Structure-Oriented Exchange of Product Model Data

vorgelegt von
Diplom-Ingenieur
Richard A. Baumann
aus Barnstaple / North Devon, Großbritannien

von der Fakultät V – Verkehrs- und Maschinensysteme
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
– Dr.-Ing. –

genehmigte Dissertation

Promotionsausschuss:

Vorsitzende:	Frau Prof. Dr.-Ing. L. Blessing	
Berichter:	Herr Prof. Dr.-Ing. F.-L. Krause	
Berichter:	Herr Prof. Dr.-Ing. Chr. Weber	(Univ. Saarland)

Tag der wissenschaftlichen Aussprach: 14. September 2004

Berlin 2004
D 83

Vorwort des Herausgebers

Der Austausch von Produktmodelldaten ist seit vielen Jahren Gegenstand internationalen Engagements durch Wissenschaft und Industrie. In dem Bestreben, aktuelle Systementwicklungen zu berücksichtigen, schreitet die Entwicklung und Standardisierung von Datenmodellen und Austauschformaten fort.

Speziell der Austausch von CAD-Modellen konzentriert sich dabei im Wesentlichen auf die Abbildung des Konstruktionsergebnisses in Form der Produktgestalt. Standard-Austauschformate, wie IGES oder nach VDA oder STEP, haben sich etabliert. Sie verhindern jedoch die weitere Bearbeitung der ausgetauschten Modelle mit modernen Mitteln des Feature-basierten Modellierens.

Die vorliegende Arbeit stellt eine neue Methode des Austausches Feature-basierter CAD-Daten vor. Ausgangspunkt ist die Überlegung, dass sowohl für die Definition als auch für die Manipulation eines Teilemodells strukturelle Mechanismen des Feature-Modells ausschlaggebend sind, und dass deshalb eine systemneutrale Modellrepräsentation primär strukturorientiert organisiert sein muss. Der Autor entwickelt hierfür einen Ansatz der impliziten Repräsentation, die auf die Abbildung umfangreicher Gestaltdaten verzichtet. Teil des Konzeptes ist auch eine Strategie für die Identifikation von Gestaltelementen aus dem Feature-Modell heraus, das der Fachwelt gemeinhin als „persistent naming problem“ bekannt ist. Die Methode des strukturorientierten Austausches adressiert auch eine XML-basierte Strategie für den Transfer von Produktmodelldaten und zeigt Mechanismen für eine fehlertolerante und anwenderorientierte Unterstützung für integrierte Prozessketten auf.

Der Autor beschreibt weiterhin die Implementierungsaktivitäten auch für kommerzielle CAD-Systeme. Stellungnahmen der Industrie und eine Kosten-Nutzen-Analyse runden die Arbeit ab.

Insgesamt zeigt die Arbeit einen neuen Weg für den Austausch Feature-basierter Produktmodelldaten auf, der sich hier auf CAD-Umfeld konzentriert. Die vorgestellte Methode lässt jedoch das Potenzial erkennen, auch für die Integration anderer CA-Systeme einen Beitrag leisten zu können. Mit dem strukturorientierten Austausch von Produktmodelldaten steht somit ein neuer und wichtiger methodischer Baustein für den Aufbau durchgängiger Prozessketten in der Produktentstehung zur Verfügung.

Berlin, September 2004 Frank-Lothar Krause

Preface of the Author

The dissertation at hand emerged from my occupation at the Institute for Machine Tools and Factory Management (IWF) of the Technical University of Berlin, and at the Fraunhofer-Institute for Production Systems and Design Technology (IPK) in Berlin, Germany.

To Professor Dr.-Ing. Frank-Lothar Krause, head of the Industrial Information Technology division at IWF and head of Virtual Product Creation division at IPK, I express my gratitude for the opportunity to broaden my horizon and for his friendly aid and guidance of my work. Professor Dr.-Ing. Christian Weber, chair of the Institute of Engineering Design/CAD at the Saarland University, I thank for the analysis of this dissertation and for rendering an expert opinion. Mrs. Prof. Dr.-Ing. Lucienne Blessing I thank for her interest in my work and for chairing the promotion committee.

Many thanks also to my colleagues at IWF and IPK for the close collaboration and expertise. Especially, I am indebted to Dip.-Ing. Matthias Seiferth, Dipl.-Ing. Mingang Wang, cand. Ing. Masoud Gholchin for their excellent assistance during the implementation phases, as well as to many colleagues and friends for fruitful discussions and immediate reviews of the manuscript.

My special thanks go to my family who enabled and encouraged me to follow a scientific education and to Ms. Diana Beiner whose energy and endorsement was of fundamental assistance.

Acknowledgement:
The findings presented in this dissertation partly emerged from the research project KR 785/11-1, 2 funded by the Deutsche Forschungsgemeinschaft (DFG) in co-operation with the Institute of Production Engineering and Machine Tools (IFW) of the University of Hannover, Germany. Sincere thanks for the financial support and the fruitful co-operation.

Berlin, July 2004 Richard Baumann

Strukturorientierter Austausch von Produktmodelldaten (German Summary)

Produktentwickelnde Unternehmen reagieren auf steigende Marktanforderungen mit Strategien zur Effizienz- und Effektivitätssteigerung für Neugestaltung ihrer Produktentstehungsprozesse. Der Produktdatenaustausch als nicht-produktive und zeitraubende Aufgabe steht dabei im besonderen Fokus der Optimierung. Die Entwicklung von Produkten erfordert einen effektiveren Datenaustausch entlang des Produktentstehungsprozesses und eine flexiblere Unterstützung verschiedener Austauschsituationen, die über den bloßen Dateitransfer hinausgehen.

Der Modellaustausch zwischen CAD-Systemen ist hier von zentralem Interesse. Es werden Austauschmechanismen benötigt, welche die Produktgestalt unverändert über Systemgrenzen hinweg zu transferieren vermögen. Darüber hinaus sollten die ausgetauschten Modelle änderbar sein, das heißt, sie sollen sich im importierenden System wie native Modelle verhalten. Derartige Austauschmechanismen sollten standardisiert sein, um Kosten für die Schnittstellenentwicklung zu minimieren und Investitionen zu sichern.

Die Betrachtung des Standes der Technik zu Repräsentation und Austausch von Produktmodelldaten führt zu der Erkenntnis, dass weder etablierte Standardaustauschformate, hier ist primär die STEP-Technologie zu nennen, noch bekannte Forschungsaktivitäten adäquate Lösungen für diese Erfordernisse bereitstellen. Aus dieser Divergenz zwischen industriellen Anforderungen und verfügbaren Technologien heraus wird das Ziel der vorliegenden Arbeit definiert: Es soll eine neue Methode für den Austausch von Produktmodelldaten entwickelt werden. Die Methode soll eine systemneutrale Repräsentation sowie eine Transferstrategie für Produktmodelle beinhalten. Dabei wird der Anwendungsbereich auf den Austausch Feature-basierter Modelldaten zwischen CAD-Systemen eingegrenzt.

Eine Diskussion wesentlicher Anforderungen führt zur Definition der Strukturorientierung als struktur- und gestalterhaltende Charakteristik der systemneutralen Modellrepräsentation. Es werden zwei methodische Ansätze zu solch einer strukturorientierten Repräsentation identifiziert. Der explizite Ansatz verbindet strukturelle Informationen mit einer Hierarchie von Gestaltmodellen. Der implizite Ansatz macht sich den Umstand zunutze, dass die Modellstruktur alle für die Generierung und Modifikation eines CAD-Modells essentiellen Informationen enthält. Der Datenaustausch reduziert sich somit auf die Modellstruktur; auf den Austausch von Gestaltdaten kann, bis auf bestimmte Ausnahmen, verzichtet werden.

In Hinblick auf technische Anforderungen ist der implizite Ansatz vorzuziehen. Hierfür wird eine systemneutrale Repräsentation für Feature-basierte Modelle entwickelt. Als standardisierte Basis für die Austauschmethode wird eine generalisierte Feature-

Bibliothek empfohlen. Für Modellstruktur, benutzerdefinierte Features, Parametrik- und Constraintmodelle, Baugruppenstrukturen und andere Modellinhalte werden detaillierte Repräsentationsmechanismen erarbeitet. Eine erzeugungsorientierte Taxonomie für Freiform-Features wird vorgeschlagen.

Ein weithin diskutiertes Problem im Zusammenhang mit Feature-basierter Produktmodellierung ist die Identifikation von Gestaltelementen – bekannt als das *persistent naming problem* – das einen Gegenstand aktueller Forschungen darstellt. Es wird ein Mechanismus zur intelligenten Koordinatenreferenzierung vorgeschlagen. Entgegen der generellen Literaturmeinung erübrigt sich somit die Applikation eindeutiger Bezeichner (identifier) für die explizite Referenzierung von Gestaltelementen.

Als Grundlage für eine Transferstrategie für strukturorientierte CAD-Modelldaten werden drei Austauschszenarien definiert, die – in Erweiterung des klassischen Dateitransfers – Austauschsituationen in Web-basierten und systemintegrierenden Umgebungen beschreiben. Auf Basis dieser Szenarien werden zentrale Funktionalitäten und Komponenten für den Modelltransfer identifiziert. Als optimale Technologie für die Definition des physischen Austauschformates wird XML ausgewählt und ein entsprechendes Funktionskonzept für Modellanalyse, Transkription und CAD-Interaktion entwickelt. Als Voraussetzung für eine vollständige und effiziente Realisierung der Konzepte wird eine implizite CAD-API vorgeschlagen, deren Charakteristik und Funktionalität beschrieben werden. Die Integration dieser Funktionalitäten in eine zukünftige Version der OMG CAD-Services Spezifikation wird empfohlen. Zu diesem Zweck wird eine entsprechende Spezifikation präsentiert.

Die Realisierung der vorgestellten Konzepte verlief in zwei Phasen. Eine Umsetzung grundlegender Mechanismen war Gegenstand eines Forschungsprojektes, in dessen Verlauf Prozessoren für die universitären Feature-Modelliersysteme FEAMOS und EMOS implementiert wurden. Eine formale Feature-Definitions- und Austauschsprache (Feature Definition and Exchange Language – FEADEL) stellt Konstrukte für die Repräsentation von Feature-Typen und Feature-basierten Modellen auf Basis der Gestaltmodellier-Bibliothek ACIS bereit.

Die zweite Phase adressierte eine konzeptionelle Detaillierung der Austauschmethode und ihre Anpassung an kommerzielle CAD-Systeme und industrielle Austauschsituationen. Die Realisierung fand für die Systeme Unigraphics und I-Deas statt. Die Austauschprozessoren wurden sowohl in eine Web-basierte als auch in eine systemintegrierende Infrastruktur implantiert. Als Benutzungsschnittstelle für erweiterte Austauschfunktionalitäten wird ein interaktiver Austausch-Client vorgestellt, der einen selektiven Modellaustausch und eine integrierte Konsistenzanalyse bereitstellt. Für eine Bewertung des Austauschverhaltens wurden industrielle Modelle herangezogen, die auch für die ProSTEP AP214 Prozessoren-Benchmarks Verwendung finden. Die erzielten Ergebnisse können im Hinblick auf die zuvor definierten Benutzeranforderungen als erfolgreich bewertet werden.

Die Implementierung für kommerzielle CAD-Systeme und die Ergebnisbewertung legen dar, dass es sich bei dem impliziten Ansatz für einen strukturorientierten CAD-

Datenaustausch um eine neue und qualifizierte Lösung handelt, die den Anforderungen an eine zeitgemäße Technologie gerecht wird. Stellungnahmen bekannter Unternehmen unterstreichen die Anwendbarkeit im industriellen Umfeld.

Structure-Oriented Exchange of Product Model Data

> I had a real problem and needed a real solution and I used the tools, environment, and know-how then available to me in a new and unique way to provide that real solution. Isn't that what "design" really *is* – whether computer-aided or not? Isn't that what engineering is all about?
> *Douglas T. Ross* [1]

Table of Contents

Abbreviations

Abbreviation	Term
ASP	Application Service Providing
BOM	Bill of Materials
B-Rep	Boundary Representation
CACD	Computer-Aided Conceptual Design
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CAM	Computer-Aided Manufacturing
CC	Conformance Class – a set of capabilities within a defined scope of application, specifically used to satisfy a set of requirements by a certain functionality of a data exchange standard
CORBA	Common Object Request Broker Architecture
CSG	Constructive Solid Geometry
DMU	Digital Mock-Up
DOM	Document Object Model – run-time model of an XML schema
DTD	Document Type Definition
ECAD	Electric CAD – CAD system with specialised functionality for electric or electronic components and structures
EXPRESS	Formal information requirements specification language; part of the STEP methodology
EXPRESS-G	Graphical representation of EXPRESS
FMEA	Failure Mode and Effect Analysis
HTML	HyperText Markup Language
IGES	Initial Graphics Exchange Specification
IT	Information Technology

Abbreviation	Term
J2EE	Java 2 Platform, Enterprise Edition – defines the standard for developing component-based applications based on, e.g. Web-services
MCAD	Mechanical CAD – CAD system with specialised functionality for mechanical parts
MDA	Model Driven Architecture
NURBS	Non-Uniform Rational B-Splines
OEM	Original Equipment Manufacturer
OMG	Object Management Group
PDM	Product Data Management
PDTnet	Product Data Technology and Communication in an OEM and Supplier Network – a joint project funded by the German Ministry of Economics and Labour (Germany)
SDAI	Standard Data Access Interface – part of the STEP methodology
SET	Standard d´Echange et de Transfert (France)
SOAP	Simple Object Access Protocol
UDF	User-Defined Feature
VDA	Verband der Automobilindustrie (Germany)
VDA-FS	VDA Flächenschnittstelle (Germany)
XSLT	Extensible Stylesheet Language Transfromations

Trademarks

Product	Ownership
ACIS	Spacial Corporation, Westminster, Colorado, USA (Member of Dassault Systèmes Group)
BISON	The general-purpose parser generator is under the GNU public license
CADDS5	Parametric Technology Corporation, Needham, MA, USA
CADdy ++ Maschinenbau	DataSolid GmbH, Mänchengladbach, Germany
CATIA	Dassault Systèmes, France
FLEX	The fast lexical analiser generator is under the GNU public license
FOD	CADsys GmbH, Chemnitz, Germany
I-Deas	UGS PLM Solutions, Plano, Texas, USA (formaly SDRC)
Inventor	Autodesk GmbH, München, Germany
Orbacus	IONA Technology, Dublin, Ireland
Parasolid	UGS PLM Solutions, Plano, Texas, USA
Pro/Engineer	Parametric Technology Corporation, Needham, MA, USA
SolidWorks	Solid Line AG, Walluf, Germany (Member of Dassault Système Group)
Unigraphics	UGS PLM Solutions, Plano, Texas, USA

List of Figures

List of Tables

1 Introduction

Globalisation has noticeable effects on export-oriented economies as it generates high market pressure on producing companies. Not only in Germany enterprises have to realise a near economic recession situation and a consolidation of established markets. Aiming at a constantly increasing productivity, engineering industries apply concepts like profit centres and “make or buy” strategies that have led to rather diversified supply chains. In German automotive industry, OEMs have repositioned themselves to system integrators, whereas nearly 80 % of the value chain is obtained by suppliers [2].

As illustrated in Figure 1-1, producing companies have developed strategies to raise efficiency of production processes in general and of product creation processes in particular:

- Reduction of development time;
- Increasing collaboration; and
- Efficient IT strategies ([3], [4]).

Market demands

Increasing pressure on engineering companies / departments

Enterprise reaction

1. Reduction of development time
2. Increasing cooperation
3. Consolidation of IT strategies

Engineering consequence

Highly efficient exchange of product model data for better process chains

Baumann 2004

Figure 1-1 Situation of engineering industries and product development strategies

Reduction of development time is realised by concepts like platforming, extreme frontloading, parallelisation of product development and production planning. Essential for a comprehensive development process in platform engineering is the design of a so-called product architecture that allows the integration of different product models into an overall model [5]. Special effort is raised for synchronisation of corresponding processes and for harmonisation of information flows: Early product concepts containing requirements, functionality, and shape information form the basis for design, validation and planning processes and are updated constantly. Data from former development projects is re-used as early conceptual input until the new product has reached a certain stage.

Increasing engineering collaboration within virtual enterprises has lead to a complex net of interrelated partners and processes [6]. Mechanical, electric, hydraulic, and other components are developed simultaneously and in close information cycles. Product evaluation methods like FMEA are applied as a means for quality assurance among the partners. The number of development tools for design, calculation, simulation, or NC planning, increases with the number of engineering partners involved.

Consequently, business strategies are also concerned with the efficient and effective application of IT systems and infrastructures. In the beginning of computer aided product development and planning, various different systems were introduced which in many cases has led to a complex environment of isolated system islands. Today's efforts aim at integrating these islands into a cross-company infrastructure to which also cooperating partners are granted access. Mechanisms like application service providing and Web-based front-ends and online-interfaces to product development databases and PDM systems are realised [4], [7]. On the systems side, companies have tried to reduce the number of software tools anticipating that a one-system-strategy would solve the problem of data incompatibility hindering the inter-process information flows [8]. Nevertheless, a consequent one-system or even one-provider strategy has not been realised today due to the enormous number of systems applied for various purposes. Furthermore, considering the virtual enterprise as a whole, a one-system strategy is little effective. OEMs force their first-tier suppliers to provide native design data, which increases the number of systems applied by the suppliers.

The strategies described attain greatest effects on the information flows within and among the involved partners' product creation processes (Figure 1-2): Information exchange occurs earlier, more frequently, and often implying intermediate instead of final results. Especially, CAD concept and shape data is frequently transferred. Due to the large number of systems, a corresponding variety of data formats demands powerful data exchange mechanisms. Especially OEMs in automotive and aerospace industries depend on the ability to migrate product data from their suppliers to an overall product design. For example, the application of feature-based modelling increases with the availability of feature-based CAD systems. With increasing feature-based designs grows the demand for exchange formats capable of handling corresponding data without loss of information.

