzipiellen Ablauf einer Berechnung, die Funktion der einzelnen Systemkomponenten und die erforderliche Kommunikation. Der Ablauf wird zuerst an einem Interaktionsdiagramm erklärt und anschließend an einem Anwendungsbeispiel veranschaulicht.

6.5.1 Interaktionsdiagramm

Die an einer integrierten Berechnung beteiligten Komponenten und deren Zusammenwirken sind in Bild 6.8 dargestellt. Über das in Java implementierte *Control Center* können alle Anwendungen im System gestartet und auch die Modelle ausgewählt und verknüpft werden. Das Arbeitsmodell der integrierten Berechnung wird vom ebenfalls in Java implementierten Modellmanager gehalten und dargestellt. Die Methodenaufrufe des Modellmanagers gehen über den *Object Request Broker (ORB)* an die entsprechenden Serverprozesse.

1. **Aufruf eines Geometriemodells.** In einem Auswahlfenster des *ControlCenters* kann der Benutzer das CAD–System und ein Modell wählen. Im *Implementation Repository* der CORBA Umgebung ist eine Zuordnung von Modelltypen zu Applikationen und damit zu Serverprogrammen gespeichert. Anhand dieses Eintrages wird ein geeigneter, lokal implementierter Server ausgewählt.

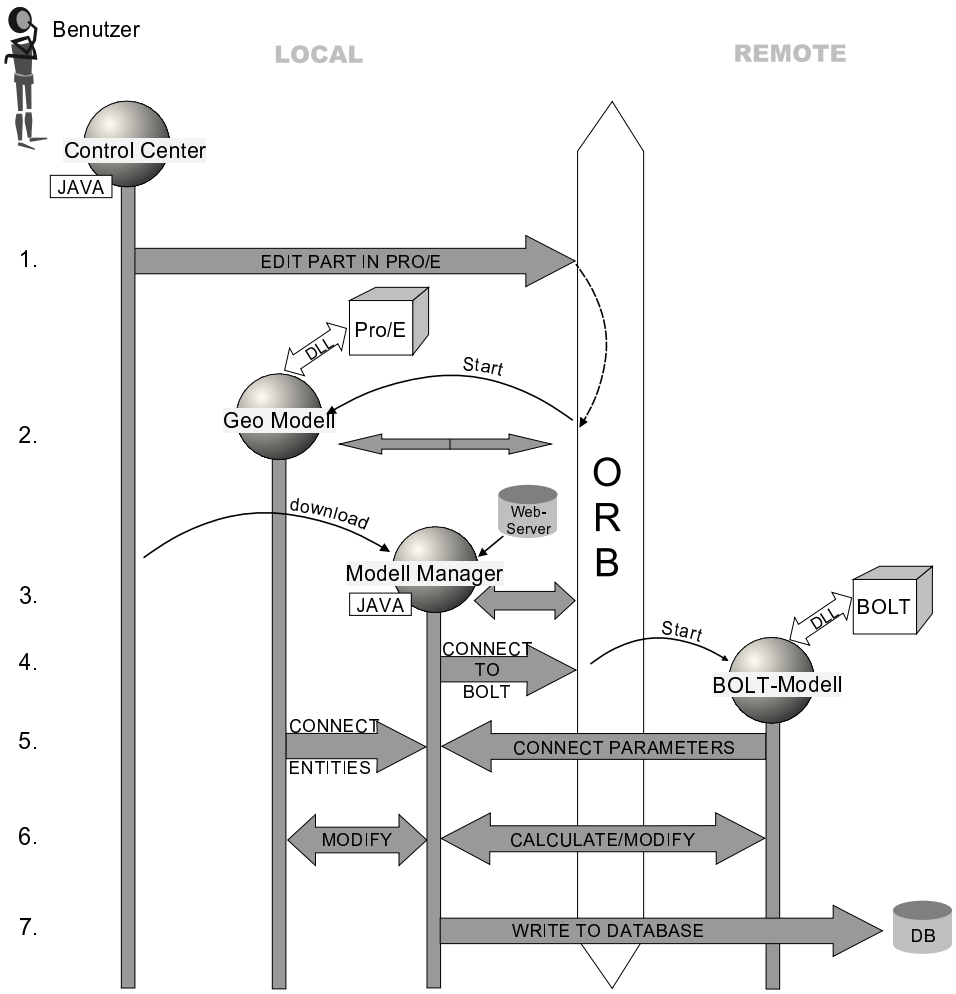


Bild 6.8: Interaktionsdiagramm einer integrierten Berechnung

2. **Start des CAD-Systems.** Der ORB startet den Geometriemodellierer, indem er den zuvor identifizierten Server startet. Der Server hält über Interprozeßkommunikation Kontakt zum ORB und lädt die als dynamisch gebundene Bibliothek implementierte Pro/TOOLKIT Applikation. Der Modellmanager veranlaßt den Geometrieserver das ausgewählte Geometriemodell zu laden.

3. **Verknüpfen mit einem Rechenmodell.** Aus der Liste der verfügbaren Berechnungsmethoden wählt der Benutzer im *ControlCenter* eine für sein Problem geeignetes Berechnungsprogramm und dazu ein passendes Modellschema. Anschließend startet er den *Modellmanager*, um eine Verknüpfung zwischen dem lokalen Geometriemodell und dem gewählten Rechenmodel herzustellen.

4. **Start des Berechnungssystems.** Vom *ModellManager* wird eine Anfrage an den ORB gesendet, um einen Zugriff auf das Berechnungssystem zu erhalten. Mit Hilfe des *Implementation Repository* lokalisiert der ORB den entsprechenden Server und startet ihn. Der neue Server ist ebenfalls über Interprozeßkommunikation mit dem ORB verbunden und lädt die dynamisch gebundene BOLT-Bibliothek nach.

5. **Verknüpfung mit dem Strukturmodell.** Durch Analyse der im *Modell-Manager* gespeicherten Bedingungen und Relationen und durch Auswahl der entsprechenden Komponenten im Geometriemodell werden die Komponenten des Strukturmodells mit den Elementen in den Servern verknüpft.

6. **Berechnung und Variation der Modelle.** Mit dem jetzt vorliegenden Strukturmodell kann die Berechnung durchgeführt werden. Veränderungen im Strukturmodell müssen in beiden Partialmodellen nachgeführt werden. Sofern die betroffenen Objekte über Bedingungen mit dem Strukturmodell verknüpft sind, ist eine automatische Variation der Partialmodelle möglich. Anderenfalls muß der Benutzer die Änderungen manuell vornehmen.

7. **Archivierung der Ergebnisse.** Die Veränderungen am Geometriemodell werden vom CAD-System gespeichert. Das bei der Berechnung erzielte Ergebnis wird vom ModellManager als Datei gespeichert.

6.5.2 Anwendungsbeispiel

Während im vorigen Abschnitt die Interaktionen der Systemkomponenten beschrieben wurden, steht in diesem Abschnitt die Darstellung des Systems aus Benutzersicht im Vordergrund. Im folgenden Beispiel wird beschrieben, wie mit dem System eine bestehende Pleuelschraubenverbindung (siehe Bild 6.9) berechnet und variiert werden kann. Das Vorgehen wird anhand der durchzuführenden Arbeitsschritte erläutert. Die Funktionsweise der einzelnen Softwarekomponenten wird an den entsprechenden Stellen im Text erläutert. Folgende Arbeitsschritte sind durchzuführen:

1. **Start des ControlCenters.** Über das ControlCenter werden die im System vorhandenen Programme gestartet und die Modelle ausgewählt. Mit dem Aufruf des ControlCenters wird die CORBA–Umgebung initialisiert. Die Benutzungsschnittstelle enthält die Auswahlmenüs für die Systeme und die zu den Systemen verfügbaren Modelle. Über drei Bedienfelder am Kopf des Fensters können die Applikationen einzeln oder gemeinsam gestartet werden. Bild 6.10 zeigt die Benutzungsoberfäche des ConrolCenters mit einem aufgeklappten Menü zur Auswahl des CAD–Systems. Der Benutzer wählt hier das Modell “pleuel_bg” , das im CAD–System Pro/Engineer dargestellt und mit dem Modul für zylindrische Einschraubenverbindungen (VDI 2230) des Berechnungsprogramms BOLT berechnet werden soll.