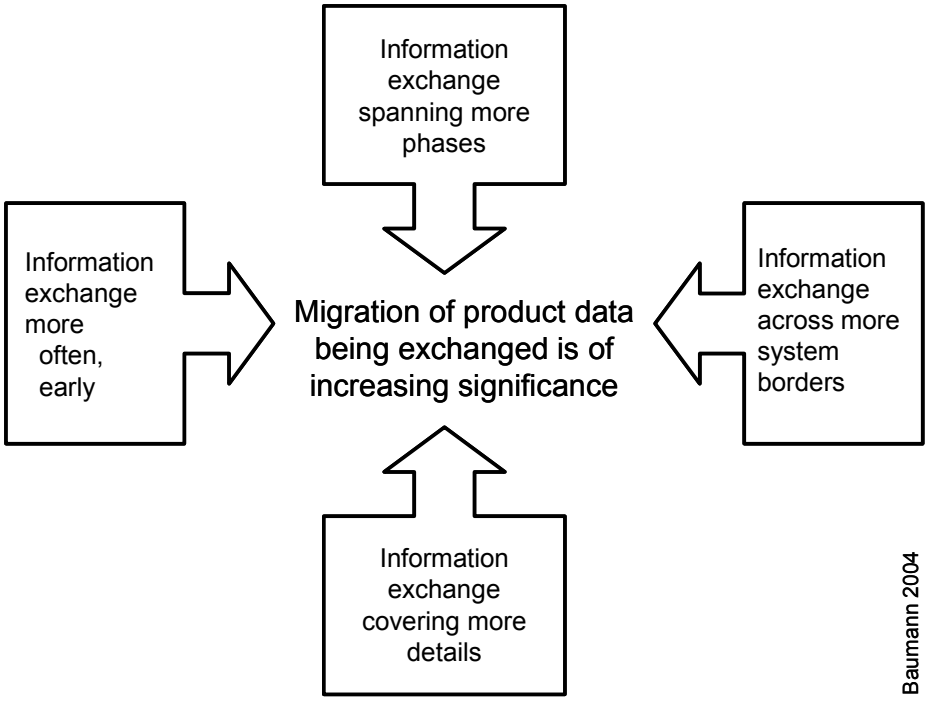


Figure 1-2 Effects on engineering information flow

The example shows that an effective information flow and the ability to migrate various product data formats in a way that provides the necessary data for every development step is a key factor for efficient product creation. Migration in this context implies the possibility to integrate all relevant engineering data into a compound product model in a reusable form. This demands for exchanging more information in quality and quantity and for enhancing the information flow across product creation phases.

Unfortunately, compared to the aggregation of product information and the corresponding diversification of corresponding product data, mechanisms for an effective engineering data exchange – be it native or based on standard formats – are rather underdeveloped (Figure 1-3). In this context, CAD model data exchange is the critical point because beyond all others, CAD models represent the core result of product development – the product's shape.

When examining the field of CAD model data exchange in detail, it can be noted that single-system strategies are not only followed to meet the demand for aggregation. Another reason lies within the fact that later modification of CAD models provided by third parties is only possible when using the same CAD system. Standardised exchange formats for CAD model data do not provide the necessary data integrity, whereas the development of powerful native interfaces for all involved systems appears to be too costly.

For CAD model exchange the transfer of exact product shape and dimensions is the most important requirement. Subsequent product development phases strongly depend on the availability of correct shape information. Nevertheless, for the integration of product creation processes the exchange of pure shape information is insufficient. Design results represented by CAD models contain further information, such as parametric and constraint information for product functionality, calculation, assemblies and variant parts, and other semantic content like tolerances, material and surface quality data. Two examples may be outlined:

1. In CAM operation planning, a complex and failure-prone feature recognition facility is needed if the original feature information was lost during data exchange between CAD and CAM system.

2. In mechatronic systems development, mechanical and electronic components are designed using different systems. The inability to exchange the complete mechanical model to the ECAD system results in a complicated change processes whenever a modification of an electronic component requires changes to the mechanical parts (and vice versa).

The examples illustrate the demand for an exchange mechanism capable of representing all relevant product information and of providing means for later alterability of the imported data model within the receiving system.

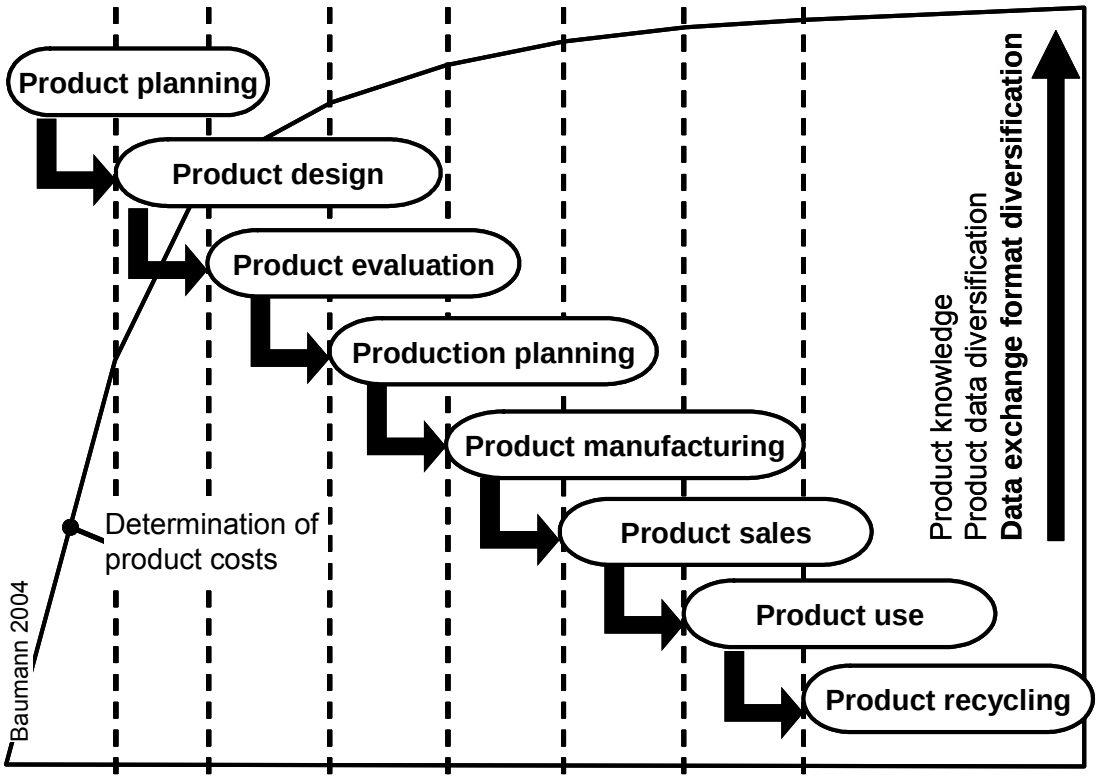


Figure 1-3 Aggregating product information along the product life cycle – ideal development of product knowledge and diversification of product data and data exchange formats

Regarding business strategies of engineering industries, the demand for adequate data exchange mechanisms can be summarised:

- The application of product data exchange mechanisms increases. The demand evolves towards representation of more and high-level product information.
- Data exchange methods based on standardised representation and transfer can be assumed as the economically most reasonable approach. Available standard formats, however, do not provide adequate solutions.
- Further effort must be made to overcome engineering system borders by enabling a fully alterable “behaviour” of exchanged data models within the receiving system.
- CAD data exchange can be identified as a central concern within the information flow of product creation.
- Data exchange strategies must be compliant with modern collaboration and IT infrastructure concepts.

This dissertation describes the development of a new strategy to the exchange of CAD model data. It proposes a so-called structure-oriented method as a solution to the demand for a more powerful data exchange mechanism as illustrated above. The scientific procedure will be as follows:

Chapter 2: Summary of the state of the art in product model data exchange. Model representation and model transfer methods and known approaches are outlined separately. Complementing findings are described as an input to conceptual discussion.

Chapter 3: Discussion of shortcomings of the state of the art and derivation of precise objectives and scope of application for the development of a new exchange strategy; definition of requirements from an application specific perspective; postulation of a thesis statement identifying a structure-oriented data exchange as a solution for the new exchange strategy.

Chapter 4: Definition and conceptual development of a structure-oriented representation method by introducing and discussing an explicit and an implicit approach of which one is selected according to technical requirements; specification of various detailed aspects of standardised model representation; definition of representative exchange scenarios; development of a model transfer method; synthesis of both representation and transfer method.

Chapter 5: Two phases of concept realisation for the structure-oriented data exchange as a proof of concept: implementation of representation principles using university feature modelling systems, and adaptation to commercial CAD systems and industrial exchange infrastructures; description of applied use-cases; proving and evaluation of concepts; cost-benefit analysis.

Chapter 6: Summary of scientific findings and outlook to consecutive research work.

2 State of the Art in Product Model Data Exchange

2.1 Model Representation Technologies

2.1.1 System Architecture and Internal Model Representation

In this chapter, the state of the art in product model data exchange is summarised. Primarily, for later discussion in a conceptual context four main disciplines have to be distinguished:

1. The internal architecture of CAD systems and corresponding model content;
2. Technologies for product model representation with the aid of system neutral formats;
3. Transfer strategies for product model data for the exchange between CAD systems; and
4. Complementing findings that deal with side effects and less related technologies.

This section will illustrate technologies for system internal representation of product models.

General CAD System Architecture and Model Representation

General CAD system architectures and strategies for internal model representation are described, for instance by SPUR / KRAUSE in [9], [10]. By systematising these general disquisitions and including CAD system vendors' product descriptions [11], [12], [13], [14] it is possible to depict an abstract CAD system architecture (Figure 2-1): A part modeller usually comprises a feature modelling engine which is separate from but depends on the shape modeller, often referred to as a geometric core modeller. An assembly modeller usually is a separate application.

A system internal model architecture represents and handles the corresponding shape model, feature model and additional data in separate structures. The overall part model strongly interrelates these models by means of object referencing (bi-directional associations) and appropriate algorithms for parametric values, constraints and further more. Following, the characteristics of modelling and, specifically, representation of shape, features, parametric, free-formed shape and assembly information will be discussed in further detail.

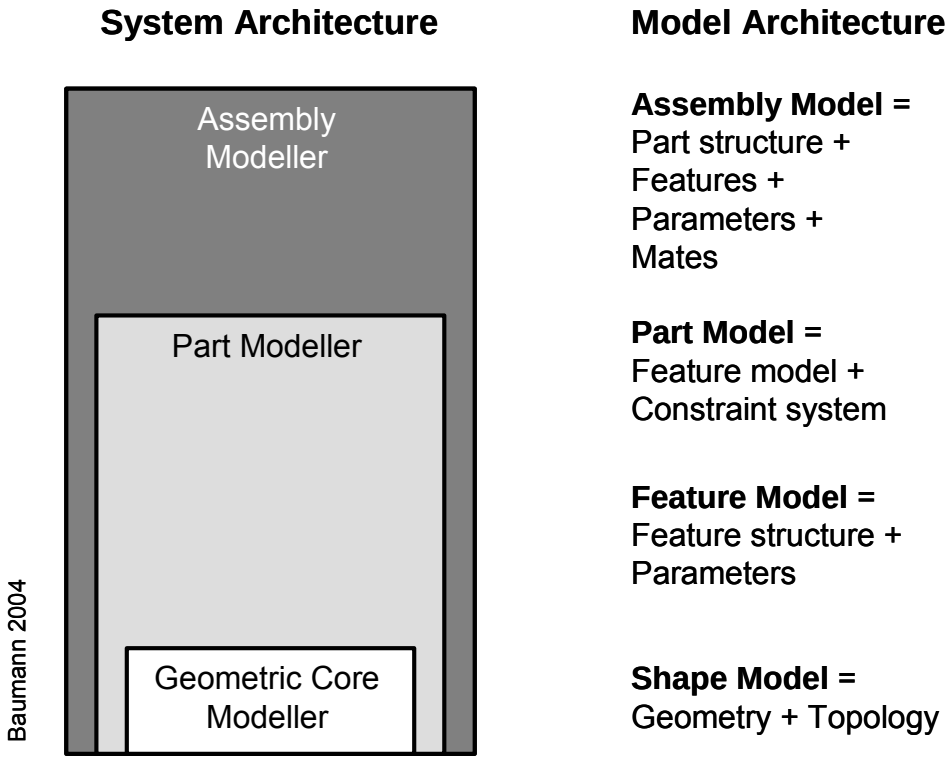


Figure 2-1 Generalised CAD system model architecture

Shape Modelling and Representation

A preliminary remark may be made to the term *shape*: In this dissertation *shape* is considered to consist of inter-dependent structures of geometry and topology entities. Literature often does not distinguish between geometry, topology and shape in this hierarchy. Instead, geometry is referred to in synonym to shape. In order to avoid misunderstandings the author will strictly abide to the term shape in the above sense.

Commercial shape modelling systems or libraries are, for instance, ACIS [15], Designbase [16] or Parasolid [17]. An overview on shape modelling and representation principles is given by SPUR / KRAUSE [10], classifying shape models into wireframe, surface and solid models. Solid modelling systems today are usually hybrid in the sense of being capable of handling surfaces and solids in one shape model. As shown in Figure 2-2, WEILER [18] further classifies solid models into

- Explicit or generative,
- Object-oriented or space-oriented,
- Volume- or boundary-oriented.

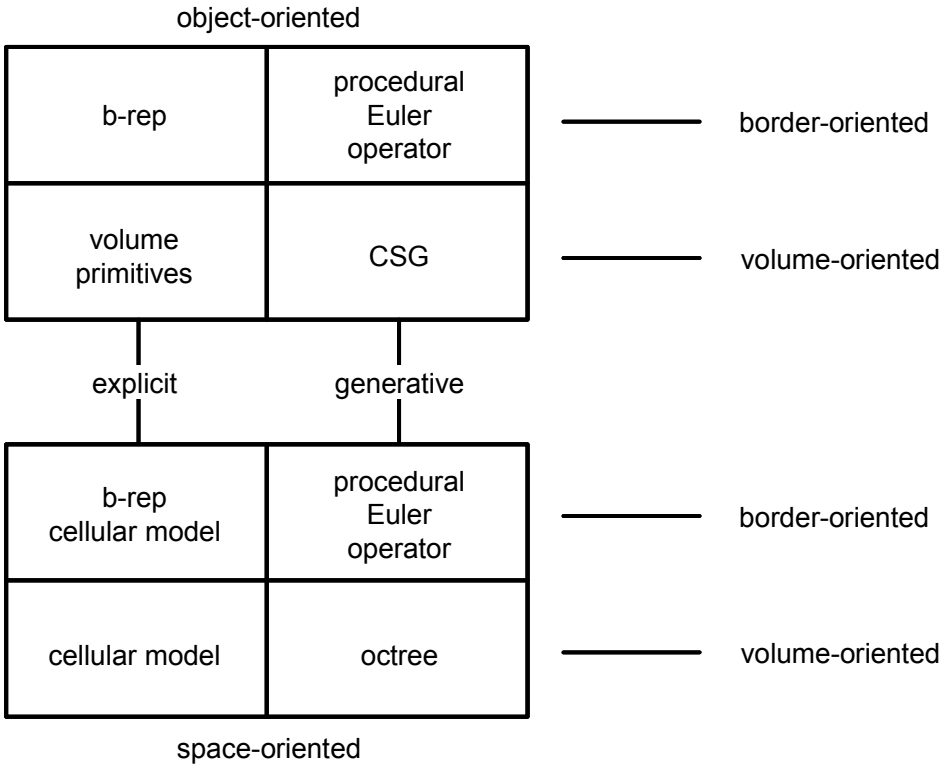


Figure 2-2 Classification of solid models [18]

The established representation of today's solid shape modelling systems is *boundary representation* (b-rep) [19], [20], [21], [22]. A b-rep structure defines spaces of n^{th} order by boundaries of $n\text{-}1^{st}$ order, i.e. solids by faces, faces by edges, edges by vertices. These topology objects (body, shell, face, loop, edge, vertex, etc.) are defined with reference to geometry objects (volume, surface, curve, point, etc.): a topological vertex resides at a geometric point in space, a straight edge on a line, a planar face on a plane, and so forth. This structure of topology and geometry objects is common to commercial shape modellers and standardised exchange formats, such as STEP [23].

The importance of CSG operations on b-rep shape structures as a basis for feature-based modelling was already stressed by, e.g. PRATT [24]. Different to boundary representation, shape information in *constructive solid geometry* (CSG) is represented as a sequence of Boolean operations onto solids. The resulting structure is a so-called CSG tree. Boolean operations may be union, difference or intersection. Today's shape modellers are based on the b-rep philosophy and additionally support CSG functions. This hybrid behaviour is the foundation for modern feature modelling capabilities. The interested reader will find a detailed explanation of b-rep and CSG representation philosophies and of corresponding data structures and modelling algorithms in, e.g. [25] and [26].

There exists a variety of further representation methods for product shape, such as facetted surfaces, octrees, voxel or other structure [27], [28]. Those are usually applied in DMU and other non-constructive tools rather than in CAD systems [29]. For that reason they will not be discussed here.

Feature Modelling and Representation

As early as 1991, the great variety of feature concepts and modelling approaches in literature and in software was classified by Shah [30]. Traditionally two main categories are distinguished for creating feature-based product models, known as *feature recognition* and *design by features*. In the feature recognition approach, a pure shape model is created first, then features are created by either a software algorithm that processes the resulting model to automatically find features, or by a human user interactively picking entities in a shape presenting image of the part [31], [32], [33], [10].

In today's general understanding, features are the main modelling objects of modern CAD systems. Feature-based CAD systems support the design by feature approach by providing the user with a set of feature types that are instantiated and interrelated in order to define shape and other characteristics of the product. The term *feature* itself was subject to harmonisation by the so-called Feature Modelling Experts (FEMEX) group. FEMEX was an international and interdisciplinary researchers and developers association from universities and industry, which has generated a unified view upon feature-based product development. Main results were a unified feature definition as well as a classification of fields of application, modelling methods and tools [34], [35], [36]. These results were summarised in the guideline number 2218 released by the Society of German Engineers (Verein Deutscher Ingenieure – VDI) [37], [38].

The lexical feature definition traces back to approaches from SHAH, SALOMONS to RIEGER [39], [40], [41]. According to the standardised definition, a feature is considered to contain semantic and (optional) shape information:

$$\text{feature} := \text{semantic} \vee \text{form-feature}$$

RIEGER also describes essential feature characteristics:

- The possibility for the user to define new features;
- The ability of the feature to represent semantic, not necessarily shape-related information;
- The capability to be identifiable throughout the shape model; and
- The representation of dynamic content.

Furthermore, RIEGER classifies features as being:

- Shape generating, i.e. feature instantiation initially generates shape elements;

- Shape manipulating, i.e. feature instantiation manipulates existing shape elements;
- Shape neutral, i.e. feature instantiation refers to existing shape elements without altering the shape model; or
- Shape independent, i.e. feature does not refer to any shape element.

Although feature objects are individually defined within every CAD system, regarding orientation in space, parameterisation, etc., the above characteristics can be considered as common to feature objects of practically all modern feature-based CAD systems.

Regarding their semantic and shape content, features in modern CAD systems can be considered as being complex design objects, characterised by a set of attributes or properties that determine the resulting behaviour of the feature within the CAD model and, specifically, the shape model. The semantic content varies with the scope of application of features and system. Features for design, assembly, manufacturing or quality assurance contain different information specific to their application [10].

An instantiated feature object may – to some extent – be altered by the user: changes to parameter values, suppression or deletion, repositioning within a structural tree (if admitted by the system's modelling logic). These alterations cause minor or major changes to the corresponding shape model.

The concept of user-defined feature (UDF) enables the user to define her/his own design objects according to specific requirements, application or design context. Feature-based CAD systems provide functionality for defining such re-useable design elements. Usually, a parameterised part model can be saved as a UDF in a local management facility that also handles those feature types pre-defined by the CAD system.

Parametric and Constraint-Based Modelling and Representation

Today's CAD systems provide parametric and constraint-based design both in shape and feature modelling. The user may assign values to dimensional and feature variables (synonymously spoken of as parameters) or may define constraints, i.e. mathematical formulas that do so. The so defined constraint system is calculated by a constraint solving engine. From the user's point of view, so-called geometric constraints, i.e. shape elements being parallel, intersecting, co-planar, rectangular, etc., and constraints on parameters owned by feature, part or assembly objects, have to be distinguished. Additionally, CAD systems provide means for defining shape or feature independent variables.

From the system's internal management perspective, two constraint satisfaction algorithms have to be distinguished, as classified by SHAH [42]:

(Fully) Parametric: Parametric systems solve constraints by sequentially applying assignments to model variables, where each assigned value is computed as a function of the previously assigned values. Unlike procedural systems, the order of the assignments is flexible, determined by a constraint propagation algorithm.

Variational: Variational systems solve constraints by constructing a system of equations representing the constraints, and solving all constraints of the system simultaneously on the basis of a numerical equation-solving procedure or some equivalent method.

Although SHAH propagates this classification with regard to constraints in shape models, it analogously applies to constraints on feature, part and assembly parameters. SHAH also reports that some disadvantages of variational models have led some researchers to look at hybrid techniques that try to decouple constraint equations in subsets that can be processed sequentially; see e.g. [43] [44].

Variational constraint systems within CAD systems are typically under- or well-determined, over-constrained systems also occur. An overview on relevant strategies and algorithms may be found in, e.g. [45]. The interested reader may also delve into [46] as one exemplary description of representation and solving strategies of a geometric constraint solver. More actual discussions on the subject matter are presented by LEE ET. AL. [47]. A comprehensive introduction to constraint programming is given by FRÜHWIRTH [48]. GUESGEN and HERZTBERG discuss specific consistency questions of constraint systems [49].

Free-Formed Shape and Feature Modelling and Representation

In addition to analytically represented shape elements created by form-features, CAD systems also support the generation of surfaces and curves that cannot be analytically described, i.e. they cannot be represented by simple non-parametric equations. These complex curves and surfaces are also referred to as free-form shape [10]. In the field of mechanical engineering, free-form elements are applied for the design of outer skin and vertical frame of airplanes, for propeller design in shipbuilding, for car body design, and further more [50].

Parameterised representations for free-from shape are of the form $\mathbf{x} = \mathbf{x}(\mathbf{u})$ for curves, and $\mathbf{x} = \mathbf{x}(\mathbf{u},\mathbf{v})$ for surfaces. An overview on approximation methods for the system internal representation of free-form shape is given by MÜLLER [51]. The de-facto standard for free-from representation is non-uniform rational B-splines (NURBS) as they contain most of the geometry formats used in existing CAD/CAM systems, and are the most flexible transfer and storage format existing for sculptured surface geometry [52] as special cases. More detailed information about the theoretical fundament of NURBS and their application can be found in [53], [54], [55], [56]. In the future, free-form representations based on subdivision surfaces could replace NURBS as a mathematical basis due to their “more intelligent” behaviour [57], [58]. At the time of writing, subdivision surfaces are, however, not state of the art in free-form representation in commercial CAD systems.

Due to many of today's CAD systems being hybrid in modelling solids and surfaces, free-form generation is supported via curve and surface generating free-form features. These elements can again be used to generate or manipulate solid elements, e.g. generate a solid via a loft feature based on a free form curve; KRAUSE / STIEL propose feature modelling functionality for free-from shape in [59].

Assembly Modelling and Representation

The assembly model is needed to drive several engineering analyses and applications like interference detection between parts, motion simulation, constraint satisfaction, assembly analysis, and assembly manufacturing planning [60]. According to SHAH / MÄNTYLÄ, the information that needs to be captured and represented at the assembly level by an assembly modeller includes the following:

- Hierarchical relations (assemblies, sub-assemblies, components, features, etc.);
- Mating conditions (geometric constraints, fits, contact, etc.);
- Component / sub-assembly positions (global or relative);
- Degrees of freedom (possible relative motions of parts or sub-assemblies).

Positioning of assembly entities is achieved by coordinate referencing, which requires all positioned entities to have their own coordinate system, or – preferably, because more flexible in case of model changes – by mating conditions, e.g. facing or coplanar faces, co-axial axes, coincident points. SHAH / MÄNTYLÄ also emphasize the importance of assembly features that allow assembly creation at a higher level by storing mating and constraint information and thus enabling parametric feature modelling functionality rather than geometric constraint handling at shape level.

In principle, the situation in constraint handling in assemblies is little different from that in part modelling. In detail, assembly constraints may be used for specific questions like assembly testing and tolerance handling. KRAMER has developed a degree of freedom approach for solving assembly constraints [61].

NOORT presents an integrated part and assembly modelling method and system. A multiple-view technique is applied to support feature-based part and assembly modelling for a product. A special system environment is proposed to support both part and assembly design [62].

2.1.2 System External Model Representation

General Approaches to the Exchange of Product Model Data

Since the 1950s, when the use of computer programs for defining products and manufacturing was first experienced, there was a need to exchange the corresponding product descriptions among these programs. Product descriptions emerged from

simple drawing information to highly complex data models handled by today's generation of product development systems. In the same time, strategies for data exchange emerged from ad hoc solutions for system-to-system data exchange to the use of neutral formats. Latter are those approaches to data exchange that the following summary will concentrate on. The interested reader may consult, e.g. [63], for more detailed descriptions of the development of data exchange strategies and formats.

In the domain of data exchange within the whole product life cycle, the Standard for the Exchange of Product Model Data – STEP (ISO 10303) was identified as a key technology already in 1990 [64], [65], [66]. Today, the STEP technology is considered the third generation of data exchange strategies. The STEP Product Data Representation and Exchange standardisation initiative covers computer-interpretable representation of product data, and its exchange. The objective of ISO 10303 is to provide means of describing product data throughout the product life cycle independent from any particular computer system. A general introduction to the STEP methodology is given in, e.g. [67], [68], [69], [70].

Shape Representation

In part 42 STEP defines general shape representation constructs that are applied in almost every product data model concerned with shape content in any form [23]. This schema is the basic understanding of shape modelling capabilities and shape model management that emerged from the early IGES, VDA and SET standards until today. Nevertheless, it is worth mentioning that many fundamental concepts were taken over from the ACIS [11] shape modelling library that is still available and applied as a commercial tool. STEP part 42 specifies geometry, topology and – based on these two – geometric model schemas for explicit representation of the shape of a product model. Both, b-rep and CSG representations are provided.

Procedural approaches to represent CAD shape model content started using general-purpose programming languages, such as Lisp or C [71]. An approach to represent shape model descriptions using a special-purpose formal language was developed by KRAUSE ET. AL. in the IMPPACT project at the Technical University of Berlin [72]. The IMPPACT modeller used the textual geometry definition language PDGL (Part Design Graph Language). Different aspects of the geometry definition covered by PDGL include geometry of form-features, derivable attributes, constraints, rules, and technology definitions for process planning applications.

A rather shape-related procedural approach to constructive constraint-based model representation is proposed by SOLANO / RUNET [73].

SHAH describes the *ASU Testbed Procedural Language* as an approach to represent feature specifications and CSG operations for geometric modelling using a procedural language for geometry definition [74]. The approach aims at the representation of so-called feature producing volumes (FPV) in a formal language by describing the generation and manipulation of the CAD shape model and its entities.

Feature Harmonisation and Library Concepts

There are projects and standards to be named that mainly, or as part of a larger product model, harmonise CAD design elements.

A feature taxonomy spanning a larger variety of design and manufacturing features has been defined in the *CAM/I* project [75]. A sheet metal features classification can be found in [60]

STEP shows various approaches to standardise design objects:

- ISO 10303-48 [76] was one of the first attempts to standardise form-features within the STEP methodology. Scope of part 48 was to provide for the characterisation and representation of shapes that are of broad industrial interest. These shape generating features were parameterised and strictly limited to shape relating content, i.e. they did not to contain any semantic meaning apart from the ability to identify themselves through the corresponding shape, which was to be represented by means of STEP part 42. ISO 10303-48 was abandoned before being finished.

- ISO 10303-214 is a so-called application protocol. Its scope of application comprises products of automotive manufacturers and of their suppliers including parts, assemblies of parts, tools, assemblies of tools, and raw materials [77]. The conformance classes CC14/15 specifies form-features for shape representation sub-structures. Two methods are supported: Firstly, form-features can be defined through recognition of represented b-rep sub-structures as a unit. Secondly, for multiple instantiation, b-rep sub-structures may be pre-defined as a form-feature. These rather macro-like approaches are specifically suitably for pattern instantiation. Other typical feature content, e.g. parametric or semantic information, is not represented.

- ISO 10303-224 [78] is another STEP application protocol. Its scope is mechanical product definition for process planning using machining features. AP224 supports the definition of a mechanical piece part, utilising form-features that are of particular value in process planning. The geometry of slots, holes, faces, etc., can be explicitly stated using these features, which can then be related during process planning to machining operations. The aim is to enable standardised exchange of product data to manufacturing planning software rather than among mechanical design systems. The feature information does not necessarily reflect a design history, nor is it meant to enable later alterability of the feature-based model.

- ISO 10303-111 Construction history features [79] specifies the resource constructs for the representation of modelling or design features supported by modern CAD systems. It is intended to facilitate the exchange of intelligent models that contain the history of the design operations. In scope are features that represent operations that capture filleting and chamfering; representation of features that enable the result of offsetting, thickening, and shelling operations; features that model operations associated with several types of

holes, pockets, and slots; representation of patterns of features, such as circular and rectangular arrangements. Out of scope is representation of features as aspects of the shape of a model, and features that explicitly relate to manufacturing operations or processes. The documentation has the status of a working draft and represents ongoing work at concept level.

HOFFMANN'S *Editable Representation* (EREP) [80] has been an early attempt to use a standardised set of features for model exchange purposes. The sequence of feature creation, modification and deletion operations should be captured and replayed for model regeneration. The concept is limited to pure feature modelling and does not cover shape based modelling operations like revolves and protrusions based on 2D sketches or shape manipulations like blends and chamfers.

Specification of Feature Types and User-Defined Features

At the Institute for Machine Tools and Factory Management (IWF) of the Technical University of Berlin, a so-called *Feature Modelling System* – FEAMOS has been developed [81]. The FEAMOS system demonstrates how internal management mechanisms and interrelationships between feature model and shape model can be organised. A similar approach has led to the development of a feature-based modeller for prismatic work pieces at the Institute for Production Engineering and Machine Tools (IFW) of the University of Hannover. The so-called *Elementorientiertes Modelliersystem* – EMOS [82] (element-oriented modelling system; an technical element is regarded as a synonym to a feature) aims at generating models specifically suitable for cutting machining operation planning.

The further development of PDGL (see above) has facilitated representation of feature types as a general modelling basis for the FEAMOS system. With the procedural TEBES language (*Technische Elemente Beschreibungssprache* – technical elements description language) a similar approach occurred for the EMOS system [83] [82]. The languages provide constructs for the declaration of features, their parameters and for the specification of functional relations between these parameters. Additionally, feature functionality can be described through object methods. A creation method, analogously to a constructor of a C++ class, is obligatory for the instantiation of feature objects by the feature modeller [41]. The languages differ in their main emphasis of application. While TEBES puts the main emphasis on workpiece modelling (with sculptured surfaces) and the integration of CAD/CAPP (vertical data exchange), PDGL puts the main emphasis on design and modelling (representation of assemblies and modular feature libraries) and horizontal data exchange [84]. Common to both languages is the procedural declaration of shape down to the level of topological b-rep faces. They also enable UDF specification via a CSG operation tree. A UDF is represented as a shape modelling result, i.e. the feature is represented without modelling structure on feature-level.

MUN ET. AL. present a set of *standard modelling commands* as part of the macro-parametric approach (see below) [85]. In five levels, a hierarchy of standard modelling commands is proposed that unifies capabilities of the six commercial

CAD systems CATIA, Pro/Engineer, Unigraphics, I-DEAS, Solid-Works, and SolidEdge.

Representation of Parametric and Constraint Information

Representation of parametric and constraint information has been of interest in the area of parametric and constraint-based shape modelling before recently the achieved mechanisms were also applied to feature-based models.

SHAH distinguishes between two approaches to parametric geometry definition for data exchange [42]:

Procedural parametric geometry definition: Geometry definitions are encoded in a procedural language that combines the definition of geometric entities and relationships among the entities with procedures for computing the resulting geometry.

Declarative parametric geometry definition: Geometry definitions are encoded in a nonprocedural language where the declaration of entities and their relationships is decoupled from the computation of the same. Roles and constraints are two important methods for declarative geometry definition.

The *ENGEN* project (Enabling Next GENeration mechanical design) has aimed at extending STEP EXPRESS information models with parameterisation and constraint information [86]. The emerged ENGEN Data Model (EDM) mainly explicitly represents shape model information similar to ISO 10303-42.

SHAH, KHAN and SOLKHAN describe constraint specification and constraint representation techniques and algorithms for geometric constraint specification in declarative modelling. Here, shape entities and their parametric interrelationships are represented and solved with the aid of hierarchical constraint graphs. The concepts emerged from the ENGEN project [87].

ISO 10303-108 [88] is currently under development. It will provide facilities for representing parameterisation and geometric constraints, as they apply to explicit shape models. The topic of features is not addressed in the part 108 context [89]. Concepts from the ENGEN project will be included and made compatible with other STEP shape containing product models. As a basis for STEP part 108 SHAH / BETTIG describe a standard set of geometric constraints for parametric modelling and data exchange [90].

CHOI ET. AL. propose a macro-parametric approach to exchange design intent in CAD models between different CAD systems [91]. The models are exchanged in the form of macro files, which are a sequence of modelling commands used by the designer in the modelling process (Figure 2-3). Since the approach necessarily includes the transcription of user selection operations, an identification mechanism for selected shape entities is needed. Missing information for the mapping of selected

entities is generated by the use of an external shape model generated by an ACIS kernel.

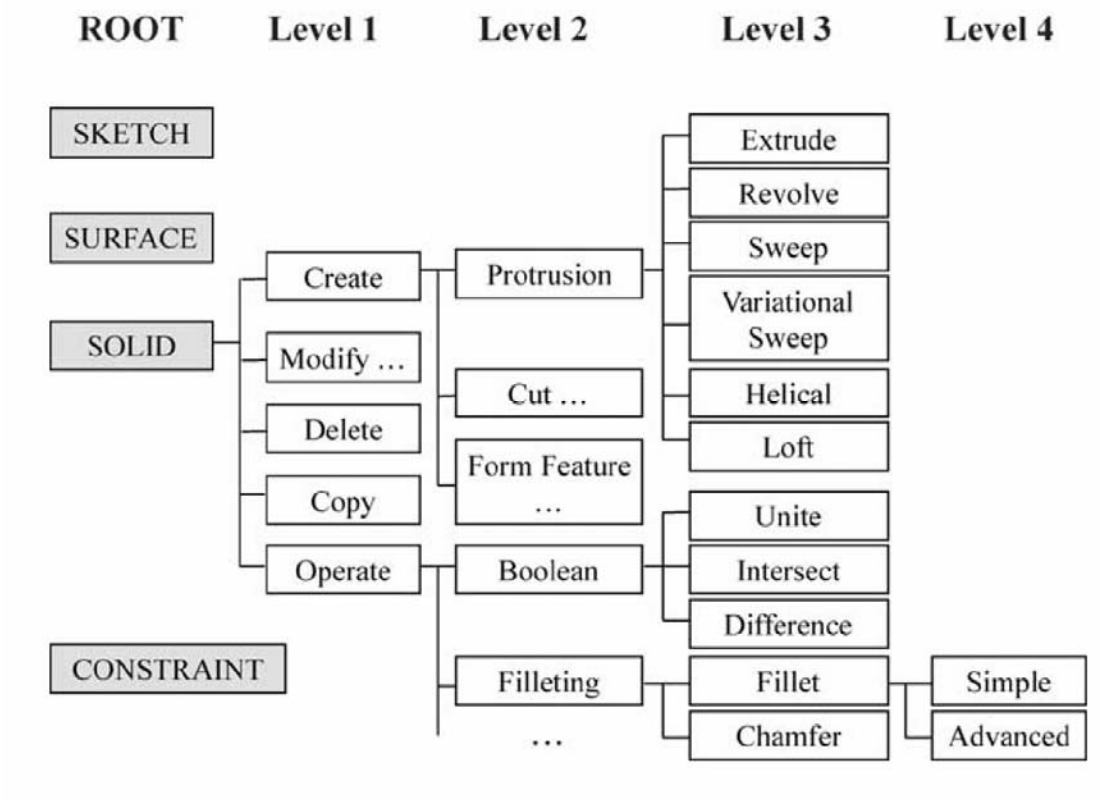


Figure 2-3 Standard modelling commands for the macro-parametric approach [91]

Free-Form Shape and Feature Representation

The established standardised exchange formats, such as IGES, VDA-FS, and STEP, delimit the representation of free-form model information to pure shape. They are based on an explicit representation.

AURICH [92] distinguishes between the implicit and the explicit representation of free-form features: The implicit form describes a free-form feature through a set of general, simple, and additional complex geometric attributes and their values. The explicit form consists of a number of faces, at least one of them being free-formed and controlled by a set of simple and complex parameters that result in a non-manifold volume.

TÖNSHOFF provides an overview and evaluation of approaches on free-form feature classifications [93]. He also proposes a representation schema for free-formed features.

Assembly Representation

There are various methods and schemas for assembly representation in existing standards. Approaches to system independent representation of assembly models can be found in, e.g. [94], [95], [96], [97].