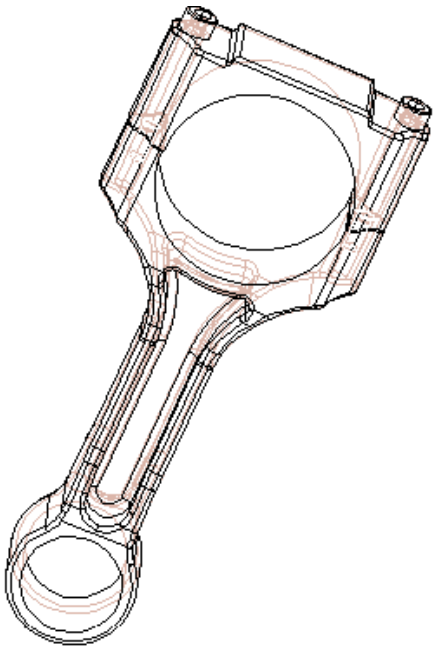


Bild 6.9: Baugruppe mit der zu berechnenden Schraubenverbindung

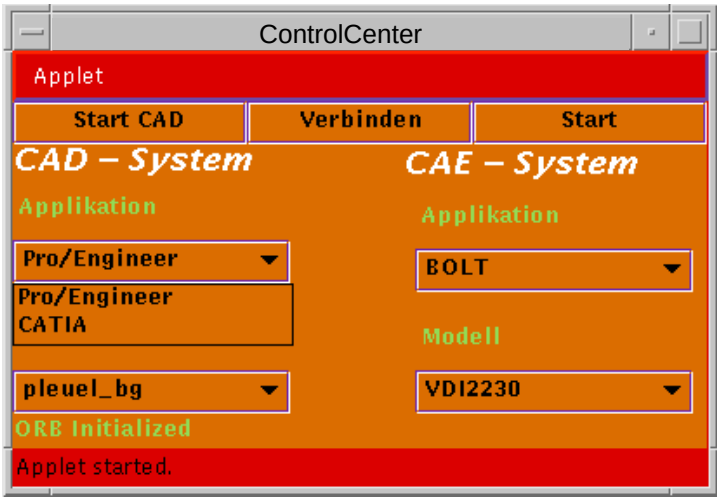


Bild 6.10: Benutzungsoberfläche des ControlCenters

2. **Aufruf des ModellManagers.** Durch Anwählen des "Verbinden" Bedienfeldes werden die Systeme mit der aktuell ausgewählten Konfiguration von Modellen und Systemen gestartet. Anschließend wird mit dem *ModellManager* die Benutzungsoberfläche für die integrierte Berechnung gestartet.

ModellManager

Der ModellManager liefert die Benutzungsoberfläche für die integrierte Berechnung. Mit ihm können die Bestandteile des Strukturmodells und die Verknüpfungen zwischen den Objekten überwacht und visualisiert werden. Das Arbeitsfenster enthält auf der linken Seite die Darstellung der Baustruktur in Form eines Baumes (vergleiche Bild 6.11). Auf der rechten Seite sind die in Gruppen geordneten Parameter der aktuell ausgewählten Baustrukturkomponente und die Ergebnisse der zuletzt durchgeführten Berechnung dargestellt. Am unteren Rand des rechten Bildschirmbereichs kann die gewünschte Rechenoperation gewählt werden. Die Auswahl verändert die in den anderen Fenstern dargestellten Informationen und legt die jeweiligen Bedingungen für die Verknüpfungen mit den Partialmodellen fest.

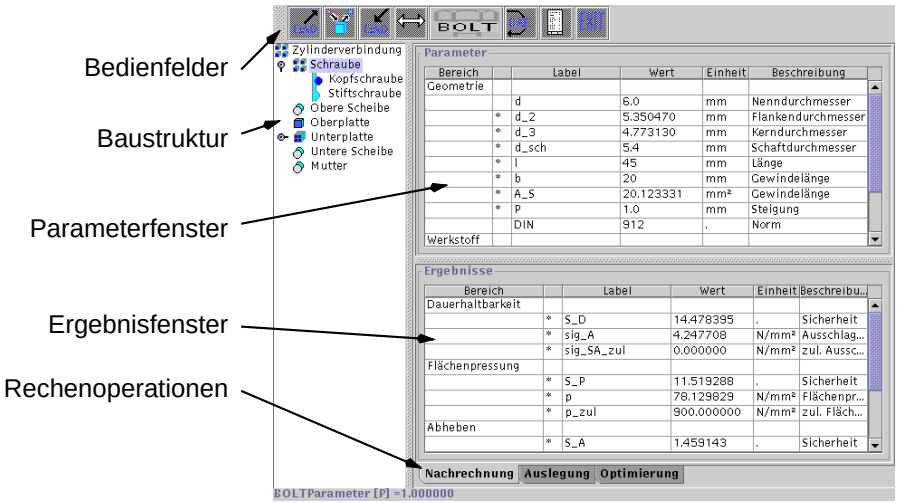


Bild 6.11: Benutzungsoberfläche des ModellManagers im Überblick

In der Baustrukturdarstellung sind die für eine Berechnung erforderlichen Komponenten mit einem Quadrat und die optionalen Komponenten mit einem Kreis gekennzeichnet (siehe Bild 6.12). Kompenenten, die mit einem Element des Geometriemodells verknüpft sind, werden farbig markiert. Für Komponenten, die in verschiedenen Ausprägungen vorkommen können, beispielsweise eine Schraube als Kopfschraube oder Stiftschraube, wurden für

die Darstellung die Symbole unterteilt. Durch Anwählen des Symbols wird der Baum aufgeklappt und die möglichen Ausprägungen werden angezeigt.

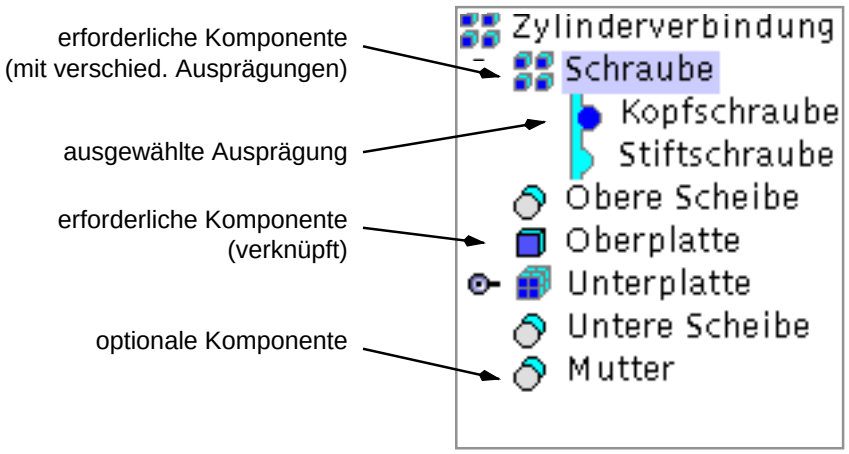


Bild 6.12: Erläuterung der Symbole für die Baustrukturelemente

Mit dem Aufruf des ModellManagers wird das Strukturmodell in den Speicher geladen. Das Strukturmodell bildet das Modellschema, in dem alle durch das gewählte Programm berechenbaren Modelle abgebildet sind.

3. **Auswahl einer Rechenoperation.** Im ModellManager wählt der Benutzer die gewünschte Rechenoperation, in diesem Fall die Durchführung eines Festigkeitsnachweises. Daraufhin werden im Ergebnisfenster die bei dieser Operation erzielbaren Ergebnisse dargestellt und die Bedingungen im Modell angepaßt.

4. **Verknüpfung mit dem Geometriemodell.** Durch Auswählen des entsprechenden Bedienfeldes wird die Prozedur zum Verknüpfen des Strukturmodells mit dem Geometriemodell gestartet. Der Benutzer erhält daraufhin ein Fenster mit den im Geometriemodell vorhandenen Bauteilen und den im Strukturmodell zu besetzenden Komponenten. Durch Auswählen den "Connect" Bedienfeldes (siehe Bild 6.13) werden die markierten Elemente miteinander verknüpft und die Berechnungsparameter aus dem Geometriemodell ermittelt.

   Norm- und Standardteile werden über den Teilenamen identifiziert. Die Parameter von bekannten Standardteilen werden automatisch vom System erkannt und in das Rechenmodell übertragen. Die Parameter der übrigen Teile müssen durch den Benutzer im Geometriemodell identifiziert werden. Wenn ein Parameter im Geometriemodell als eigenständiger Parameter vorhanden ist, hat der Benutzer die Möglichkeit, diesen direkt zu auszuwählen.

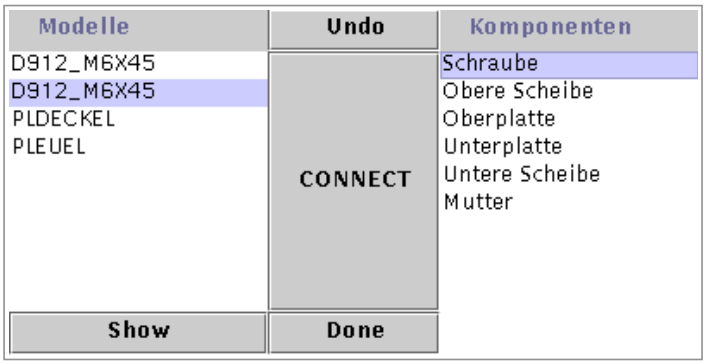


Bild 6.13: Dialog zum Verknüpfen der Modelle

Anderenfalls wird der Wert des Parameters aus einer geometrischen Relation eines oder mehrerer geometrischer Objekte bestimmt. In diesem Fall muß der Benutzer die entsprechenden Objekte auswählen. Bild 6.14 zeigt die Auswahlmöglichkeiten für den Parameter “Plattenhöhe” der Komponente “Oberplatte” und die zugehörige Darstellung im CAD–System. Der Wert kann durch den Abstand zweier Flächen, die Länge einer Kante oder direkt durch einen Parameter (“DIMENSION”) bestimmt werden. Die direkte Verknüpfung von Parametern hat den Vorteil, daß der Wert durch den ModellManager geändert werden kann.

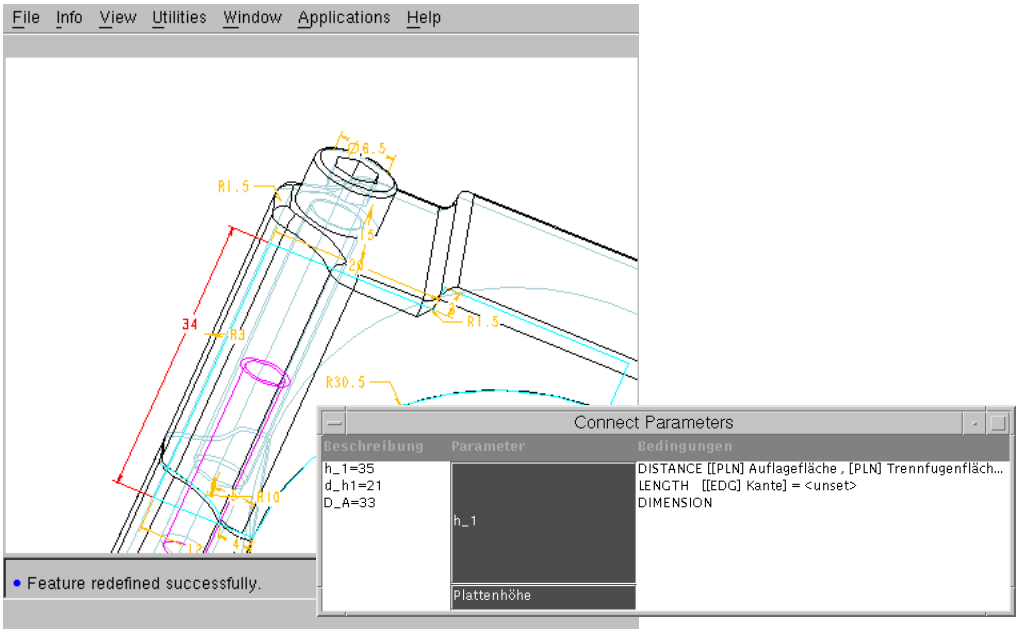


Bild 6.14: Dialog zum Verknüpfen der Parameter

Am Ende dieses Schrittes ist das Strukturmodell in einer gültigen Konfiguration mit dem Geometriemodell verknüpft.

5. **Berechnung.** Durch Anwählen des "Berechnen" Feldes wird aus der aktuellen Konfiguration ein Berechnungsdatensatz erstellt. Dafür werden die Bedingungen und Relationen für die Erstellung der Berechnungsparameter ausgewertet. Die Ergebnisse der Berechnung werden in der Ergebnistabelle dargestellt. Werte die sich seit der letzten Berechnung geändert haben, sind durch einen Stern gekennzeichnet. Parameterwerte, die nicht vom Benutzer eingegeben werden und vom Berechnungsprogramm ermittelt wurden, werden ebenfalls in das Strukturmodell übertragen und gegebenenfalls als geändert markiert. Bild 6.15 zeigt das Parameterfenster und das Ergebnisfenster nach der Berechnung. Die Detaildaten der Schraubengeometrie, Flankendurchmesser und Kerndurchmesser wurden für den gewählten Nenndurchmesser durch BOLT ermittelt. Optional kann die von BOLT erzeugte Ergebnisdatei durch Anwählen des Bedienfeldes angezeigt werden (vergleiche Bild 6.6 auf Seite 116). Diese Option ist hilfreich, wenn die Berechnung wegen fehlerhafter Eingabeparameter abgebrochen wurde. In diesem Fall enthält die Ergebnisdatei aussagefähige Fehlermeldungen.

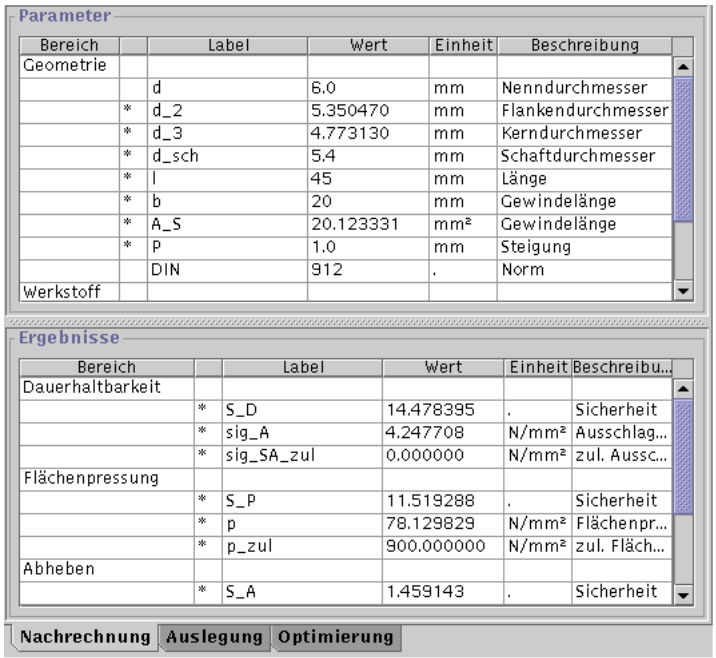


Bild 6.15: Parameterfenster der Schraube und Ergebnisfenster nach der Berechnung

6. **Variation des Rechenmodells.** Die Parameter des Strukturmodells können durch den Benutzer verändert werden. Im Modellmanager stehen für

einige Parameter Listen mit vordefinierten Einträgen zur Verfügung. Bei den meisten anderen Parameter sind Gültigkeitsbereiche vordefiniert, die über automatisch aufklappende Informationsfenster angezeigt werden (siehe Bild 6.16). Durch Ändern von Parametern ist das Strukturmodell zunächst inkonsistent mit den Partialmodellen. Für die Aktualisierung der Partialmodelle stehen in der Menüleiste entsprechende Felder zur Verfügung.

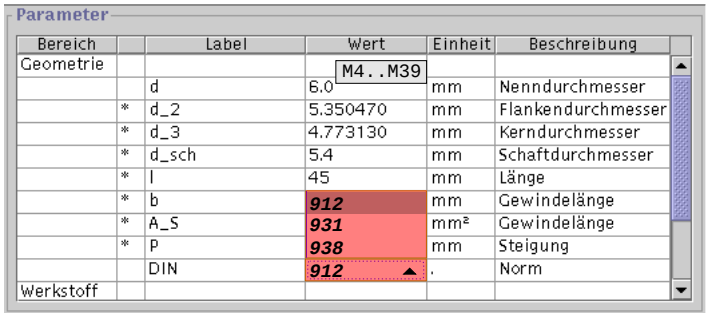


Bild 6.16: Vordefinierte Optionen und Gültigkeitsbereiche der Parameter

7. **Variation des Geometriemodells.** Das Geometriemodell kann durch Ersetzen von Normteilen und durch Verändern von Parameterwerten modifiziert werden. Wird in der Menüleiste das Feld “Ersetzen eines Bauteils” gewählt, erscheint, sofern es sich um ein Teil aus einer Teilefamilie handelt, eine Liste der weiteren in dieser Familie verfügbaren Teile. Durch Auswählen einer Variante wird das Bauteil im Geometriemodell ersetzt und die geänderten Parameter werden automatisch in Strukturmodell übernommen. Nicht standardisierte Komponenten können durch Setzen der Parameter geändert werden. Durch Wählen des Bedienfeldes wird versucht, alle Geometrieparameter des Strukturmodells im Geometriemodell nachzuführen. Dies ist nur möglich wenn die Parameter zuvor direkt verknüpft wurden. Im gezeigten Beispiel wurde der Außendurchmesser des verspannten Körpers durch den Abstand zweier Flächen definiert und kann daher nicht im Geometriemodell nachgeführt werden. Die betroffenen Parameter werden im Strukturmodell entweder mit einem Stern als geändert oder mit einem “M” als nicht änderbar gekennzeichnet (siehe Bild 6.17).

8. **Abschluß der Berechnung.** Auf Wunsch wird dem Benutzer die Ergebnisdatei von BOLT übermittelt. Die Änderungen im Geometriemodell bleiben erhalten und werden zusammen mit dem Modell gespeichert.

Parameter

Bereich		Label	Wert	Einheit	Beschreibung
Geometrie					
	*	h_1	34.0	mm	Plattenhöhe
	*	d_h1	8.5	mm	Bohrungsdu...
	M	D_A	32	mm	Außendurch...
Werkstoff					
		Wst_1	C45	.	Werkstoff

Bild 6.17: Parametertabelle nach Aktualisierung des Geometriemodells

6.6 Variationsmöglichkeiten

In der implementierten Version des ModellManagers sind die Bedingungen und Relationen der berechenbaren Modelle als Programmcode implementiert. Dadurch ist eine sehr spezielle, auf genau die beiden verwendeten Programme abgestimmte Lösung entstanden. Die Implementierung ist aber so gestaltet, daß der Programmteil für die Verwaltung und die Darstellung der Modelle unabhängig ist von dem Teil, der für den Modellaufbau verantwortlich ist. Ersetzt man den Programmteil für den Modellaufbau, läßt sich der ModellManager auch auf andere Berechnungssysteme anpassen. Folgende Varianten sind hier vorstellbar:

Scripte für Modellschemata. Eine flexible Lösung ergibt sich, wenn die Informationen zur Verknüpfung der Modelle in Scripten gespeichert werden. Der ModellManager müßte dann um einen Parser für die gewählte Scriptsprache erweitert werden. Die Scripte würden vom Anbieter der Berechnungsmethoden zur Verfügung gestellt werden.