STEP considers assembly structure in various parts. AP214 being the most applied exchange standard for mechanical parts, provides means for assembly design with 3D shape representation (CC2), and product data managements purposes and configuration controlled design (CC6–10). Latter are supposed to enable exchange of management data between CAD and PDM rather than between CAD systems [77]. An integrated application resource part 109 is currently under development and will address cinematic and geometric constraints for assembly models.

A strategy for mapping product structures between CAD and PDM using UML is proposed by OH ET. AL. [98]. The mechanism is based on the STEP AP214 PDM schema and concentrates on mapping part structures within an assembly model.

ZHA presents in [99] a STEP-based data model that supports integrative design for assembly method and a corresponding assembly planning algorithm. The assembly data model contains connector features as a means for inter-part relationships instead of direct positioning. The data model is the basis for knowledge-based assembly planning procedures based on an assembly process model. His concepts are based on findings by HEISSEMANN/MATTIKALLI about representing relationships in hierarchical assemblies [100].

2.2 Model Transfer Technologies

Model transfer technologies imply all techniques that either transport information items from one system to another, or facilitate the interconnection of computer applications. An information item in this connection may be a file or data stream compliant with a certain specification or schema. Of all those means for system interconnection, generalised interfaces, i.e. APIs, and communicational infrastructures, are of the most interest for standardised data exchange.

Exchange Formats for Product Model Data

The STEP methodology specifies its own model transfer formats. For consistent, contradiction-free and semantically explicit description of STEP product models, the formal description language *EXPRESS* and its graphical representation *EXPRESS-G* have been defined [101]. The EXPRESS data modelling language was published as the first international standard for the specification of data models. Due to an object-oriented approach, various kinds of model information can be represented:

- Objects (entities) with properties (attributes);
- Inheritance rules;

- Integrity conditions (local rules for objects and global rules for overall object characteristics);
- Object classes (schemas);
- Relationships between object classes (schema-interoperability).

EXPRESS is a specification language for the logical description of information models, i.e. it is not designed for programming purposes. EXPRESS has object-oriented characteristics as well as such defined through the entity relationship method. It enables the formal, unambiguous, and complete description of a static product model through objects, relations and conditions.

STEP also defines different strategies for product model transfer. The *physical file specification* was designed to transfer EXPRESS data models as clear text encoded files. The mapping from the EXPRESS language to the syntax of the exchange structure is specified in ISO 10303-21 [102]. Due to its EXPRESS-orientation, STEP physical files are not able to represent parametric information and constraints. There are activities within the STEP community to extend part 21 accordingly. Figure 2-4 illustrates the representation of two-dimensional objects by an EXPRESS class description and the mapping to a STEP physical file [103].

The STEP *Standard Data Access Interface* (SDAI) defines a low-level application programmable interface to EXPRESS defined data. STEP defines a set of general SDAI operations in ISO 10303-22 [104]. These operations are implemented in a specific programming language by a language binding. SDAI bindings have been defined for C, C++, and Java. Functionality for manipulating EXPRESS-based data is provided, e.g. set attribute A with value B, which can be used with any application protocol or other EXPRESS information model. SDAI does not provide any functions that understand higher-level semantics, e.g. get/set the owner of a product, get/set the length unit that applies to the product geometry [105].

Other formats for information transfer have emerged for and within the world-wide web. The *Extensible Markup Language* (XML) [106] was originally envisioned as a language for defining new document formats. XML is derived from the *Standard Generalized Markup Language* (SGML, ISO 8879), and can be considered a meta-language, i.e. a language for defining markup languages. SGML and XML are text-based formats that provide mechanisms for describing document structures using markup tags or key words. As the use of XML has grown, it is now generally accepted that XML is not only useful for describing new document formats for the Web but is also suitable for describing structured data. Examples of structured data include information that is typically contained in spreadsheets, program configuration files, and network protocols. XML is preferable to previous data formats as it is capable of representing any kind of structured information. This has led to the widespread adoption of XML as the lingua franca of information interchange [107]. Further information on XML and XML-based programming is provided, e.g. by GOLDFARB and PRESCOD [108].

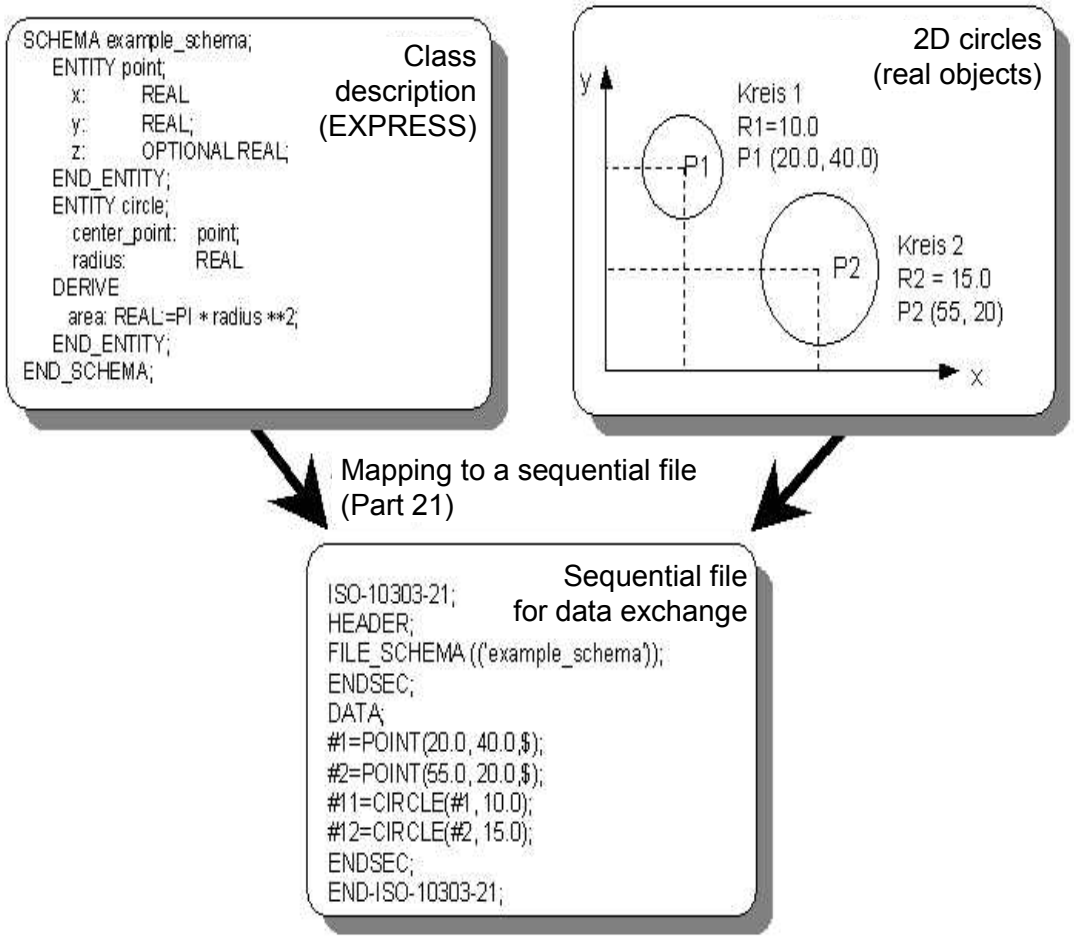


Figure 2-4 EXPRESS class description of two-dimensional circles and mapping to STEP physical file [103]

Product Data Markup Language (PDML) is an Extensible Markup Language (XML) vocabulary designed to support the interchange of product information among commercial systems, such as PDM systems, or government systems, such as JEDMICS. PDML is being developed as part of the Product Data Interoperability (PDI) project under the sponsorship of the Joint Electronic Commerce Program Office (JECPO), and supported by several other federal government agencies and commercial entities in the USA. Three major PDM vendors are active participants in the PDI program and are developing prototype implementations of PDML. The initial focus of PDML development is legacy product data systems that support the operation of the Defence Logistics Agency (DLA) [109], [110].

Generalising Interfaces for CAD Systems

Since the early days of commercial CAD systems, various projects have addressed the exchange of product model data between these systems on the basis of neutral

representation formats. There have been several activities to define standardised procedural interfaces, i.e. APIs, for CAD systems. The usual objective has been to provide a means of linking external application programs to CAD systems in a modeller-independent manner. ROLLER describes the value of an open CAD system architecture for the implementation of higher level design applications [111].

As early as 1979, the international research consortium CAM-I (Consortium for Advanced Manufacturing International, Inc.), Texas, USA, started to develop the *CAM-I Applications Interface Specification* (AIS) [112]. AIS should provide a standardized procedural or programming interface for CAD solid modelling systems, as a means for accessing their internal functionality and to facilitate the creation of integrated design and manufacturing systems. The object-oriented specification is independent from a specific programming language (with a C binding available), covers a lager scale of ISO 10303-42 entities and concentrates on solid shape modelling. Parameterisation and constraints are somewhat addressed on that level. Some versions of the AIS specifications were implemented and tested in practical implementations [113], [75]. A broader support by CAD software vendors is not noticeable.

Another procedural CAD interface for a standardised CAD model exchange has been defined by the CAD*I project [114].

The solid modeller API specification *Djinn* [115], [116], [117] unifies solid modelling capabilities on shape level. The specification is independent from the shape representation form and thus provides mapping on both b-rep and CSG data structures. Djinn does not support parametric, constraint-based or feature modelling.

For CAD systems residing on PC platforms and Microsoft operating systems, the Design and Manufacturing Automation Corporation (DMAC) [118] defines a runtime CAD interface based on Microsoft's OLE (Object Linking and Embedding) technology. The so-called *OLE for Design and Modelling*, provides query operations to CAD model entities, whereas model generation functionality is not addressed. Entities are accessible at the level of shape entities, shape structure and assembly structure. Persistent identifiers are provided. The full specification is available to DMAC members only.

ISO 13584 (*Parts Library*) [119] is concerned with the standardised means of representation and information access for standard parts in computer-based libraries. Parts Library facilitates the parameterised representation of families of parts, to avoid the need for separate representations of each member of what may be an extensive collection. Part 31 deals with geometric programming interfaces and defines a procedural interface for the generation of parameterised product shape models. The aim of part 31 is to enable shape modelling systems to implant standard parts from the library into the CAD system. Its model creation capabilities are based on a very limited subset of ISO 10303-42 and thus concentrate on pure shape modelling operations for CSG representation. Although its models are parameterised, it does not allow the definition of geometric or part level constraints. Included are few query operations and no modification operations apart from parameter changes.

The *Open CASCADE* technology is a software development platform freely available in open source. It includes components for 3D surface and solid modelling, visualisation, data exchange and rapid application development [120]. Modelling capabilities mainly address shape modelling in terms of 2D and 3D geometry and topology. Form features are supported as geometric primitives and for more complicated shape modelling tasks like splits. Open CASCADE also provides interfaces for standard based data exchange based on IGES 5.3 and STEP. The STEP processor supports import and export of 3D geometry and topology data compliant with AP202 and AP214.

PRATT / ANDERSON propose a *shape modelling applications programming interface for the STEP standard* [89]. They summarise the required functionality for a general-purpose API to contain creation, deletion, modification and query operations for high-level and shape-level modelling. It is to provide functions for creation of all the fundamental shape-defining entities defined in ISO 10303-42 and ISO 10303-108.

The OMG CAD Services [121] specification is an interface standard for mechanical CAD systems that enables the interoperability of CAD, CAM and CAE tools. The aim is to provide an interface for seamless integration of engineering applications through a CORBA interface. The specification focuses on establishing mechanical CAD system interfaces that provide geometry and topology data to analysis and manufacturing applications and tools. Feature modelling capabilities are intended for future versions of the CAD Services specification.

Communicational Technologies for System Integration

For establishing inter-system communication, several technologies have been developed. Essential functionalities are referred to as *middleware* technologies, which provide a standardised layer of software between the underlying network and the application. Specific middleware technologies are Sockets, Remote Message Invocation (RMI), Distributed Component Object Model (DCOM), Common Object Request Broker Architecture (CORBA) and Web-services. Of all of these technologies, only those shall be discussed that have been applied by major engineering system integration initiatives.

The Object Management Group (OMG) is a non-profit consortium created in 1989 with the purpose of promoting theory and practice of object technology in distributed computing systems. One of the major standards that emerged from OMG activities is the *CORBA* standard [122]. CORBA specifies a system which provides interoperability between objects in a heterogeneous, distributed environment. The central component of CORBA is the Object Request Broker (ORB). It encompasses the entire communication infrastructure necessary to identify and locate objects, handle connection management and deliver data.

For Web-based environments, the *Simple Object Access Protocol* (SOAP) [123] has been developed as an XML-based protocol to let applications exchange information and access Web services. SOAP Version 1.2 is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It

uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics.

A *Web-service* is a software application identified by an Universal Resource Identifier (URI), whose interfaces and bindings are capable of being defined, described and discovered by XML artefacts, and supports direct interactions with other software applications using XML-based messages via internet-based protocols, such as SOAP. An overview on Web-services development is given by, e.g. [124], [125].

As application example for engineering application integration in a Web-environment the *PDTnet* project can be identified. Within the joint project the automotive industry has developed an industry-wide solution for the manufacturer/supplier integration in the area of product data applying available communication technologies. The results are claimed to also be valid for other industries. PDTnet provides the fundament for the electronic supply chain based on coordinated solutions and implemented standards [126]. Specifically, Web-services technologies, such as XML / SOAP, are applied to transfer STEP compliant data as specified by the automotive application protocol AP214 (Figure 2-5).

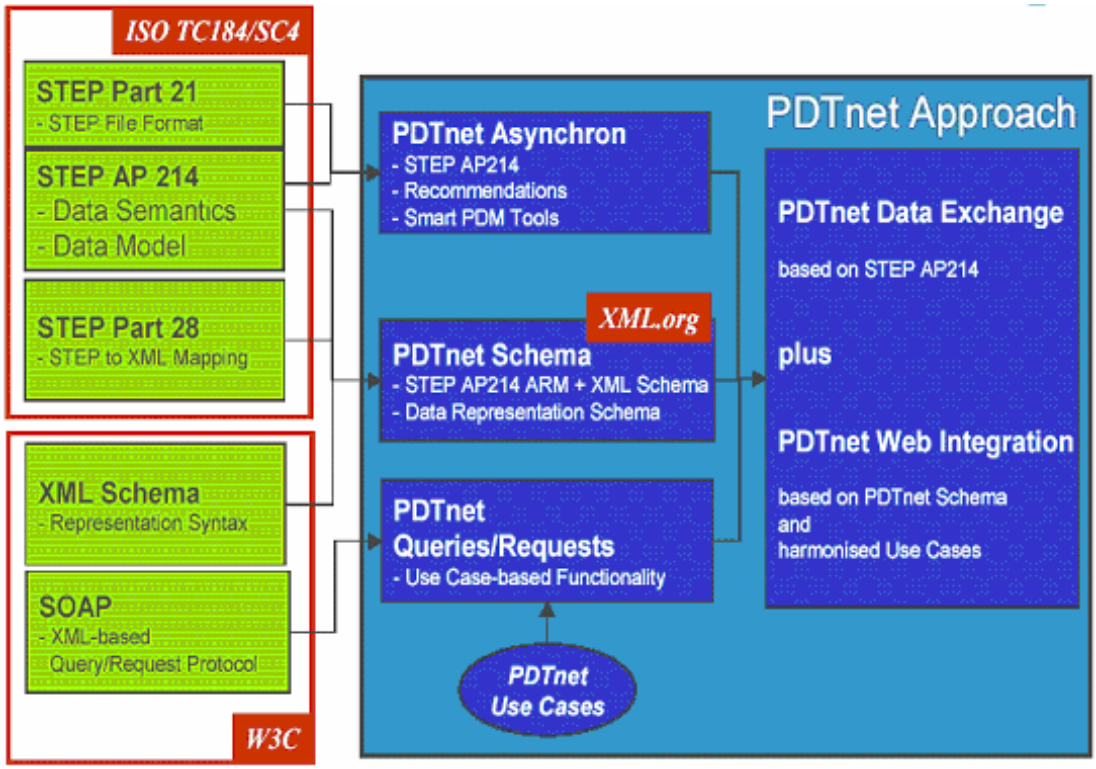


Figure 2-5 PDTnet approach and standardisation [126]

As another example for a engineering application integration initiative the German Lead-Project *integrated Virtual Product Creation* (integrierte Virtuelle Produktentstehung – iViP) can be highlighted [3]. As a means for on-line system integration, the iViP project has developed a CORBA-based iViP integration platform. This platform consists of a data bus, system services and an arbitrary dynamically changing number of components called iViP-tools, such as „Software on Demand" services, and one or more iViP Clients. The iViP Client provides a homogenous user interface to utilise components in a harmonised way, regardless different services functionality hosted by different applications. An Object Request Broker enables communication between platform, clients and the accessed components. Components that emerged from the project are realised with a CORBA compatible interface. Legacy systems are plugged in via a wrapping mechanism [127].

Finally, an abstraction philosophy shall be mentioned. The Object Modelling Group proposes the so-called *Model Driven Architecture* (MDA), which is a new way of writing specifications and developing applications without anticipation of a specific middle-ware technology, such as CORBA or Web-services. By this means, MDA divorces implementation details from business functions (Figure 2-6). Thus, it is not necessary to repeat the process of modelling an application or system's functionality and behaviour each time a new communication technology emerges. Key standards that make up the MDA suite of standards include Unified Modelling Language (UML); Meta-Object Facility (MOF); XML Meta-Data Interchange (XMI); and Common Warehouse Meta-model (CWM) [128].

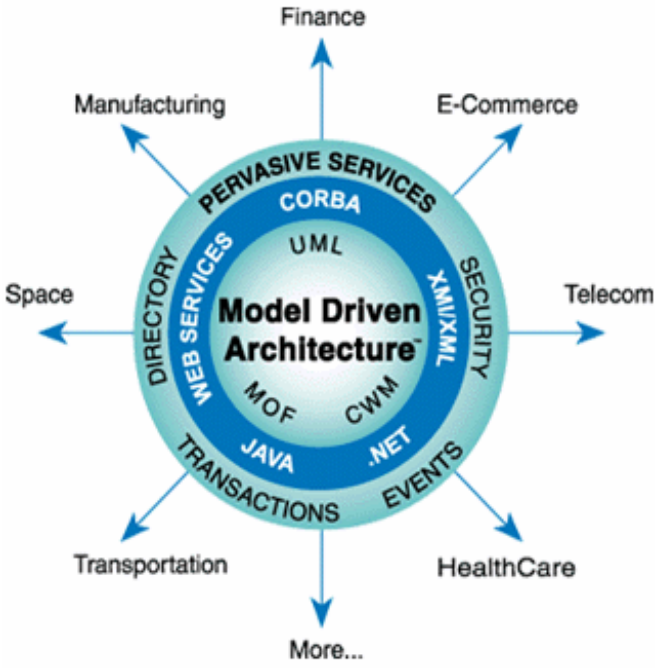


Figure 2-6 OMG Model Driven Architecture (MDA) [128]

2.3 Complementing Findings

Shape Model Entity Identification – Persistent Naming Problem

Out of a large number of research findings somewhat related to data exchange, the so-called persistent naming problem may be named as an input to later conceptual discussion.