Scripte mit Formeln. Oft sind für die Verknüpfung von Parametern zusätzlich mathematische Rechenoperationen erforderlich. Die Bearbeitung einer in einem Script gespeicherten Formel zur Laufzeit erfordert eine zusätzliche Softwarekomponente. Die Werte der in Formel gespeicherten Parameter müssen ermittelt und entsprechend der angegebenen Rechenoperationen verknüpft werden. Eine Implementierung eines solchen Programms wird in [Wöl98] als *Solver* bezeichnet. In dem dort beschriebenen System ermöglicht der Solver die Berechnung von mathematischen Formeln und überwacht modellübergreifend bestehende Bedingungen zwischen den Parametern. Steht in einem System ein hinreichend leistungsfähiger Solver zur Verfügung, können auch vollständige Berechnungsalgorithmen als Scripte übertragen werden. In diesem Fall kann der Anwender die Berechnungsmethode in allen Teilen seinen Bedürfnissen anpassen.

Portable Programme. Methoden, die aus komplexen Algorithmen bestehen, können auch mit Hilfe von JAVA-Applets in den lokalen Speicher geladen werden. Durch den standardisierten Zugriff über die CORBA Schnittstelle

kann das Applet direkt mit dem Geometriemodell arbeiten. Die Plattformunabhängigkeit von JAVA ermöglicht dabei die Portabilität sowohl des Programmcodes als auch der Benutzungsoberfläche. Eine Variation des Programms zur Laufzeit ist bei JAVA-Applets nicht möglich. Dennoch können zur Laufzeit Programmkomponenten, auch von verschiedenen Servern nachgeladen werden. Auf diese Weise ließen sich Methoden mit Komponenten aus verschiedenen Servern zusammensetzen.

Geometrieobjekte im Berechnungsserver. Wenn der Berechnungsserver so erweitert wird, daß er direkt mit den Geometrieobjekten arbeiten kann, entfallen sowohl die Scripte als auch die portablen Programme. Diese Struktur ist insbesondere dann von Bedeutung, wenn die Methode nicht dem Anwender überlassen werden soll.

6.7 Bewertung des Lösungsansatzes

In diesem Abschnitt wird das Integrationssystem in bezug auf die in Kapitel 4 gestellten Anforderungen bewertet. Eine Übersicht in Tabelle 6.1 zeigt, welche Forderungen erfüllt wurden, welche bedingt erfüllt wurden, welche prinzipiell mit CORBA erfüllbar wären, aber nicht umgesetzt wurden und welche nicht erfüllbar sind.

6.7.1 Kopplung der Modelle

Ein durchgängiges Produktmodell für die rechnerunterstützte Produktentwicklung, das von den betrachteten Programmen unterstützt wird, existiert zur Zeit nicht. Daher wurde in einem externen Clientprogramm ein Strukturmodell implementiert, das die Objekte aus dem CAD–System und dem Berechnungssystem zu einem integrierten Modell zusammenfügt.
Das Zusammenfügen der Objekte und das Zusammenstellen der Verknüpfungsinformationen waren sehr umfangreiche Arbeitsschritte. Durch die bei CORBA verwendete Kapselung der Schnittstelle wurden diese Arbeitsschritte wesentlich vereinfacht. Mit der Formulierung einer Schnittstellenbeschreibung in IDL können die Programmiersprache und die eigentliche Struktur einer Programmierschnittstelle verdeckt werden. Dadurch können auch Objekte aus verschiedenen Applikationen in einer Client–Anwendung zusammengefügt werden. Durch die Definition von Oberklassen für systemweit gültige Objekte läßt sich das Verhalten von Objekten vereinheitlichen. Beispielsweise können Methoden für den Aufruf und die Beendigung von Applikationen, definiert werden.
Aus der doppelten Datenhaltung bei der Kapselung der Applikationen entstehen Nachteile durch erhöhten Speicherplatzbedarf und die Gefahr von Inkonsistenzen. Außerdem verursachen Methodenaufrufe über CORBA höhere Laufzeiten als beispielsweise *Remote Procedure Calls*, da die Aufrufe von der CORBA–Umgebung

Modelle	**Prozesse**
⊕ Kopplung Geometriemodell mit Parametermodell	◐ Berechnung als Werkzeug beim Gestalten
⊕ Identifizierung von Norm- und Standardteilen	⊕ synchrone Anwendung der Programme
⊕ verschiedene Modellkonfigurationen	⊕ Kopplung von Methoden über das Intranet
⊕ Geometrie als Haupt-Arbeitsmodell	⊖ Kopplung von Methoden über das Internet
	⊕ Geometrieanalysen
Programme	⊕ Geometrievariation
◐ direkte Kopplung	⊖ Geometrieerzeugung
⊕ Nutzung vorhandener Programmierschnittstellen	⊕ Visualisierung der Arbeitsmodelle in den Appl.
⊕ Plattform AIX	◐ Visualisierung der Verknüpfungen
⊕ andere Plattformen integrierbar	⊖ Sicherheit gegen unberechtigen Zugriff
⊕ CAD-Schnittstelle in C	⊖ Sicherheit der Datenübertragung
◐ Berechnungsschnittstelle in FORTRAN	⊖ Zugriffskontrolle
	⊕ Benutzung durch einen Anwender
	⊖ Benutzung durch mehrere Anwender
	⊕ kein zusätzliches IT-Know-How erforderlich
	⊕ Auslegung, Nachrechnung, Optimierung

⊕ Forderung erfüllt ◐ Forderung bedingt erfüllt ⊖ Forderung nicht erf. prinzipiell erfüllbar ○ Forderung nicht erfüllbar

Tabelle 6.1: Erfüllbarkeit der gestellten Anforderungen

abgefangen und an den Empfänger weitergeleitet werden. Serverseitig kann es zu höheren Laufzeiten kommen, wenn aufgrund unterschiedlicher Semantik des internen Modells und der verwendeten Schnittstellenbeschreibung komplexe Datenstrukturen analysiert und neu aufgebaut werden müssen.

6.7.2 Kopplung der Programme

Mit der realisierten Kopplung über einen zusätzlichen Client konnte die Forderung nach einer direkten Kopplung nicht erfüllt werden. Dennoch konnte eine plattformübergreifende Integration realisiert werden.
Voraussetzung für eine Integration eines Servers ist das Vorhandensein einer CORBA–Laufzeitumgebung und die entsprechende Bibliothek mit den Schnittstellenimplementierungen. Für die Erstellung der Bibliotheken werden sogenannte *Emitter*, benötigt, die aus den Schnittstellenbeschreibungen die Methodenrümpfe in der Programmiersprache der zu kapselnden Applikation erzeugen. Mit den bestehenden Emittern für *C*, *C++*, *Smalltalk* und *Java* läßt sich ein großer Teil der existierenden Softwaresysteme integrieren. Wie im obigen Beispiel gezeigt, läßt sich auch vorhandener FORTRAN Code in das System einbinden.
Client–seitig werden neben der CORBA-Laufzeitumgebung die Deklarationen für die Umsetzung der Methodenaufrufe in die entsprechenden CORBA–Aufrufe be-

nötigt. Diese Deklarationen werden, ähnlich wie bei den Servern, über entsprechende Emitter generiert.

Die Verwendung von Java-Clients macht auch die client–seitige CORBA–Laufzeitumgebung überflüssig, da diese zusammen mit dem eigentlichen Applet über das Netzwerk bezogen werden kann. Damit wird auf dem Client–Rechner lediglich eine Java–Umgebung (*Virtual Machine*) benötigt.

Die in der oben beschriebenen Implementierung benötigten Softwarekomponenten sind in Bild 6.18 dargestellt. Dabei werden plattformgebundene und portable Bibliotheken unterschieden.

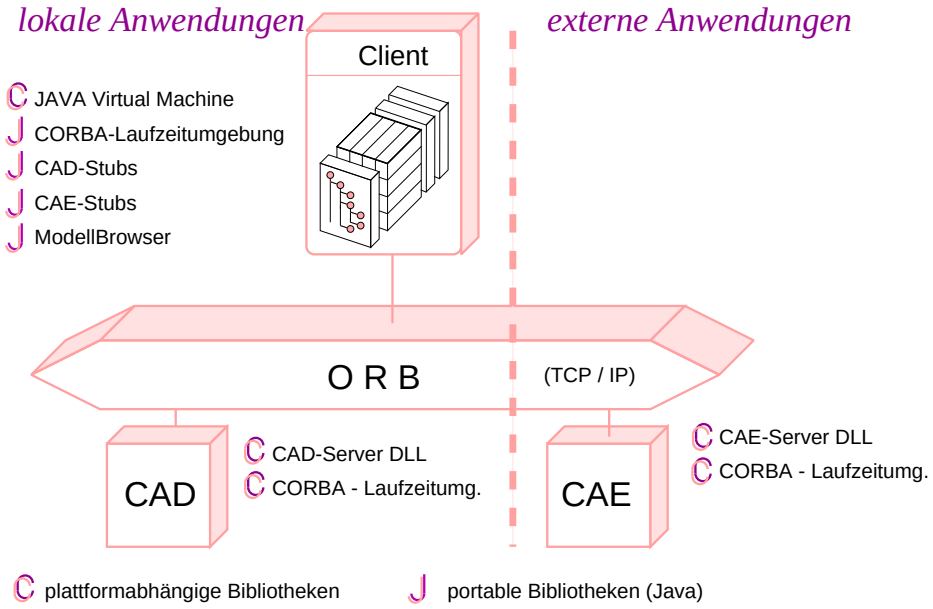


Bild 6.18: Erforderliche Softwarekomponenten für eine einfache Kopplung der Systeme

Die in Kapitel 4 geforderte Netzwerkfähigkeit ist mit CORBA gegeben. CORBA Objekte sind netzwerktransparent. Ein Client–Programm kann auf diese Weise ein Objekt eines anderen Servers verwenden, als wäre es lokal im Speicher vorhanden. Darüber hinaus werden als Bestandteil der CORBA–Umgebung Verzeichnisse der von den verschiedenen Servern zur Verfügung gestellten Objekte gehalten, sodaß sich der Anwender nicht um die tatsächliche Lokalisierung von Programmkomponenten kümmern muß. Er kann Objekte eines anderen Servers durch eine Anfrage an die CORBA–Umgebung erhalten. Die CORBA–Umgebung ermittelt aus dem entsprechenden Verzeichnis die Lokalisierung des Objekts, aktiviert es, und übergibt dem Client eine Referenz auf das Objekt, mit der der Client direkt arbeiten kann.

6.7.3 Kopplung der Prozesse

In der Clientanwendung können Berechnungssystem und Gestaltungssystem von einer einzigen Oberfläche aus bedient werden. Damit ist auch eine prozeßtechnische Kopplung realisiert. Die Funktionalität und die Benutzungsoberfläche sind abgestimmt auf die Anforderungen der integrierten Berechnung und laufen parallel zu den bereits bestehenden Systemen. Die Verwendung eines Berechnungssystems als Werkzeug in der Arbeitsumgebung des Geometriemodellierers ist damit nicht gegeben. Dafür müßte ein systemübergreifendes Prozeßmodell definiert und in beiden Systemen implementiert werden.

Auch hier sorgt der Mechanismus der Kapselung dafür, daß die Systeme zusammengeführt werden können. In der Client–Anwendung können Funktionsaufrufe an Objekte, die aus verschiedenen Servern stammen, gesendet werden. Darüber hinaus ermöglicht der *Life Cycle Service* die Erzeugung von Objekten, unabhängig von der Art und der Lokalisierung der zugehörigen Server.

Zusammenfassend sind in Bild 6.19 die Anforderungen dargestellt, die bei der Verwendung von CORBA automatisch erfüllt sind, oder deren Erfüllung durch CORBA unterstützt wird (vergleiche Bild 4.1 auf Seite 80). Aus dem Bild wird deutlich, daß mit Hilfe von CORBA eine plattformübergreifende und netzwerkübergreifende Kopplung möglich ist. Die weiteren Bereiche modelltechnische Kopplung und prozeßtechnische Kopplung werden zwar durch CORBA unterstützt, sind aber nicht automatisch bei der Verwendung von CORBA als Integrationsplattform realisiert.

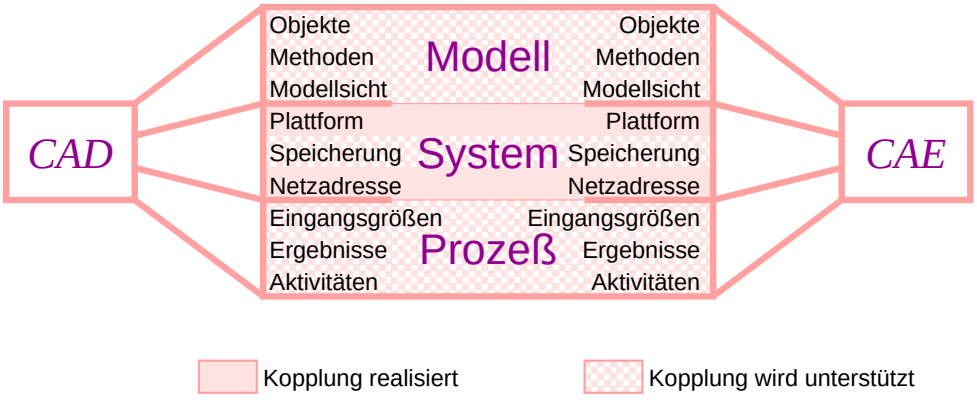


Bild 6.19: Anforderungen, die bei Verwendung von CORBA erfüllt werden können

6.8 Zusammenfassung

Das beschriebene Integrationssystem zeigt exemplarisch eine mögliche Form einer CORBA-basierten Kopplung eines parametrischen CAD–Systems und eines parameterorientierten Berechnungsprogramms. Das CAD–System Pro/ENGINEER und das Berechnungsprogramm BOLT wurden über objektorientierte Schnittstellen gekapselt und in eine CORBA Umgebung eingebunden. Die Struktur der Schnittstellen spiegelt dabei einen Auschnitt aus dem internen Modell der jeweiligen Applikation wider. Die Verknüpfung der Modelle erfolgt in einem separaten Client.
Die Bewertung des Ansatzes zeigt, daß die Forderungen nach Netzwerkfähigkeit und Plattformunabhängigkeit erfüllt sind.
Für die modelltechnische und die prozeßtechnische Kopplung waren zusätzliche Informationen erforderlich, die im Client enthalten sind. In einem im Client implementierten Strukturmodell für integrierte Berechnungen werden die Objekte aus den beiden Applikationen und die Verknüpfungen zwischen den Objekten abgebildet. Für die prozeßtechnische Kopplung wurden einige der in den Applikationen verfügbaren Funktionen im Client abgebildet.
Herauszustellen ist die Flexibiltät im Zugriff auf die internen Modelle, die sich durch die Kapselung der Applikationen ergibt. Im gezeigten Beispiel werden die in C und FORTRAN implementierten Applikationschnittstellen in einer JAVA–Applikation miteinander verknüpft und Objekte aus beiden Applikationen parallel verwendet. Der Ansatz ist damit geeignet, bestehende Programme als Server in moderne, objektorientierte Softwaresysteme zu integrieren, sofern eine CORBA–Laufzeitumgebung für die Serverplattform existiert und die Programme mit einer von CORBA unterstützten Programmiersprache zusammengebunden werden können.

7 Ausblick

7.1 CORBA als Integrationsplattform

Mit dem implementierten Integrationssystem wurde nachgewiesen, daß mit CORBA eine Kopplung bestehender Softwaresysteme realisierbar ist. Die Technik, Applikationen über neutrale Schnittstellenbeschreibungen zu kapseln, ermöglichte dabei auch die Zusammenführung zweier in bezug auf die Datenstrukturen, den Programmablauf und die Zugriffsmethoden auf die Modelle unterschiedlicher Systeme. Darüberhinaus ist es durch die Plattformunabhängigkeit und die Netzwerkfähigkeit von CORBA möglich, die auf verschiedenen Rechnern mit unterschiedlichen Betriebssystemen laufenden Programme zu koppeln. Die Möglichkeit, auch FORTRAN–Programme zu integrieren, macht CORBA zu einem geeigneten Werkzeug, eine große Zahl bestehender Berechnungsprogramme in eine moderene Systemumgebung zu integrieren.

Die wesentlichen Probleme der Integration lagen in der modelltechnischen Kopplung der Systeme. Die Schaffung eines einheitlichen, integrierten Produktmodells ist hierfür unumgänglich. Dabei steht der Ansatz der OMG, die STEP als Produktmodell für CAx–Anwendungen einsetzen will, im Konflikt mit den Interessen der CAD–System Hersteller, die bisher auf proprietäre Datenformate setzen. Für die Integration von Berechnungen muß das Produktmodell auch nichtgeometrische Daten enthalten können. Der umfassende Ansatz von STEP, sämtliche produktbezogenen Daten abbilden zu können, ist dafür eine gute Grundlage.

Durch die Kopplung von CORBA und JAVA ergeben sich interessante Möglichkeiten für die Bereitstellung von Berechnungsmethoden über das Internet. Von Berechnungsdienstleistern angebotene Methoden können über das Internet geladen werden und erhalten über die lokale CORBA–Umgebung Zugriff auf das lokale CAD–System. Dabei kann das JAVA–Applet das vollständige Berechnungsprogramm enthalten oder es kann die Dienste eines entfernt laufenden, über CORBA gekapselten Berechnungsservers in Anspruch nehmen. Voraussetzung für dieses Szenario sind eine in IDL formulierte, über CORBA zugängliche Schnittstelle des CAD–Systems und eine lokale CORBA–Laufzeitumgebung. Die Schaffung einer CORBA–konformen Schnittstelle durch die CAD–System–Hersteller wäre somit die Basis für eine netzwerkübergreifende Integration von Gestaltung und Berechnung.