As a means for internal data management, CAD systems need to be able to identify those elements of their shape model that for further modelling operations have been selected, i.e. picked, by the user. As an example, an edge of a body is selected by the user as an input to a blend or chamfer operation. Even if the original entity has lost validity due to later model alterations, the originally selected entity must remain identifiable for the system. For that purpose, the application of persistent identifiers has been proposed. The strategy for assigning persistent identifiers to shape entities is not trivial. In literature, it is referred to as the persistent naming problem [129].

For system internal application, some heuristic solutions to persistently naming of model entities have been proposed for shape modelling (see for example [130], [129], [131], [132]). KRIPAK describes an application for history-based parametric solid modelling systems in [133].

All these proposals do not provide a closed solution to the persistent naming problem. The authors limit their investigations to the scope of system internal model management. An adaptation to the area of product model exchange is not known.

3 Preconditions for the Structure Oriented Model Exchange

3.1 Identification of Technological Shortcomings

In the preceding chapter, the state of the art in product model exchange has been summarised. In the following, the presented findings will be analysed to identify technological gaps and shortcomings and to explain the need for a new exchange method. Objectives and scope of application will be defined from which a set of requirements will be derived that also suite as a means for results evaluation in chapter 5. Finally, a thesis statement will introduce the idea of a structure-oriented representation and transfer as a new approach to product model exchange.

The motivation for a new approach to data exchange has been discussed in chapter 1:

- In order to be efficient, product creation processes have to be integrated into process chains via an continuous information flow;
- Consequently, data exchange mechanisms have to become more efficient regarding quantity and aggregation of data through the product life cycle and more effective regarding the range of information types within a product model;
- Further processing and, specifically, alteration of the transferred product model must be guaranteed in the receiving system, while the basic information – the product shape – remains unchanged by the exchange procedure;
- Exchange of CAD model data can be identified as the most critical point within the product creation process.

As measured by this situation, currently available technologies for the exchange of product model data among CAD systems only limitedly provide sufficient solutions.

Although still subject to optimisation, exchange of shape models via STEP and other standardised data models is well supported and established in industry. Additional activities have started to address parametric and constraints information. Higher-level model information is only selectively represented. Feature information is either limited to shape-centred form elements, or to a certain modelling philosophy or application domain or system architecture.

Known exchange strategies mainly provide a static data exchange, i.e. they represent and transfer the result of a modelling process at exchange time. Apart from single

exceptions, possible alterations to the imported models within the receiving systems are not considered. The STEP methodology particularly focuses on transmission of the final shape model from which the original shape creating design elements cannot be reconstructed.

Regarding model transfer mechanisms, the established way of data transmission is based on physical files. Only a few activities have addressed different technologies and exchange situations.

The variety of scientific disquisitions permits a generalisation on CAD system internal modelling and representation mechanisms. For CAD model exchange, established standardised formats are not suitable. Available technologies and some of the ongoing activities and scientific approaches may suite as enablers for a new exchange method.

3.2 Determination of Objectives and Scope of Application

Objective of this dissertation is the development of a new method for the exchange of product model data. This method shall convey product models while retaining product shape and semantic information and while guaranteeing further processing and alterability within the receiving system.

The situation in engineering, as described in chapter 1, would require an adequate solution for the whole product creation process. However, from the engineering perspective it appears evident that data exchange during product definition phases, i.e. while product model is generated, is the fundament of all later product creation processes. Thus, the scope of application for this dissertation will be delimited to data exchange between CAD applications (Figure 3-1).

Explicitly, as in-scope will be regarded:

- The product design phase, namely embodiment and detailed design in which 3D shape modelling is the main focus;
- Product data exchange among different CAD systems, so-called horizontal exchange, based on a system neutral representation;
- Feature-based CAD systems due to their ability to manage hierarchical high-level structures above shape level, and for their provision of design-by-feature modelling mechanisms;
- Exchange of parts and assemblies, with the main emphasis on the former;
- General-purpose CAD modellers and models, i.e. no principle limitation regarding the type of product, although the main experiences of the author

descend from the field of mechanical design; specialties of, e.g. ECAD systems, would be interesting to examine but are not considered here.

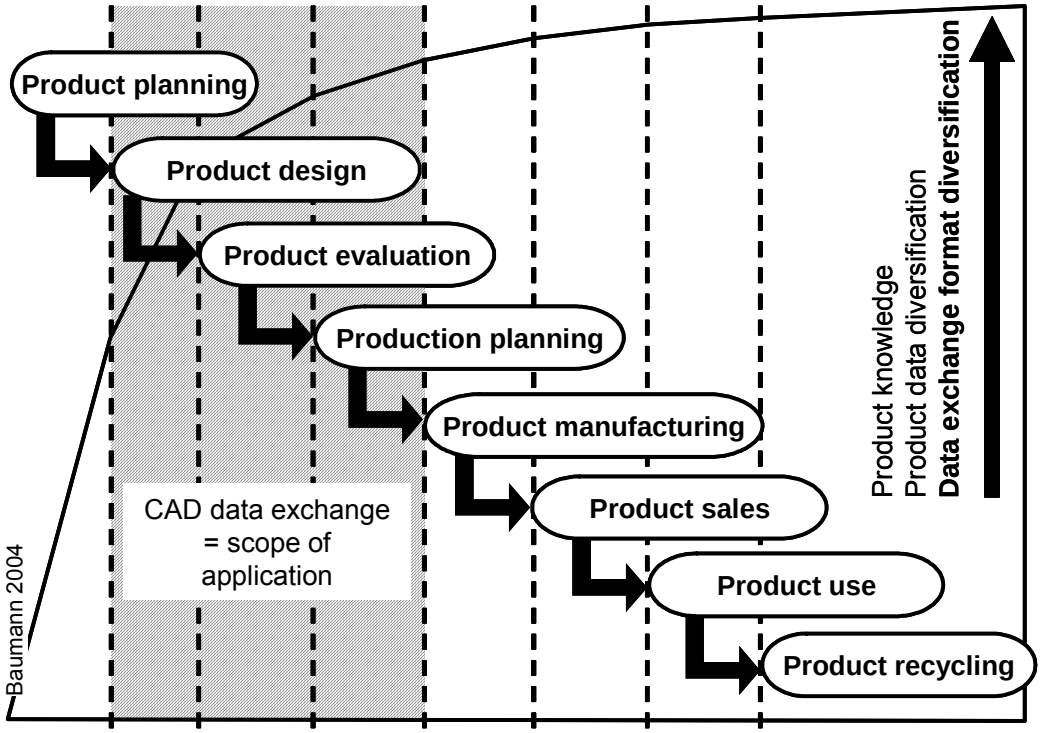


Figure 3-1 Thematic delimitation

The following aspects are regarded as being out-of-scope, which means that they are not considered regarding their specialties but also not explicitly excluded from the conceptual discussion:

- Pure shape modelling applications that do not handle and manage structural design elements on for instance feature level,
- Other CA system applied in design, like CAE systems for finite elements analysis, calculation, kinematic and other analysis tasks performed during part and assembly design, which require specialised data formats;
- Further methods like quality function deployment and failure mode and effect analysis;
- So-called vertical exchange, i.e. exchange from CAD to CAM, assembly planning, and other manufacturing planning applications;
- Representation and transfer functionality specific to CAD-to-PDM interfacing with special requirements on the mapping of CAD assembly hierarchy and part interrelationships onto the PDM part structure;

- Feature-based product models emerging from feature recognition methods; as recognised features usually cover only certain areas of the product shape and do not necessarily provide a complete view of the shape model.

3.3 Requirements Definition

Based on the engineering situation and industrial strategies, as illustrated in chapter 1, essential requirements on a new exchange method can be outlined from a user's perspective. These business requirements, shown in Table 3-1, shall be formulated with regard to the scope of application set above. They will also suite as a matter of decision making in the conceptual phase in chapters 4 and for result evaluation in chapter 5.

Business requirements
Retain design results: Guarantee for product shape to be transferred in its full spectrum and unaltered
Enable model alterability: Support design of different components and even regions of a single part by different people by enabling model alterability after data import from different CAD system
Shorten engineering cycle times: Reduce amount of time spent for non-creative jobs, i.e. for data exchange and manual rework of imported models
Reduce data traffic: Reduce file or equivalent data stream in size and transfer time
Support multi-CAD part library: Enable central administration of one part library for multiple CAD systems
Maintain design systematics: Transfer parametric and semantic model content
Support multi-disciplinary design: Support necessary data exchange between tools for mechanical, electrical, hydraulic, and software part design (horizontal alterability)
Support iterative process chains: Support data loop-backs between conceptual and embodiment design and technological planning processes (vertical alterability)
Support selective model analysis: Enable CAD model analysis with focus on special information types, e.g. on surface quality as a basis for technological planning or cost analysis
Support data management: Support small and large scale data distribution and management strategies

Support collaboration: Support modern Web-based exchange mechanisms
Provide fault tolerance: Ignore or compensate information incompatibilities even for different CAD architectures
Support CA systems interaction: Enable / enhance direct information exchange at systems run-time
Provide applicable strategy: Ensure feasibility for an industrial application

Table 3-1 Requirements on a new exchange method from a business' perspective

3.4 Thesis Postulation

At this point, a solution to the challenge of a new exchange method shall be formulated as a thesis statement:

1. The fundamental information that enables model alterability is contained in the product model structure, specifically in the feature and part structure of a CAD model; the new exchange method must therefore be *structure-oriented*, i.e. it must include structural information that is associated with the shape model and that the user still can alternate or change.

2. There are two possible approaches to a structure-oriented exchange, distinguished into the explicit and the implicit approach. Regarding shape information, the explicit approach is based on a discrete shape representation based on, for instance, the STEP methodology. The implicit approach utilises the structural feature model for data exchange, from which the shape model is regenerated by the receiving system.

3. Regarding requirements, the implicit approach is preferable and can be detailed to a structure-oriented exchange method. This method consists of two elements:

   - A method for a system neutral representation of CAD model data, and
   - A corresponding model transfer strategy supporting the actual data transmission process.

Both elements of the structure-oriented exchange method will be developed in the following chapter.

4 Concept of a Structure-Oriented Exchange Method

4.1 Characteristics Definition for a Structure-Oriented Exchange Method

In the preceding chapter, the objective of this dissertation has been defined as the development of a structure-oriented method for CAD model exchange. Requirements have been formulated from a business-perspective. This chapter deals with the development of a concept for the new exchange method. Firstly, central challenges are analysed by defining and discussing the characteristics of a structure orientation and by mapping requirements from a business into a technical view.

In the following sections, two general approaches to a structure-oriented CAD model exchange are developed from which one is selected according to technical requirements. A detailed model representation concept is developed for this approach. A subsequent definition of representative exchange scenarios leads to the development of model transfer strategy for the new exchange method. Finally, these individual conceptual parts will be synthesized to an overall technical concept for the realisation of a structure-oriented method for the exchange of CAD model data.

The two main aspects of the new exchange method have already been mentioned:

1. A model representation schema or data structure, i.e. a set of data objects and interdependencies between these objects, that in its whole represent the CAD model content outside the system's environment;

2. A model transfer strategy, including algorithms for converting the system internal from and to the system neutral model representation, a technical infrastructure for the interconnection of CAD systems and for transportation of model data streams between the systems involved in the exchange process.

In this context, the two most important requirements on the new exchange method, namely the conservation of product shape and the guarantee of later model alterability, primarily have influence on model representation. Thus, model representation is in the focus of the following discussion.

Conservation of product shape is a general requirement on every exchange method for CAD model data and has always been the main concern in the development of standardised exchange formats. As product shape defines the result of later manufacturing processes in terms of the physical border of a work piece, its dimensions, position, surface quality, etc., it quasi constitutes an invariant element in data exchange. From the shape representation perspective, this demands for the ability to regenerate a shape model with an alike – but not necessarily identical –

outer b-rep structure. A corresponding CSG structure may also differ from the original. This “alike” shape model shall be referred to as the *resulting shape.*

For a discussion of the second main requirement, the guarantee of model alterability within the receiving system, CAD capabilities for product modelling may be illustrated (Figure 4-1). The CAD system enables the user to create or delete shape generating or shape manipulating features, to define feature parameter values and constraints, and to order features within the feature structure of the part by the aid of Boolean operations. Considering the CAD system’s internal model representation, as generalised in the state of the art discussion, modelling operations and alterations of corresponding internal representation elements can be identified, as summarised in Table 4-1.

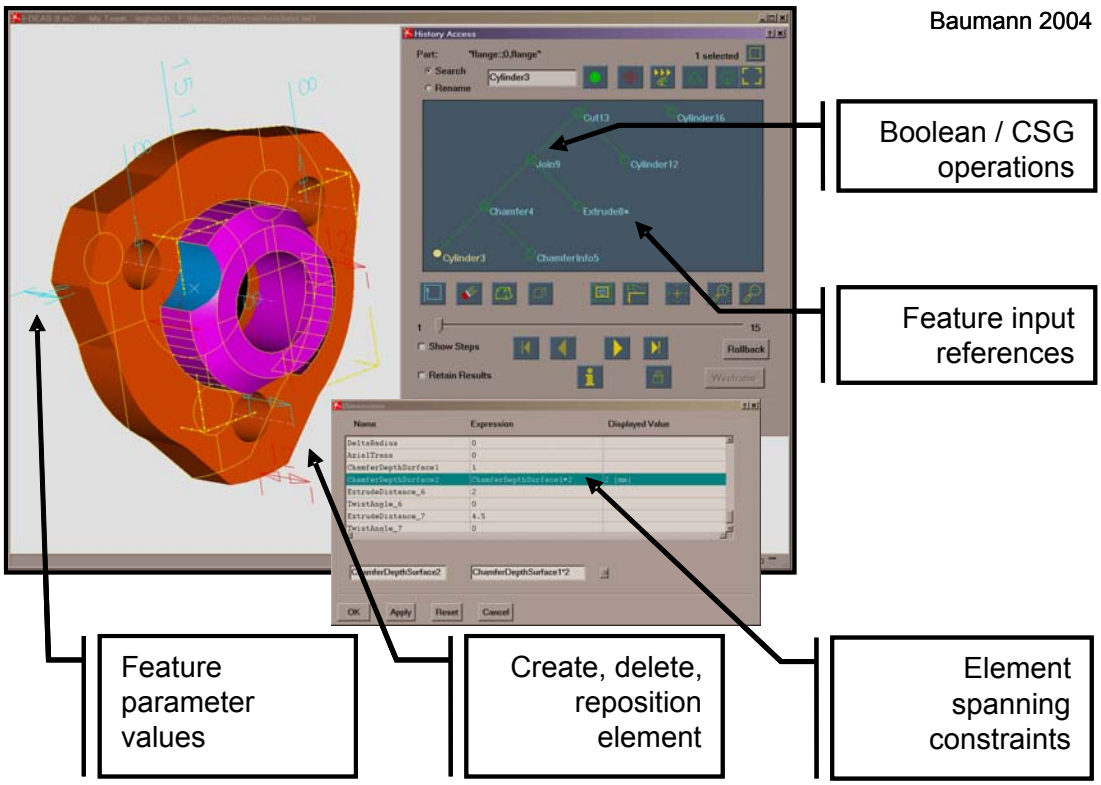


Figure 4-1 General CAD product modelling capabilities

CAD Modelling Operation	Corresponding CAD Model Elements
Create, delete model entities: Create a new assembly or part model, insert parts or features, coordinate systems, etc.	Assembly, part, feature entities, auxiliary geometry, coordinate systems
Reposition model entities: by direct positioning via translation or rotation with respect to coordinate systems or by indirect positioning via adjacency relationships of topological entities (mating faces, collinear edges, equal or opposed directions, etc.)	Coordinate systems, part mates, auxiliary geometry, transformation matrices
Modify parameter values of model elements, i.e. alter parameter values of one specific model element	Part, feature parameters, part mates, shape dimensions and distances, constraints on part / feature parameters
Alter element spanning constraints, i.e. change value of parameters defined independently from any specific element; define constraints spanning parameters of various elements	Feature and part independent parameters, constraints including parameters of more than one feature / part
Alter input references, e.g. redefine the set of edges to be blended	Features, feature parameters, auxiliary geometry, sketches
Change Boolean / CSG operations in elements hierarchy to arrange shape to be united or subtracted	Change semantic sign of Boolean operation on features / parts

Table 4-1 Correspondence of CAD model modifications with internal representation

An exchange method guaranteeing that an imported model is fully alterable by the user must represent all necessary elements that corresponding modelling operations rely on. An overview of possible modifications to a CAD model and effected representation elements is given in Figure 4-2. In any case, modifications to product shape are performed via alterations to high-level model entities, namely to either feature parameters or to the feature structure within the part model. Modifications to an assembly model are realised through alterations to the part structure, analogously. Specifically, the position of shape elements, dimensions, parameter values, and feature internal constraints are part of and are altered via feature attributes. Feature spanning constraints, position of and Boolean operations on features within the structural tree are managed and modified through the part structure. Shape model elements are only addressed directly when they have been explicitly defined by the user, e.g. sketched curves as a basis to extrude or revolve features. A modification of these elements is usually initiated through the corresponding feature. The same applies to curves for loft and sweep features and for shape manipulating features

creating blends or chamfers to a given set of edges. Apart from these exceptions, the b-rep structure is altered through variation of feature parameters.

It can be noted that CAD modelling operations performed by the user generate a high-level management structure. The instantiation of this structure results in the generation of the corresponding product shape. It can further be noted that all modifications to the CAD model are initiated through the same structural model elements, whereas the shape model is not subject to direct modifications. Exceptions are explicitly generated of referenced shape entities.

It can be concluded that these structures must be part of a system neutral representation in order to guarantee for alterability of the imported model and of product shape in particular. This prerequisites the representation of such structures in a system neutral form that is compatible with the wide range of CAD system internal representation mechanisms – or can at least be mapped onto them. This does not necessarily mean that the original model structure needs to be represented completely. Considering the objective of this dissertation, it is sufficient to represent only those model entities that are necessary for a regeneration and modification of the product model.

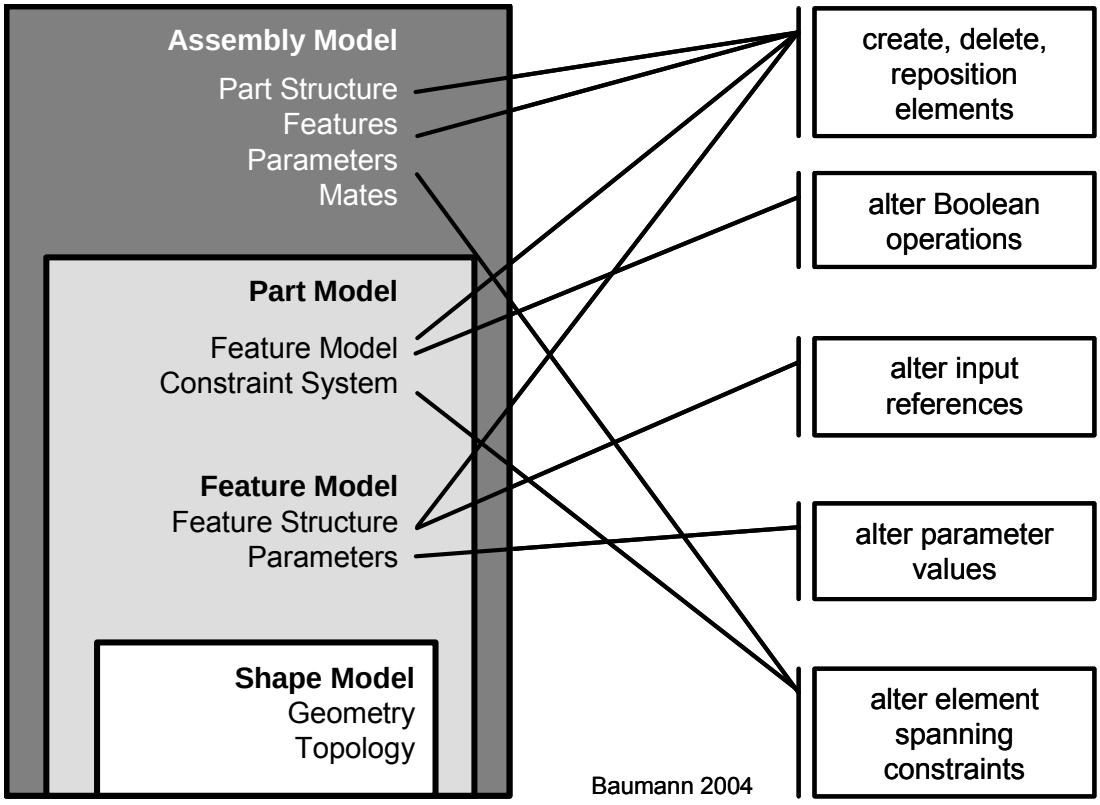


Figure 4-2 Correspondence of CAD model modification to internal representation

Regarding the actual semantic content and structural condition of a specific product model, it may be considered that not all imaginable modifications are sensible, and are supported by the CAD system only in certain circumstances. For instance, the system will probably allow the user to delete or suppress a feature which later modelling operations depend on, but will not be able to generate a proper shape model. Adjustments to dimensions and other feature parameters may result in inconsistencies and unpredictable behaviour of the shape model, usually due to implicit geometric constraints.

It is not the aim of an exchange method to heal inconsistencies of product models. Neither is its aim to enlarge the product modelling capabilities within the receiving CAD system. This leads to a hypothesis for the development of a new exchange method:

1. The product model to be exchanged is consistent within itself according to the internal representation of the sending system.

2. Alterations to the imported model are limited to the modelling capabilities of the receiving system. It is assumed that these alterations are meaningful according to the system internal model management structures and algorithms, i.e. they are performed without failures.

Summarising the characteristics of a new exchange method discussed above, the following system of concepts shall be defined as shown in Figure 4-3. The conservation of product shape requires the regeneration of a *resulting shape* representing the product's outer appearance. In that sense the new exchange method is to be *shape preserving*, i.e. is to transfer the resulting shape unchanged. Furthermore, the new exchange method must be *structure preserving*, i.e. it needs to be capable of representing and transferring the model structure allowing for later alterability within the receiving system. The combination of the latter two characteristics shall be referred to as *structure orientation*. The new exchange method will therefore be referred to as a *structure-oriented* exchange method.

structure-oriented	: =	structure preserving ∧ shape preserving
structure preserving	: =	representing and transferring a model structure that allows model alterations
shape preserving	: =	transferring the resulting shape unchanged
resulting shape	: =	subset of the product shape determining its outer appearance

Figure 4-3 Characteristics of a new method for the exchange of CAD model data

Having defined the characteristics of the new exchange method, the objective of this dissertation can be re-formulated as the development of a structure-oriented method for the exchange of CAD model data.

According to this objective, the requirements to this exchange method, as defined in section 3.3, can now be converted into a technical specification (Table 4-2). At this point, only model representation aspects are considered. Requirements regarding the model transfer strategy will be discussed in section 4.5.1.

Business Requirements	**Technical Requirements**
Retain design results: Guarantee for product shape to be transferred in its full spectrum and unaltered	Preserve resulting shape: Guarantee for resulting product shape to retain in its outer form, dimension, position
Enable model alterability: Guarantee for fully alterability of imported model within receiving CAD system	Provide model structure: Transfer a model structure as a means for fully alterability of assembly and part models
Shorten engineering cycle times: Reduce amount of time spent for non-creative jobs, i.e. for data exchange and manual rework of imported models	Shorten data exchange time: Minimize effort for manual model regeneration / post-processing after import
Reduce data traffic: Reduce file or equivalent data stream in size and transfer time	Reduce processed data volume: Reduce number of data bytes (compressed or exempt from white-spaces)
Support multi-CAD part library: Enable central administration of one part library for multiple CAD systems	Provide CAD independent feature specification: Provide standardised features; enable specification of user-defined features (UDF)
Maintain design systematics: Transfer parametric and semantic model content	Maintain model systematics: Support shape generating, shape manipulating, shape neutral, shape independent (semantic) features; transfer feature parameters and model constraint systems

Support multi-disciplinary design: Support distributed mechanical, electrical, hydraulic design (horizontal alterability)	Support CA model exchange: Enable regeneration of basic CAD model content to support design changes (at least minor ones) in e.g. ECAD systems
Support iterative process chains: Support data loop-backs between conceptual and embodiment design and technological planning (vertical alterability)	
Support selective model analysis: Enable CAD model analysis with focus on special information types, e.g. on surface quality as a basis for technological planning or cost analysis	Enable selective model analysis: Provide means for identification of model entity types and content specific model analysis; enable selection of model subset for regeneration

Table 4-2 Requirements on the structure-oriented model representation from business and technical perspective

4.2 Development and Discussion of Two General Methodical Approaches

4.2.1 Introduction of Two Approaches

After identifying the technical objective of this dissertation as the development of a structure-oriented exchange method, general approaches shall be discussed in this section.

In the state of the art summary, a so-called macro-parametric approach to data exchange has been named. CHOI ET. AL. propose a realisation that has long been discussed as the idea of a history-based data exchange. The idea is to protocol user interactions with the CAD system in log-file while he is modelling a product. This log-file can then suite as an input for a model regeneration algorithm.

There are several reasons why such a history-based logging mechanism is not sufficient for a structure-oriented data exchange method. At first the basic instruments of a CAD system's history has to be considered: Many systems generate a report of user interactions, i.e. a sequence of user interactions with the graphical interface. This sequence is usually stored in an operation list, or log-file, and represents the design history of the CAD model. The initial purpose of such a model history is the support of the CAD system's undo and redo functions. It contains *all* user actions including those that may have emerged from design mistakes and forced the user to delete and redesign significant parts of the model. It also contains selections, such as picking of edges as an input for a blend operation. In the case of a redo call, literal inputs, like object names and parameter values, are newly requested

from the user and are not consistently stored. Consequently, a re-processing of a history file is not possible without the user's assistance.

Another aspect is essential for the idea of using such a history file as a means for data exchange. User interactions as recorded according to the system's modelling philosophy. The kind and structure of a history file strongly depends on the internal modelling mechanisms and specifically on the level of granularity the system allows for atomic user operations, regarding e.g. input options and menu hierarchy. All modelling operations invoke internal procedures that the system provides to the interface. This set of procedures varies with the different "look-and-feel" concepts of commercial CAD systems. Furthermore, the set of available modelling procedures is usually not distributed to the public and is therefore of doubtable value for the development of standardised exchange methods.

It can be summarised that an approach to CAD model exchange that makes use of a history file, storing user modelling interactions with the system's graphical interface, is afflicted with technical problems and fundamental incompatibilities. It is also doubtful that these incompatibilities can be harmonised as part of a standardisation process.

There are two possible approaches to a structure-oriented method for the exchange of feature-based CAD models that shall be introduced and discussed in the following:

1. A shape-centred approach that makes use of standardised shape representation techniques provided by the STEP methodology and adds further structural information. This idea shall be referred to as the *explicit approach*.

2. A feature-centred approach that makes use of the structured feature model representation as the central information of a CAD model. It aims at solely exchanging the feature model and leaves the regeneration of the shape model to the modelling facilities of the receiving CAD system. This approach shall be referred to as the *implicit approach*.

4.2.2 The Explicit Approach

The explicit approach is based on the state of the art in CAD model representation. As summarised in chapter 2, STEP data models available today are basically shape-centred and provide a snap-shot of the model at exchange time. Several data structures must be added to the standardised shape model in order to achieve the characteristics of a structure-oriented exchange (Figure 4-4).

Analogously to the generalised CAD model architecture, an assembly model, a part model and a shape model is to be represented with uni- or bi-directional associations between their entities. These associations have to be represented at export time and re-established at import time.

On assembly level, a part structure model **F** (A,…,Z) is required representing a hierarchy of parameterised parts and assemblies. Positioning information and a system of part spanning constraints is also included. The representation of a part model requires a feature structure **f** (A,…,Z) with parameterised features as design elements and Boolean operations as structural elements. Positioning information and a system of features spanning constraints is necessary analogously.

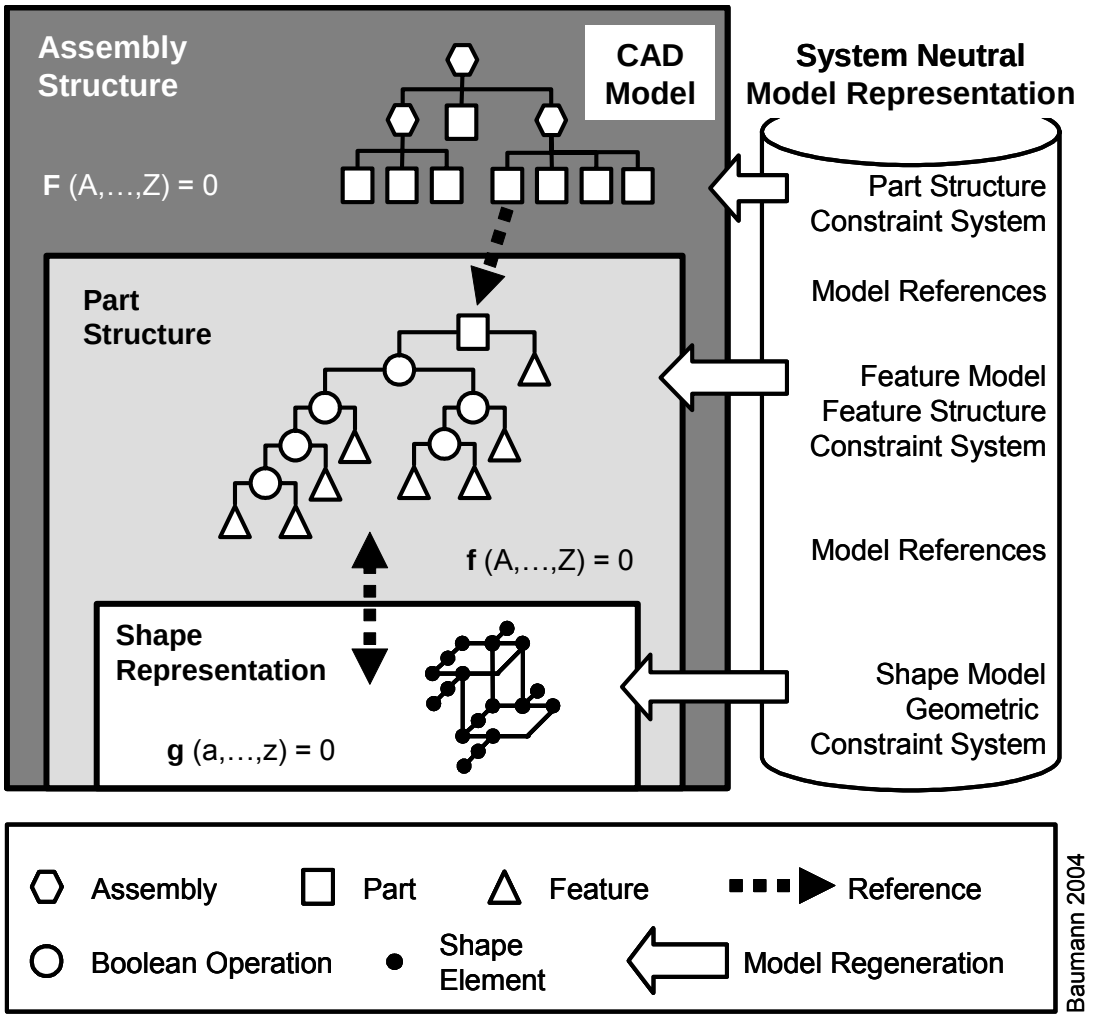


Figure 4-4 Explicit model representation

The representation of the corresponding shape model **g** (a,…,z) is shown in Figure 4-5. A feature structure of a part P can be understood as a tree-like hierarchy of instantiated design feature objects F_i in a certain status with a corresponding shape g (F_i). Every inner node B_i of this structure represents a design step with a corresponding resulting shape model g. Every additional inner node B results in a new shape model. The final part model results in the final shape model g (P).

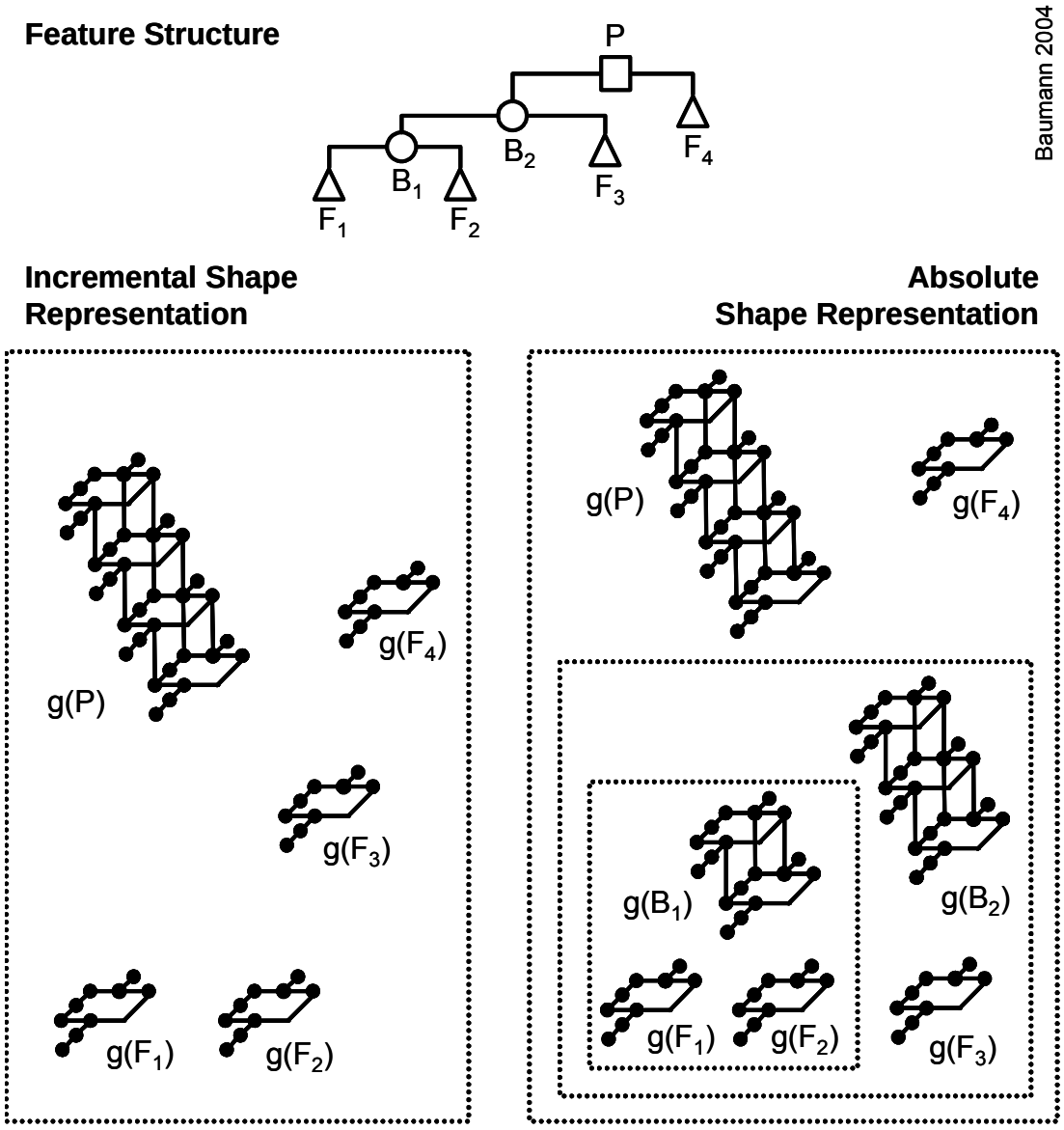


Figure 4-5 Explicit shape model representation strategies

In order to be alterable, the resulting shape model for every design step g (B_i) must be available. The simple representation of the resulting shape model, as provided by e.g. STEP data formats, is insufficient. Two strategies can be identified, to guarantee that every shape model status can be referenced:

1. An incremental shape representation strategy, or
2. An absolute shape representation strategy.

The incremental strategy represents the resulting shape model g (P) and additionally stores the shape generated by every single feature g (F_i). This way, an algorithm can compute the intermediate design steps. The absolute strategy represents a complete shape model for every design step g (B_i) so that it can be addressed directly.

The explicit approach to structure-oriented data exchange altogether has the advantage of availability of shape representation data formats. STEP part 42 and corresponding protocols such as AP214 can be applied. Schemas for the representation of constraints are under development. On the other hand, a suitable representation for features and feature structure is not available.