7.2 Vernetzte Berechnungs–Competence–Center

Auf der Basis des oben beschriebenen Integrationssystems kann ein System vernetzter Berechnungs–Competence–Center für verschiedene Berechnungsaufgaben aufgebaut werden. Das System besteht aus einem in eine CORBA–Umgebung integrierten CAD-System und einer Reihe von Servern, auf denen spezialisierte Berechnungsmethoden bereitgestellt werden. Die Methoden werden als Werkzeuge bei der Durchführung der aktuellen Konstruktionsaufgabe eingesetzt und arbeiten mit dem lokalen Geometriemodell. Dabei können die Berechnungsprogramme die Parameter aus dem Geometriemodell ableiten oder das Geometriemodell variieren, erfordern aber selber kein persistentes Produktmodell. Nicht automatisch reproduzierbare Verknüpfungen und Benutzereingaben werden als Attribute dem Geometriemodell zugefügt oder als Datei zusammen mit dem Geometriemodell gespeichert.

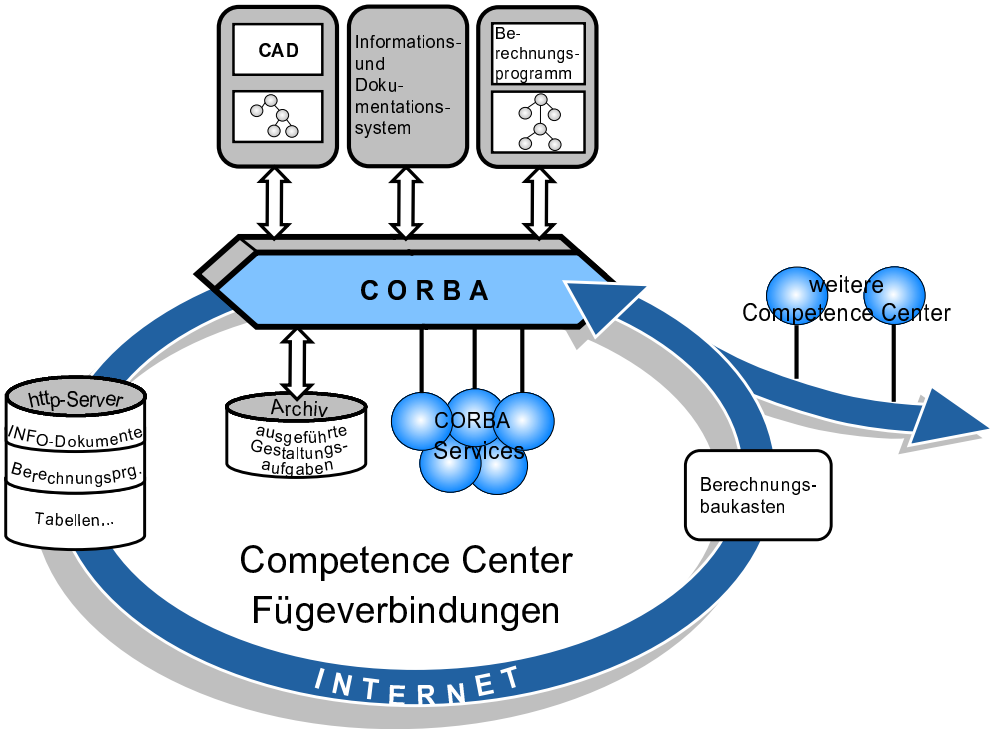


Bild 7.1: Einbindung eines Berechnungs–Competence–Centers in eine Netzwerkumgebung

Berechnungs–Competence–Center sind in diesem Szenario Anbieter von Berechnungsdienstleistungen und können sowohl innerhalb eines Unternehmens als auch durch externe Dienstleister aufgebaut werden. Dabei können entsprechend den Erfordernissen Applikationen verknüpft, einzelne Parameter oder vollständige

Modelle übertragen werden. Zur problemspezifischen Anpassung der Berechnungprgramme steht ein allgemeiner Methodenbaukasten zur Verfügung.
Entscheidend für den Nutzen eines Berechnungs–Competence–Centers ist die Bereitstellung von zusätzlichen Informationen über die Berechnungsmethoden, die eine qualifizierte Anwendung der Methoden ermöglichen. Je weniger der Anwender mit den Methoden des Anbieters vertraut ist, desto umfassender müssen die zusätzlichen Informationen sein. Insbesondere bei der Bereitstellung von Methoden durch externe Anbieter ist die Qualifizierung und Unterstützung der Anwender eine ernstzunehmende Aufgabe. Vorstellbar sind hier Schulungen durch die Anbieter, die den Kunden mit den speziellen Eigenschaften einer Methode vertraut machen und auch ein Vertrauensverhältnis zwischen Anbieter und Anwender schaffen, oder auch Call-Center, in denen Experten für Fragen zu bestimmten Methoden zur Verfügung stehen. Stehen die zusätzlichen Informationen weder als Onlinehilfe noch in Form einer persönlichen Beratung zur Verfügung, ist eine sinnvolle Anwendung eigentlich nur für einfache, allgemein bekannte Methoden möglich.
Durch die Verwendung von über IDL gekapselten Applikationen und CORBA als Kommunikationssystem wird die Verknüpfung der Modelle wesentlich vereinfacht. Die Schaffung einer standardisierten Zugriffsschnittstelle ermöglicht es den Entwicklern von Berechnungssoftware, Programme zu entwickeln, die mit allen Geometriemodellierern arbeiten. Entsprechend ermöglicht die Kapselung von Berechnungssoftware auch die Integration bestehender Programme, indem auch hier die Zugriffsmethoden, die Metadaten und die Strukturierung der Modelle vereinheitlicht werden. Insbesondere im Hinblick auf eine Prozeßplanung und Prozeßprotokollierung ist eine einheitliche Darstellung der Methoden von Vorteil. Darüber hinaus stehen durch die Basisdienste von CORBA eine Infrastruktur für die Lokalisierung, für den Transport und für eine netzwerktransparente Verwendung von Objekten, sowie standardisierte Sicherheitsmechanismen zur Verfügung.

7.3 Komponentenbasierte CAx–Systeme

Im Sinne der Objekttechnologie sollte sich ein Produktmodell aus beweglichen, plattform- und softwareumgebungsunabhängigen Softwarekomponenten zusammensetzen lassen. Entgegen dem bisherigen Entwicklungstrend, möglichst viele Inhalte in einem Softwaresystem abzubilden, müßten dann die Inhalte wieder auf verschiedene Modelle und damit auf verschiedene Applikationen verteilt werden. Allgemeine, in der Produktentwicklung benötigte Objekte und Methoden müßten in einer Klassenstruktur vorgegeben werden. Diese allgemeinen Objekte müßten Methoden zum Speichern und zum Verknüpfen untereinander enthalten und über Netzwerke transportierbar sein. Durch branchen- oder problemspezifische Erweiterungen können diese allgemeinen Objekte an jeden Anwendungsfall angepaßt werden, sind aber dennoch in das Gesamtsystem integrierbar, da sie

die Eigenschaften der Basiselemente geerbt haben. Denkbare Komponenten sind Fertigungspartialmodelle, Anforderungsmodelle, Geometriemodelle oder Stücklistenmodelle.
Bei der Schaffung einer komponentenbasierten CAx–Umgebung ist die Definition der Basiselemente, deren Gemeinsamkeiten und deren Wechselbeziehungen eine der herausragenden Aufgaben. Das Datenmodell von STEP kann hier als Grundlage dienen.
Eine weitere wichtige Aufgabe ist die Schaffung von Speichermechanismen für die einheitliche Speicherung der Partialmodelle, ohne die wirklich portable Objekte nicht denkbar sind. Alleine die Speicherung parametrischer Geometriemodelle in einem neutralen Format ist eine bisher nicht gelöste Aufgabe.
Voraussetzung für den Erfolg eines solchen Systems ist allerdings, daß die bestehtenden Systeme sich in eine solche Umgebung integrieren lassen und die allgemein definierten Objekte und Methoden in ihren Schnittstellen anbieten. Um bei einer gleichen Grundfunktionalität der Systeme dennoch die Abgrenzung untereinander zu ermöglichen, müssen allgemeine Objekte und Methoden definiert werden, die um die systemspezifischen Merkmale erweitert werden können.

Glossar

Applikation Ein Softwareprogramm zur Bearbeitung einer Aufgabe.

Client Eine Softwarekomponente, die die Dienste einer anderen Softwarekomponente in Anspruch nimmt, entweder durch einen Funktionsaufruf oder durch einen Zugriff auf ihren Status [Boo94].

Emitter Ein Softwareprogramm, das aus einer neutralen Schnittstellenbeschreibung Methodenrümpfe in einer bestimmten Programmiersprache generiert [OHE96].

Ersatzmodell Abstraktion eines realen oder geplanten Produkts durch Reduktion auf eine Auswahl von Eigenschaften. Ein Ersatzmodell wird verwendet, um das Verhalten des Produkts durch Vereinfachung berechenbar und damit vorhersagbar zu machen.