The main disadvantages of the explicit approach is the great shape representation effort. Both the incremental and the absolute shape representation strategy suffer from a strategic inconvenience: the shape models must either be represented or computed which is costly in either file size or exchange time. Independent from the chosen strategy, the resulting data volume will significantly extend the size of conventional STEP files.

Another disadvantage can be identified in the unification of system internal model representation strategies that would be the result of a standardised explicit data model. Experiences from the development and application of STEP AP214 processors have shown that the specification of a representation schema constitutes a unifying abstraction of the system internal data model – with the natural incompatibilities that such an abstraction always results in. The history of commercial AP214 processor benchmarks highlight the great difficulties in overcoming these incompatibilities [134], [135]. This problem will principally be aggravated when structural representation schemas for the feature models are specified whose data items are strongly associated with the shape model. Constraint systems and positioning mechanisms must also be included. A standardisation of these schemas is comparable with the standardisation of the complete system internal modelling algorithm. This in itself appears to be an enormous challenge.

4.2.3 The Implicit Approach

Compared to the explicit approach to structure-oriented data exchange the implicit approach operates the opposite way: it does not aspire to represent the complete CAD model including feature structure and shape model. Instead, it aims at transferring as much information as possible with the smallest data volume possible.

The central idea emerges from the discussion about design capabilities for model generation and modification in section 4.1. It has been identified that high-level model structures, i.e. feature model and structure, are those elements of the CAD model that are affected by user modifications. It has also been noted that these model structure are the same elements, which are generated while the product is initially modelled. This leads to the conclusion that the system neutral model representation can be limited to these high-level structures, whereas representation of the actual shape model content is obsolete.

As shown in Figure 4-6, an implicit exchange of a part model requires the representation of the feature model, i.e. feature structure, features, and constraint system. Positioning information and feature parameter values are part of the feature information. Only this high-level information is transferred to the receiving system, whereas the regeneration of the resulting shape model is complete left to its own modelling capabilities, similar to the original design operations invoked by the user. Associations between part model and shape model entities are established automatically by the receiving system.

On assembly level, part structure and additional features, part spanning constraints and positioning information must be represented. At that point, the necessary effort does not essentially differ from the explicit exchange.

The main challenge of the implicit approach lies within the representation of shape entities that were directly modelled by the user or are directly referenced by a shape manipulating feature, as identified in chapter 2. If shape information is not subject to data exchange, an alternative mechanism must be found. This specialty will be discussed in section 4.3.4. At present, it may be stated that explicitly represented shape elements can well be handled within an implicit exchange method.

An advantage of the implicit approach is the comparably small amount of necessary data sets. As the shape model is not part of the neutral model representation a significant reduction of transferred data volume can be expected. Another advantage is the high abstraction level of the corresponding representation schema. As the number of high-level model entities is comparably small to the number of shape entities, a harmonisation can be limited to a relatively small section of CAD modelling capabilities.

In general, the strategy of the implicit approach is to make optimal use of the CAD system’s modelling capabilities instead of limiting them due to a rigid harmonisation.

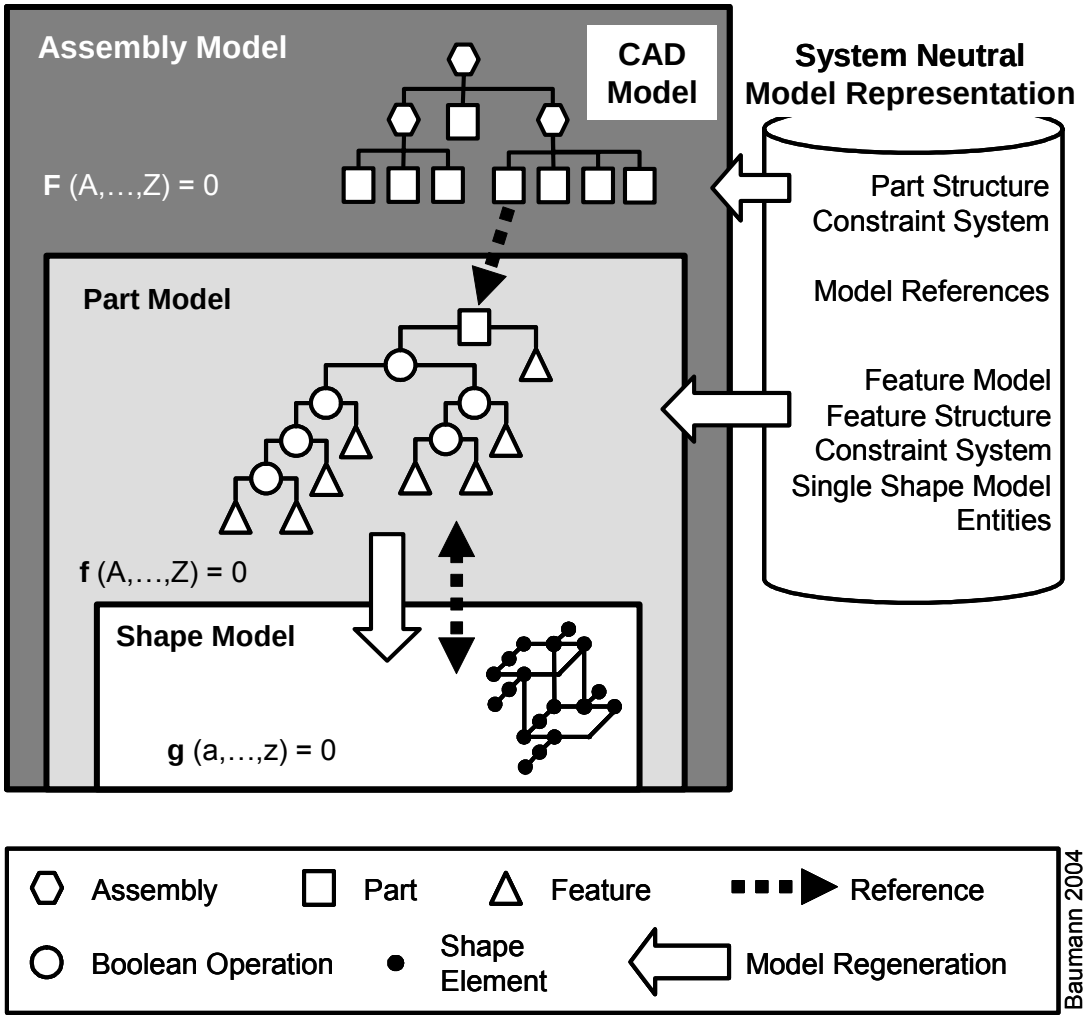


Figure 4-6 Implicit model representation

4.2.4 Comparison and Selection

After introducing the implicit and the explicit approach to structure-oriented exchange, both approaches are to be assessed. This will provide a basis for a selection of one of these two approaches for a detailed conceptual development. The technical requirements on model representation, as set up in section 4.1 (Table 4-2), suite as assessment criteria. Requirements regarding aspects of model transfer have little influence on the assessment. These will be discussed in section 4.5.

Table 4-3 shows an assessment of the implicit and explicit approach considering these technical requirements. Advantages towards one or the other approach are marked by an emphasis of the corresponding table cell.

Technical Requirements	Implicit Approach	Explicit Approach
Preserve resulting shape	Capable	Capable
Provide model structure	Capable, comparably little effort	Capable, comparably high effort
Shorten data exchange time	Capable	Capable
Reduce data volume processed	Promising	Inferior
Provide CAD independent feature specification	Capable; UDF specification is method immanent	Set of feature types not open; UDF specification complicated
Maintain model systematics	Capable; shape manipulating features probably complicated	Capable
Support CA model exchange	Suitable, promises little compatibility problems at level of feature structure	Less suitable, promises heavy incompatibilities at shape model level
Enable selective model analysis	Promising, as structural information drives shape model	Nearly impossible, as structural information depends on complete shape model

Table 4-3 Comparison of explicit and explicit approach considering technical requirements

A comparison of the approaches shows that, in principle, both ideas are capable of realising a structure-oriented exchange. Both approaches are also capable of reducing the effort for manual rework after model import and thus of shortening data exchange time. The explicit approach has the advantage of an easy access to shape information, be it directly or indirectly generated. The implicit approach has major advantages regarding the resulting file size and further feature based exchange functionality.

The resulting data volume that needs to be transferred between engineering systems is one of the most important criteria. An implicit exchange method promises to decrease the data volume significantly. Compared to today's technology, an explicit data exchange will even increase data volumes. Regarding common complains from applying industry about unmanageably huge STEP files from large designs, this becomes a crucial disadvantage of the explicit approach.

Regarding the effort for establishing and standardising high-level feature based model representation mechanisms, the implicit approach promises higher efficiency.

Outside the scope of CAD model exchange this advantage is a prerequisite for the ability of integrating other CA systems. A realisation of selective data exchange is rather uncomplicated, whereas for an explicit exchange it seams almost impossible.

Summarising, with respect to technical requirements fundamental advantages can be identified for the implicit approach. It is therefore chosen as the solution and subject to detailed conception for a structure-oriented exchange method.

4.3 Development of a Structure-Oriented Model Representation

4.3.1 Elements of an Implicit Model Representation

Conceptual development of a structure-oriented model representation following the implicit approach starts with a detailed discussion of CAD model content. Necessary elements of a neutral representation schema can be derived from generalised CAD modelling and representation mechanisms, as sketched in section 4.2 (Figure 4-7):

1. A part structure contains main information at assembly level.
2. At part level, design elements of a CAD model have to be characterised: feature objects carry semantic and shape-related information; a set of feature types has to be specified as part of a feature library concept.
3. A representation must be defined for the structural information of the part model: the feature model structure.
4. A clarification is required about which shape model elements need to be transferred, and a concept is needed for their representation.
5. The representation of parameter values, constraints and constraint systems needs conceptual care. Specifically, compatibility of different constraint solving mechanisms requires special attention.
6. The ability of specifying user defined features (UDFs), i.e. application specific features designed by the user for later reuse, has an impact on the part model representation.
7. Finally, representation of free-formed shape and features that generate such shape require a separate discussion.

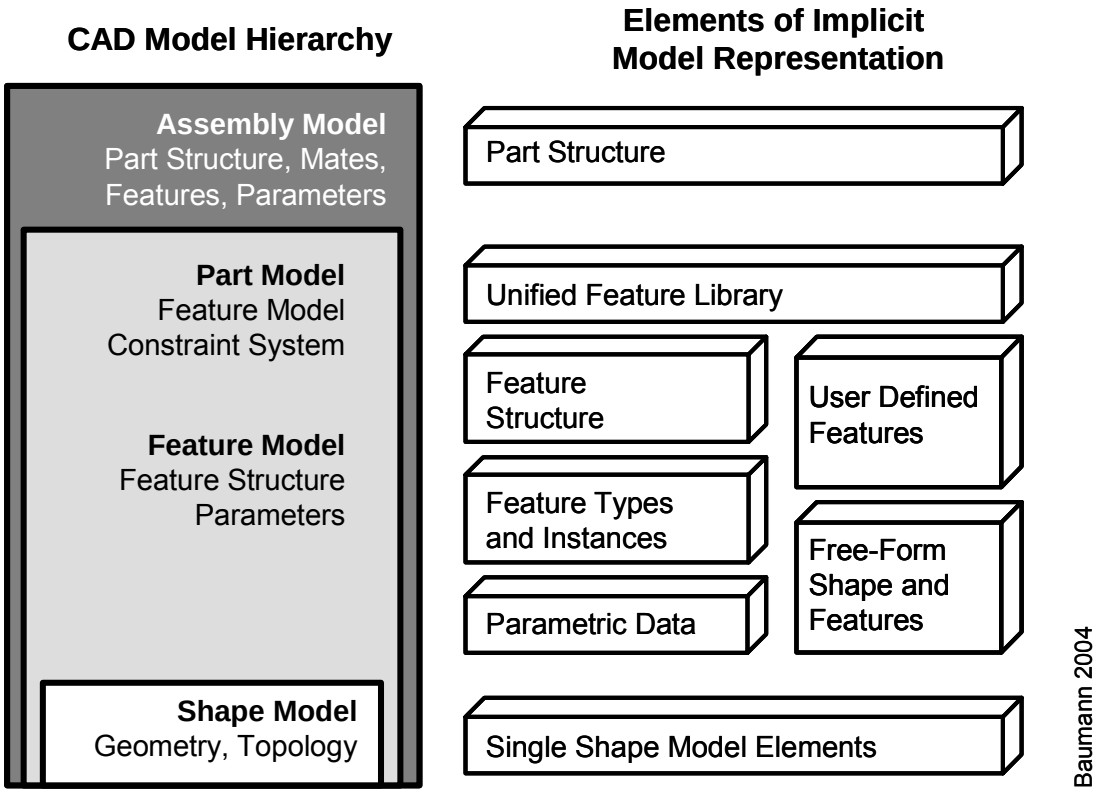


Figure 4-7 Central elements of an implicit CAD model representation

A representation of an implicit structure-oriented CAD model consists of an assembly model, a part model or a representation for single shape elements. A separate shape representation may optionally be part of an assembly or part model (Figure 4-8).

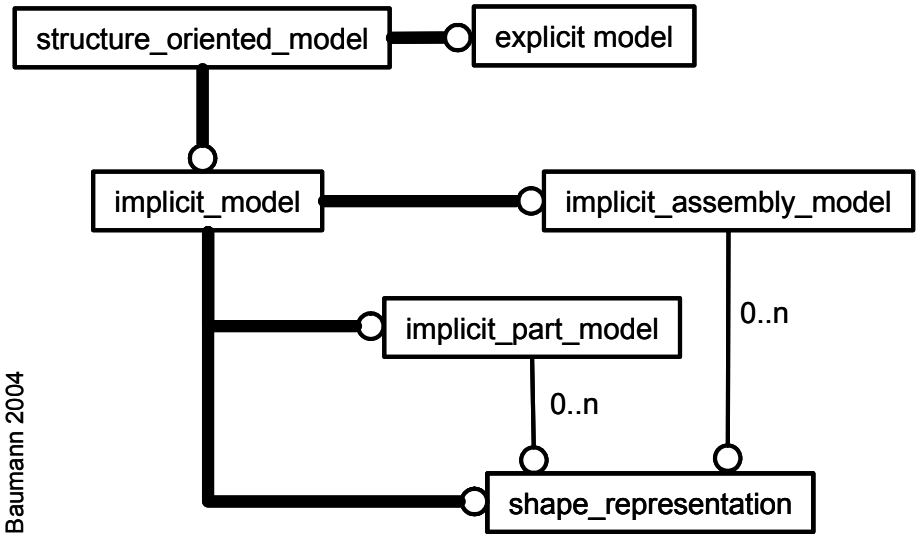


Figure 4-8 Structure-oriented model representation schema

4.3.2 Unified Feature Library

Within the conceptual discussion, a feature is considered the central design element in the sense of the generalised definition in VDI Guideline 2218 (see section 2.1.1), as it offers the most flexible understanding of feature technology and covers the capabilities of today's CA systems. CAD systems that are not feature-based have been excluded from the scope of application of this dissertation. The realisation of an exchange method for feature-based CAD models has to consider the different sets of feature types provided by various feature modellers.

In principle, all these feature types define the modelling capabilities of a CAD system and must therefore be considered by a standardised exchange format. As a prerequisite for shape model exchange, the STEP community has harmonised a set of shape model entities in part 42. Analogously, harmonisation of a set and structure of feature types as a basis for a structure-oriented exchange method is proposed, in the following referred to as a *unified feature library*.

The elements of this library have to be specified in a harmonising and system neutral manner in order to guarantee that feature objects can be instantiated with identical effects on the resulting shape models. As it cannot be purpose of an exchange method to transfer modelling capabilities between CAD systems, the unified feature library is limited to those features common to a broad variety of CAD system. Features that require modelling algorithms, which in this specialty is unique to few systems, may in fact be standardised but will probably not be supported by other systems.

A transfer of missing modelling functionality from one CAD system to another as part of the data exchange is admittedly imaginable. This would require the standardisation of a system interface, i.e. API functionality at shape modelling level and a transfer of high-level functions as a sequence of low-level API calls, e.g. as a binary library or as a interpretable algorithm coded in a formal language. Research work has been undertaken in this direction and has indeed shown first hints of feasibility [136]. Nevertheless, this approach is not subject to further conceptual discussion.

The most suitable of existing approaches to a unified feature library can be identified in STEP part 111. Standardisation within ISO 10303 ensures general applicability. Furthermore, these concepts are closest to the capabilities required for a structure-oriented exchange method. As a matter of efficient specification, features sets should be grouped in conformance classed according to their scope of application, e.g. basic design feature, mould and die design features, rotational and prismatic machining features, piping features, and further more. This enables the standardisation community to develop feature sub-libraries in parallel to the advancement of commercial CAD system's modelling capabilities.

Following RIEGER's classification, elements of a unified feature library can be grouped in shape generating, shape altering, shape neutral and shape independent features. Since the implicit approach generally does not exchange the actual shape information resulting from feature modelling operations, the instantiation of features has to produce deterministic effects on the shape model. This presupposes an appropriately meticulous specification of shape-related library features. At least the following properties have to be specified for every feature type:

- A graphical and literal description of the resulting shape produced at feature instantiation time;
- A type and name declaration;
- The origin of its local coordinate system relative to the world coordinate system;
- Its orientation in space;
- A set of attributes that determine dimensions, and other shape behaviour; and
- A Boolean operand stating whether the resulting shape can be added or subtracted from an existing solid.

Figure 4-9 shows an exemplary description of a form feature producing a rectangular block with given dimensions at a given position in space.