Framework (Auch: Umgebung) Eine Ansammlung von Klassen, die eine bestimmte Menge von Diensten für einen bestimmten Bereich zur Verfügung stellen. Ein Framework exportiert einzelne Klassen und Mechanismen, die die Clients einsetzen oder übernehmen können [Boo94].

Geometriemodell Hier: Rechnerinterne Darstellung der Geometriedaten in einem Geometriemodellier (CAD–System).

Integriertes Produktmodell Produktmodell für die Abbildung von Daten aus allen Produktlebensphasen in einem einheitlichen Datenmodell [GAP93]. Ein kontinuierliches Fortschreiben der Daten soll die informationsverlustbehaftete Modelltransformation verhindern. In dem Modell werden alle Daten abgebildet, die sich nicht aus bereits bestehenden Daten ableiten lassen [Vel99].

Methode Hier: ein auf einem Computer implementiertes Programm zur Durchführung von Berechnungen.

Persistenz Die Eigenschaft eines Objekts, mit deren Hilfe es Zeit überdauern kann [Boo94].

Modell Abstrakte Abbildung eines Ausschnittes der realen Welt mit dem Ziel, von Eigenschaften des Modells auf Eigenschaften des realen Objekts zu schließen [VDI93] [Rot82]. Im Kontext von Softwareprogrammen: als Datenmodell strukturelle Datenmenge zur rechnerinternen Repräsentation eines Weltausschnittes [Abe95].

Partialmodell Kennzeichnet eine definierte endliche Informationsmenge von Produktmerkmalen als Teil eines Produktmodells [Abe95] [Gra99].

Produktinformationsmodell Auch: *konzeptionelles Modell* Definiert die möglichen Bestandteile und denkbaren Relationen zwischen den Bestandteilen für eine Abbildung eines Modells im Rechner [Boo94].

Produktmodell Die Einheit von Produktinformationsmodell und aller zu einem Produkt gehörenden Daten [Abe95].

Rechenmodell Hier: Rechnerinterne Darstellung der Daten eines Berechnungsprogramms.

Server Eine Softwarekomponente, die anderen Softwarekomponenten Daten oder Dienste zur Verfügung stellt.

Strukturmodell Hier: Produktinformationsmodell für die Integration von Gestaltung und Berechnung. Enthält Ausschnitte aus einem Geometriemodell und einem Rechenmodell und die für die Verknüpfung der Modelle erforderlichen Informationen.

Literaturverzeichnis

[Abe95] Abeln, Olaf (Hrsg.): *CAD-Referenzmodell*. Stuttgart : B. G. Teubner, 1995

[Abe97] Abeln, Olaf (Hrsg.): *Innovationspotentiale in der Produktentwicklung*. Stuttgart : B.G. Teubner, 1997

[AGL98] Abramovici, M. ; Gerhard, D. ; Langenberg, L.: Unterstützung verteilter Entwicklungsprozesse durch EDM/PDM. **In:** *Prozeßketten für die virtuelle Produktentwicklung in verteilter Umgebung* (siehe [VDI98]), S. 69–86

[AJSK98] Arnold, F. ; Janocha, A. ; Swienczeck, B. ; Kilb, T.: Die CAx-Integrationsarchitektur ANICA und ihre erste Umsetzung in die Praxis. **In:** *Integration heterogener Softwaresysteme (IHS'98) im Rahmen der 28. GI-Jahrestagung Informatik '98*. Magdeburg, 1998, S. 43–54

[AK98] Arnold, F. ; Kilb, T.: Data Management in distributed CAx Systems. **In:** Anderl, R. (Hrsg.) ; Trippner, D. (Hrsg.): *ProSTEP Science Days '98, Product Data Technolgy - Facing the Future*. Wuppertal, 1998, S. 94–105

[And96] Andreasen, M.: The Nature of Design Features. **In:** Meerkamm, H. (Hrsg.): *7. Symposium Fertigungsgerechtes Konstruieren* Lehrstuhl für Konstruktionstechnik, Universität Erlangen-Nürnberg, 1996

[Bei90] Beitz, W.: Konstruktionsleitsystem als Integrationshilfe. **In:** *VDI–Berichte Nr. 812* (1990), S. 181–201

[Bet94] Betz, Mark: Interoperable Objects. **In:** *Dr. Dobbs Journal* (1994), October, S. 18–39

[BOL94] Institut für Maschinenkonstruktion – Konstruktionstechnik. Berlin: *BOLT 4.2 Benutzerhandbuch*. 1994. – TU Berlin

[Boo94] Booch, Grady: *Objektorientierte Analyse und Design*. Bonn, Paris : Addison-Wesley, 1994

[BW98] Braun, P. ; Welp, E.G.: Wissensbasierte Unterstützung der Produktentwicklung in objektorientiert gekapselten Ingenieuranwendungen. **In:** *Prozeßketten für die virtuelle Produktentwicklung in verteilter Umgebung* (siehe [VDI98]), S. 429–449

[Dam96] Damblon, Werner: *VDI calc Maschinenelemente*. Düsseldorf: VDI, 1996. – CD-Rom zur Auslegung und Berechnung

[Fel89] Feldhusen: Durchgängige und flexible Rechnerunterstützung in der Konstruktion. **In:** *Konstruktion* 41 (1989), S. 47–56

[FR99] Fischer, Th. ; Rautert, J.: Erfolgsfaktoren im Entwicklungsprozeß bei Heidelberger Druckmaschinen. **In:** *Verkürzte Entwicklungsprozesse durch Integration von Gestaltung und Berechnung* (siehe [VDI99]), S. 1–12

[Gal80] Galwelat, Michael ; Beitz, Wolfgang (Hrsg.): Rechnerunterstützte Gestaltung von Schraubenverbindungen. Berlin : TU Berlin, Institut für Maschinenkonstruktion-Konstruktionstechnik, 1980 (Nr.2). – Schriftenreihe Konstruktionstechnik

[GAP93] Grabowski, H. ; Anderl, R. ; Polly, A.: *Integriertes Produktmodell*. Berlin : Beuth Verlag, 1993

[GAS92] Grabowski, H. ; Anderl, R. ; Schmidt, M.: STEP: Die Beschreibung von Produktsrukturen mit dem Teilmodell PSCM. **In:** *VDI-Z* 134 (1992), Nr. 3, S. 51–55

[GEP94] Grabowski, H. ; Erb, J. ; Polly, A.: STEP - Grundlage der Produktdatentechnologie Aufbau und Entwicklungsmethodik. **In:** *CIM Management* 10 (1994), Nr. 4, S. 45–51

[GJS98] Gosling, J. ; Joy, B. ; Steele, G.: The Java Language Specification, 1998. – URL http://java.sun.com/docs/books/jls/html/index.html

[Gra99] Grabowski, Hans: CAD- und CAx-Software. **In:** *Konstruktion* Band 51 (1999), Oktober, Nr. 10

[Hah98] Hahn, Axel ; Gausemeier, Jürgen (Hrsg.): Integrationsumgebung für verteilte objektorientierte Ingenieursysteme. Paderborn : Universität-GH Paderborn, Heinz Nixdorf Institut, 1998 (Bd. 33). – HNI-Verlagsschriftenreihe

[Har97] Hardwick, Martin: STEP Business Objects ISFAA-97 Electronic Forum, 1997. – URL http://sage.wt.com.au/∼ausstep/aec-libraries/w1-reference/bizobj.html

[HE92] Hubka, V. ; Eder, E.: *Einführung in die Konstruktionswissenschaften*. Berlin : Springer, 1992

[HEX95] HEXAGON Industriesoftware GmbH. Kirchheim/Teck: *Software zur Berechnung von hochbeanspruchten Schraubenverbindungen nach VDI 2230*. 1995. – Hexagon SR1

[IBM96] IBM Corporation. Austin: *SOMObjects Developer's Toolkit: Programmers Guide*. Second Edition. 1996

[ISO94] ISO10303: Industrial Automation Systems and Integration - Product Data Representation and Exchange. Genf : International Organization for Standardization, 1994 (Part 1: Overview and Fundamental Principles). – ISO-Norm

[iVi99] iViP: Leitprojekt integrierte Virtuelle Produktentstehung. Berlin : Bundesministerium für Bildung und Forschung, 1999. – Prospekt

[iVi00] iViP, Projekt: Integrierte Virtuelle Produktentstehung, 2000. – URL http://www.ivip.de

[Küm99] Kümmlee, H.: Möglichkeiten und Grenzen des Einsatzes moderner EDV-Verfahren großer elektrischer Grenzleistungsmaschinen. **In:** *Verkürzte Entwicklungsprozesse durch Integration von Gestaltung und Berechnung* (siehe [VDI99]), S. 177–202

[KMM99] Kellner, P. ; Meissner, M. ; Müller, A.: Integration von CAx-Produktstrukturen - PDM-Systemen als Vorbereitung für virtuelle Produktentwicklung. **In:** *Verkürzte Entwicklungsprozesse durch Integration von Gestaltung und Berechnung* (siehe [VDI99]), S. 31–46

[Kol98] Koller, Rudolf: *Konstruktionslehre für den Maschinenbau*. 4. Auflage. Berlin, Heidelberg : Springer, 1998

[KOM96] KOMFORCE Arbeitskreis: Integrierter Entwicklungsarbeitsplatz. Paderborn : Kommunikations– und Forschungskreis für Informationstechnologien in Computer Aided Design und Engineering, 1996. – Referenzpapier

[KTA98] Krause, F.-L. ; Tang, T. ; Ahle, U.: Virtuelle Produktentstehung. **In:** *Prozeßketten für die virtuelle Produktentwicklung in verteilter Umgebung* (siehe [VDI98]), S. 1–12

[Löf97] Löffel, Christoph: *Integration von Berechnungswerkzeugen in den rechnerunterstützten Konstruktionsprozeß*. Erlangen, Lehrstuhl für Konstruktionstechnik. Universität Erlangen–Nürnberg, Dissertation, 1997

[LG99] Löffel, Chr. ; Goldbach, H.: Auf dem Weg zum virtuellen Produkt durch intgrierte Berechnungsmethoden. **In:** *Verkürzte Entwicklungsprozesse durch Integration von Gestaltung und Berechnung* (siehe [VDI99]), S. 123–138

[Lov97] Loviscach, Dr. J.: Paßgenau - Mit Komponenten zur persönlichen Software. **In:** *c't* (1997), Nr. Heft 3, S. 224–228

[McF98] McFall, Cynthia: An Object Infrastructure for Internet Middleware. **In:** *IEEE Internet Computing* (1998), März April, S. 46–51

[Mee98] Meerkamm, Harald: Integrierte Produktentwicklung. **In:** *Konstruktion* Band 50 (1998), September, Nr. 9

[Mer98] Mertens, H.: Aussagegüte und Zeitaufwand - Kriterien zur Auswahl von Berechnungsmethoden im Konstruktionsprozeß. **In:** *VDI Berichte Nr. 1442* VDI, Verein Deutscher Ingenieure, 1998

[MW99] Mertens, H. ; Wölfle, F.: Kooperation verteilter, dezentraler Konstruktions– und Berechnungsarbeitsplätze. **In:** Pahl, G. (Hrsg.): *Gedenkschrift Wolfgang Beitz*, Springer, 1999

[Nel99] Nell, Jim: STEP on a Page / National Institute of Standards and Technology. 1999. – ISO 10303 Editing Committee http://www.nist.gov/sc5/soap

[N.N99] N.N.: Berliner Kreis Kompetenz Netzwerk. **In:** *http://www.bkkn.de* (1999)

[OH97] Orfali, Robert ; Harkey, Dan: *Client/Server Programming with JAVA and CORBA*. New York : Wiley, 1997

[OHE96] Orfali, Robert ; Harkey, Dan ; Edwards, Jeri: *The Essential Distributed Objects Survival Guide*. New York : John Wiley & Sons, Inc., 1996

[OMG95] OMG: *Common Facilities Architecture*. 4. Revision. 1995

[OMG98] OMG: *PDM Enablers*. Revised Submission. 98. – mfg/98-01-01

[Ope91] Open Software Foundation (Hrsg.): *Guide to OSF/1*. Sebastopol : O'Reilly Associates, Inc., 1991

[Ope93] Open Software Foundation (Hrsg.): *OSF DCE Application Development Guide*. Revision 1.0. New Jersey : Prentice Hall, 1993

[Par97] Parametric Technolgy Corporation: *Pro/TOOLKIT User's Guide*. Release 18. 1997

[PB97] Pahl, G. ; Beitz, W.: *Konstruktionslehre*. 4. Auflage. Berlin : Springer, 1997

[Pot95] Potthast, A.: Datenaustausch zwischen STEP und ACIS. **In:** *CAD-CAM Report* (1995), Nr. 3

[Rüc97] Rückert, Carsten: Untersuchungen zur Konstruktionsmethodik – Ausbildung und Anwendung . Düsseldorf : VDI Verlag, 1997 (Reihe 1 Nr.293). – VDI Fortschr.-Ber.

[Red96] Redlich, Jens-Peter: *CORBA 2.0*. Bonn, Reading, Mass., Menlo Park, California : Addison-Wesley, 1996

[Rie94] Rieger, Erik: *Semantikorientierte Features zur kontinuierlichen Unterstützung der Produktgestaltung*. München : Hanser, 1994 (Produktionstechnik - Berlin 158)

[Rot82] Roth, K.: *Konstruieren mit Konstruktionskatalogen*. Berlin : Springer, 1982

[SAKJ98] Swienczek, B. ; Arnold, F. ; Kilb, Th. ; Janocha, A.: Online–Kopplung von CAx-Systemen für die virtuelle Produktentwicklung – ein Vergleich mit dem dateibasierten Datenaustauch. **In:** *VDI-Berichte Nr. 1435* VDI, Verein Deutscher Ingenieure, 1998, S. 219–238

[Sch88] Scholz, B.: *CIM-Schnittstellenkonzepte, Standards und Probleme der Verknüpfungen von Systemkomponenten der rechnerintergrierten Produktion*. München : Oldenbourg, 1988

[Sch91] Schwenke, Hendrick ; Beitz, Wolfgang (Hrsg.): Konstruktionswissen in CAD–Anwenderprozeduren. Berlin : TU Berlin, Institut für Maschinenkonstruktion-Konstruktionstechnik, 1991 (Nr.21). – Schriftenreihe Konstruktionstechnik

[Sei85] Seiler, W.: Technische Modellierungs- und Kommunikationsverfahren für das Konzipieren und Gestalten auf der Basis der Modellintegration. Düsseldorf : VDI-Verlag, 1985 (Reihe 10, Angewandte Informatik, Nr. 49). – VDI Forschungsbericht

[Stü90] Stürmer, Ulf ; Beitz, Wolfgang (Hrsg.): Informationsmodell zum Abbilden funktionaler und wirkstruktureller Zusammenhänge im Maschinenebau. Berlin : TU Berlin, Institut für Maschinenkonstruktion-Konstruktionstechnik, 1990 (Nr.17). – Schriftenreihe Konstruktionstechnik

[The99] Theissen, J.: Kopplung von Gestaltung und Berechnung während und nach der Entwicklungsphase einer Industrie Getriebebaureihe. **In:** *Verkürzte Entwicklungsprozesse durch Integration von Gestaltung und Berechnung* (siehe [VDI99]), S. 139–158

[Tur99] Turau, Volker: Techniken zur Realisierung Web-basierter Anwendungen. **In:** *Informatik-Spektrum* 22 (1999), Nr. 1, S. 3–12

[VDI80] VDI 2211: Datenverarbeitung in der Konstruktion / VDI Verein Deutscher Ingenieure. 1980. – VDI-Richtlinie Blatt 1

[VDI86] VDI 2230: Systematische Berechnung hochbeanspruchter Schraubenverbindungen / VDI Verein Deutscher Ingenieure. 1986. – VDI-Richtlinie

[VDI93] VDI 2221: Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte / VDI Verein Deutscher Ingenieure. 1993. – VDI-Richtlinie

[VDI98] VDI-Gesellschaft Entwicklung, Konstruktion, Vertrieb (Veranst.): *Prozeßketten für die virtuelle Produktentwicklung in verteilter Umgebung (VDI–Berichte 1435)*. Düsseldorf : VDI-Verlag, 1998

[VDI99] VDI-Gesellschaft Entwicklung, Konstruktion, Vertrieb (Veranst.): *Verkürzte Entwicklungsprozesse durch Integration von Gestaltung und Berechnung (VDI-Berichte 1487)*. Düsseldorf : VDI Verlag, 1999

[Vel99] Velten, V.: *Integration von Strömungsberechnungen in den rechnerunterstützten methodischen Konstruktionsprozeß*. Aachen : Shaker, 1999

[VRM99] Virtual Reality Markup Language Web 3D Consortium, 1999. – URL http://www.vrml.org

[Wah86] *Deutsches Wörterbuch*. Gütersloh, München, 1986

[WB97] Welp, E.G. ; Braun, P.: Semantische und systemtechnische Kopplung kommerzieller CAD– und Berechnungssysteme durch einen objektorientierten Integrationsprozessor. **In:** VDI-Verlag (Hrsg.): *VDI–Berichte 1357*. Düsseldorf, 1997, S. 125–141

[Web92] Weber, A.: *Ein relationsbasiertes Datenmodell als Grundlage für die Bauteiltolerierung*. Erlangen, Lehrstuhl für Konstruktionstechnik. Universität Erlangen–Nürnberg, Dissertation, 1992

[Wöl98] Wölfle, Frank: Virtuelle Berechnungskompetenzzentren als Dienstleister zur Integration von Gestaltung und Berechnung. Düsseldorf : VDI Verlag, 1998 (Reihe 20 Nr.274). – VDI Fortschr.-Ber.

[WN99] Westfechtel, B. (Hrsg.) ; Nagl, M. (Hrsg.): *Integration von Entwicklungssystemen in Ingenieuranwendungen*. Berlin, Heidelberg : Springer, 1999

[Zie99] Ziesing, J.: Technische Berechnungen als Dienstleistung für die Konstruktionspraxis. **In:** *Verkürzte Entwicklungsprozesse durch Integration von Gestaltung und Berechnung* (siehe [VDI99]), S. 103–122