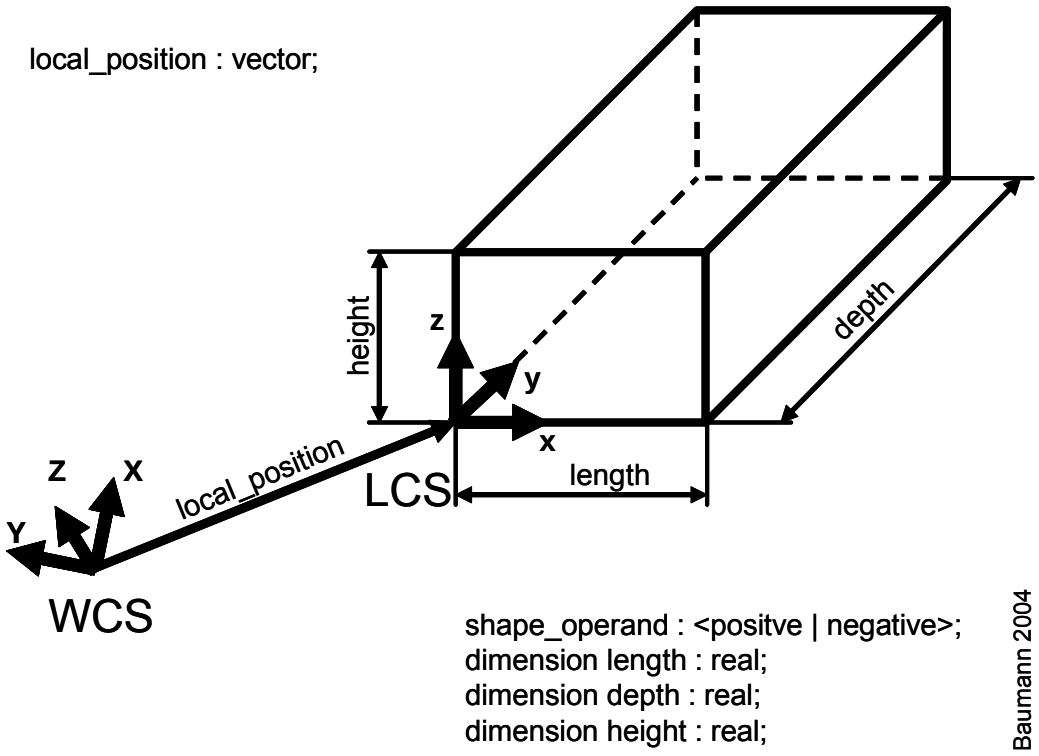


Figure 4-9 Form feature representation – specification of a block feature

Numerous approaches can be applied for the representation of shape generating form features creating shape primitives or pattern-like instances in multiple and given positions. Specifications of shape primitives and form-features found in STEP part 42, part 48, AP214, AP224, and within in CAM-I and Part-Library are to be harmonised and enhanced by mechanisms for representation of arbitrary parameters and constraints.

The semantic content of a feature can cover a large variety of information. Generally, the ability of features to recognise themselves, and the shape entities it created, as a modelling unit may be understood as the first level of semantic content. Any other information stored as semantic attributes of a feature usually is dependent on its design purpose and on the philosophy of the application module of the specific CAD system. In this way, semantic content and corresponding modelling functionality are somewhat special, if not unique, to the CAD system and therefore difficult to harmonise. On the other hand, the industrial need to exchange this particular information among a broad variety of CAD systems will assumedly be negligible.

At exchange time, feature instances have to be described in a way that allows the receiving system to regenerate a corresponding feature instance within its own modelling environment. Feature type, name, parameter names, values and value defining constraints, local position, are obligatory attributes. Feature spanning

constraints are represented within the part model. Preferably, a feature instance representation is independent from a specific type of feature and feature attribute. An example for a representation of a block feature instance is shown in Figure 4-10.

```
<featureNode isAlive="true" isSuppressed="false" name="BLOCK">
    <coorsys>
      <origin>
        <x>0</x>
        <y>0</y>
        <z>0</z>
      </origin>
    </coorsys>
    <expression>
      <pair name=“LENGTH" type=“dimension” value="100"/>
      <pair name=“DEPTH"  type=“dimension” value="100"/>
      <pair name=“HEIGHT" type=“dimension” value="100"/>
    </expression>
  </featureNode>
```

Figure 4-10 Example of a block feature instance in XML notation

4.3.3 Feature Model Structure Representation

Central challenge of an implicit exchange is the representation of the feature model structure in which features are only one type of element. Feature instances are combined to a hierarchical construction that constitutes the logical behaviour of a design. To find a representation schema for this feature structure, adequate elements have to be identified.

Considering the CAD system internal modelling mechanisms at part level, as generalised in section 2.1.1, a feature structure can be recognised as a binary tree. The elements of this tree are:

- The root node, representing the modelling result for the part;
- Inner nodes, representing a combination operation on the sub-branches or leaf nodes of the tree;
- Leaf nodes, i.e. feature objects.

This tree structure principally resides in every feature modeller, even though the appearance of its presentation to the user may differ. In some systems, the feature structure is reduced to a linear list, in which the inner combining nodes are not shown. The features are still combined in a Boolean manner.

The representation itself is equivalent. An example of a feature model tree structure is given in Figure 4-11.

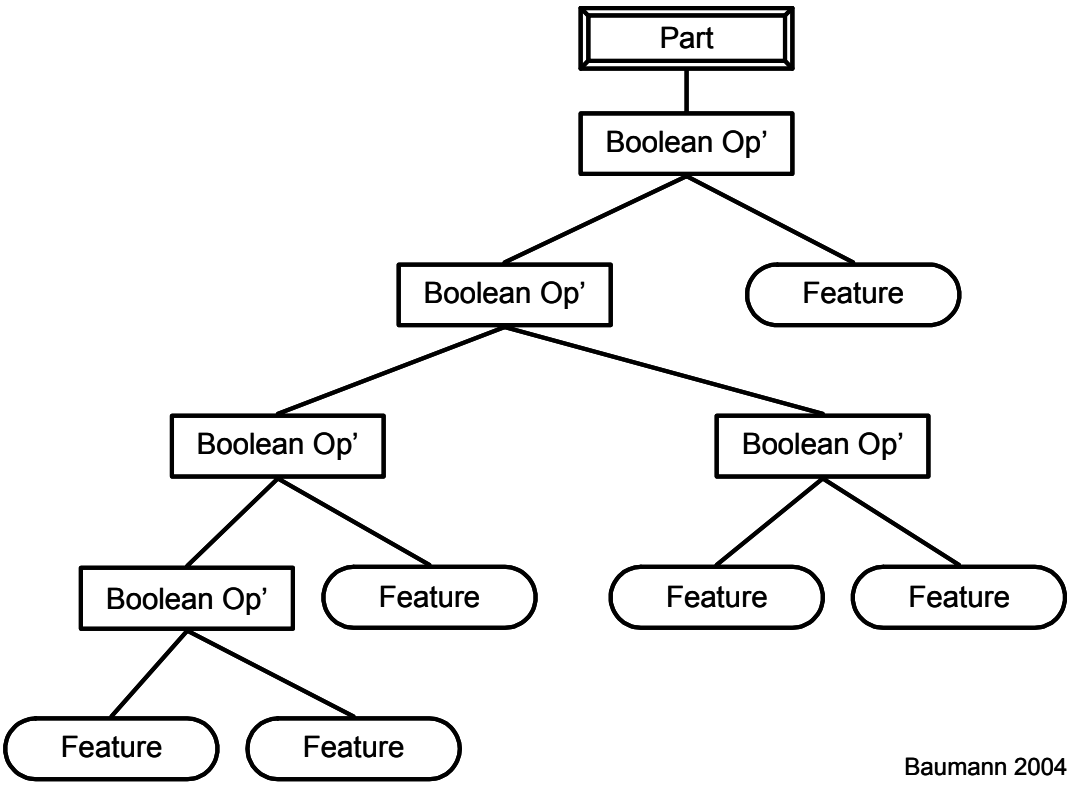


Figure 4-11 Example of a feature model tree

Feature nodes contain all information belonging to or associated with a feature instance as discussed in the previous section. The feature model automatically represents both structural and semantic information, which is an elegant consequence of the feature-based approach. Inner nodes combine these features by applying structural operations. Structural nodes may perform a

- Unite,
- Subtract,
- Intersect, or
- Modify

operation. The type of operation corresponds to the effect on the resulting shape model. New shape entities are added to or subtracted from the part. Else, the final shape results from a shape intersection or from the modification of the original shape by a shape manipulation feature applying, e.g. rounds, fillets or chamfers, to the part.

The structural operations are similar to CSG operations at shape modelling level, apart from the fact that they are applied to features and do not necessarily carry out a CSG operation. In the case of a shape neutral or shape independent feature, a unite operation has no effect on the resulting shape; it simply adds the feature to the tree. An illustration of the part model representation schema is shown in Figure 4-12.

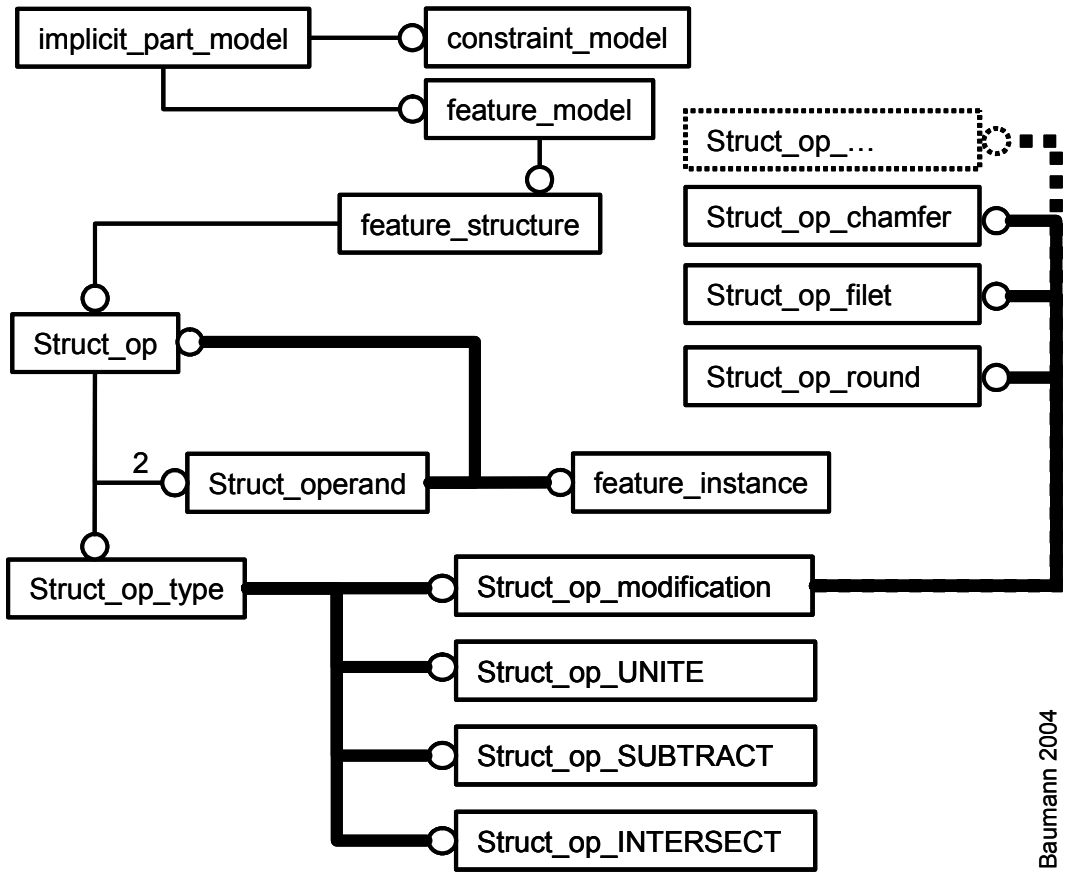


Figure 4-12 Example of a feature model tree

Some additional aspects of feature structure representation shall be mentioned. The position of a feature is part of the feature representation. Nevertheless, a part may have additional coordinate systems next to the global or world coordinate system. If a feature is located relatively to one of these origins, the corresponding association has to be represented within the feature. Definition of feature spanning constraints and constraint system solving is discussed below. Feature instantiation is performed by the import processor according to the numerical parameter values.

Another challenge emerges from the fact that some CAD systems do not totally support hybrid shape models in which b-rep and CSG functions are allowed; the Pro/Engineer system is one example. Pro/Engineer does not allow the existence of

more than one solid within a part model, which automatically prohibits CSG-like modelling. Instead, shape must be added via a protrusion or revolve operation based on a sketch that must reside on the part's surface. At feature level this leads to a one-sided tree.

A prerequisite for alterability of an imported part model is the adequate association of feature and shape model entities. Only if resulting shape entities can be identified from single feature objects, and only if the corresponding feature can be identified from a selected shape entity, the part model is fully operational and can be modified by the user. The central idea of the implicit approach is the regeneration of the shape model through the feature model. The complete shape model is generated via feature instantiation and combination operations. As these processes are invoked using native, i.e. internal, modelling mechanisms, it can be presumed that all necessary associations between feature and shape model entities are also generated. This mechanism is therefore inherent in the implicit approach.

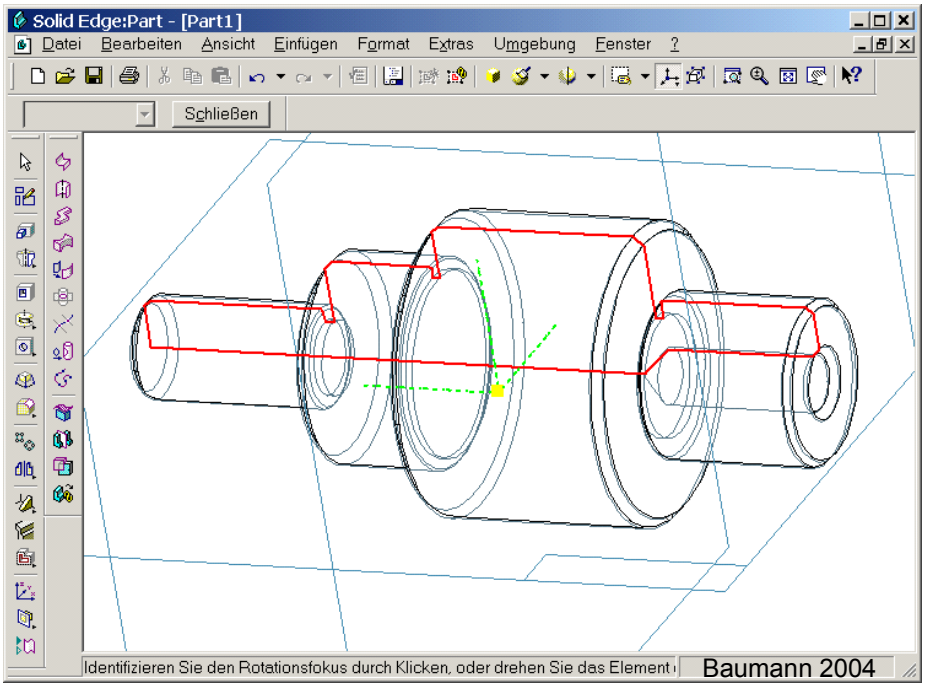
4.3.4 Shape Representation

Generally, the concept of a unified feature library specifying the resulting shape for every feature type guarantees that every CAD system implicitly regenerates an identical resulting b-rep model. As the structural elements of the feature model structure are also well defined, one of the two major requirements to a structure-oriented exchange, the conservation of the resulting shape, is fulfilled.

Two exceptions to the implicit shape regeneration have already been mentioned: shape elements that where either explicitly defined as a basis for shape generating features, or that are explicitly referenced as an input for shape manipulating features. In order to illustrate the actual challenge of explicitly represented shape elements within an implicit exchange method, these two cases shall be explained before possible solutions are discussed.

Various shape generating features require shape elements as an input information that the user has to model explicitly. Examples are protrusion and revolve operations, creating a solid from a loop of edges within a plane. A sweep feature generates a surface by "moving" a curve in space along another curve; both curves have to be defined explicitly by the user. Figure 4-13 shows the example of a part model designed with the SolidWorks system. A collar is created by a revolve feature with a revolve angle of 360 degrees and based on a planar sketch of a closed edge loop. The representation logic is illustrated in a procedural notation of the function call creating a revolve feature with given parameter values and sketched edges.

It is obvious that these explicitly defined shape elements require explicit representation, as they were not created by a feature. Fortunately, representation of 2D sketches and arbitrary curves in space is a solved issue. The STEP technology provides appropriate schemas, e.g. in part 42 that can be applied without modification. Concepts of part 108 will add the necessary schema for parameterisation and constraints.




```
edge_1 := edge (vertex_1, vertex_2);
…
edge_24 := edge (vertex_24, vertex_1);
sketch_info := closed_sketch (edge_1,…,edge_24);
rev_angle := 360;
axis := line (point1, point2);
feature collar := feature_revolve (rev_angle, axis, sketch_info);
```

Figure 4-13 Solid generation from explicitly defined shape and revolve feature

Different from shape generating features, those features that manipulate the existing shape model do not refer to shape elements the user explicitly defines. Instead, they refer to any set of existing shape entities as an input to the modelling operation they carry out. Blend and chamfer features are the most common examples. When choosing a blend feature to round one or several edges, the user manually picks these edges out of the part presented by the graphical interface. In terms of modelling commands, elements are selected from the graphical presentation of the shape model. System internal mechanisms derive the corresponding shape representation items and establish an association to these items as an input for the blend feature.

Apart from the fact that the persistence of these associations within system internal representation is still an open issue (see section 2.1.1), for model exchange purposes a mechanism is needed capable of uniquely identifying those shape elements that are explicitly referenced from shape manipulating features. The challenge is to find an identification mechanism for shape model entities that, due to the implicit character of the chosen approach to structure-oriented exchange, shall not be represented.

There are basically three approaches to this challenge:

1. Representation of feature shape elements;
2. Identification via persistent IDs
3. Identification via coordinate referencing.

When using PDGL as a means for the specification of feature type, RIEGER tackled the same questions. Using the ACIS modelling engine for feature instantiation within the FEAMOS system, he chose topological faces as the lowest level of feature shape representation. For shape manipulating features, lower-dimensional elements, i.e. edges and vertices, where calculated from intersections of these faces. Disregarding that representation of faces is not compatible with the idea of implicit model exchange, the mechanism is insufficient for other reasons: Not in all cases does the intersection of two-dimensional shape result in exactly one edge or vertex. For example, the intersection of a plane and a cylindrical face, with the middle axis of the cylinder being parallel to the plane face, may have none, one, or two resulting edges; in the latter case, the system might identify the wrong edge. There are further examples leading to the conclusion that calculation of lower-level topological entities from intersecting higher-dimensional ones is no sufficient approach [89].

As a solution to the persistent naming problem, i.e. the persistent identification of topological elements, naming convention mechanisms have been proposed (see section 2.3). These mechanisms address the scope of system internal model representation. They are based on the prerequisite of being executed as part of the system's internal modelling engine and thus having access to all representation items and structure. For model exchange the situation is different. A naming algorithm must be deterministic even outside and without having access to the system's management environment. Furthermore, a naming algorithm for implicit exchange must identify shape elements that are not actually represented.

Figure 4-14 illustrates the application of a persistent naming algorithm according to HOFFMANN / CHEN [130]. A round feature is applied to a block that has already been "named". The round feature refers to an edge identified as V3 according to the system's internal naming convention. For model import, a processing engine must be able to identify this edge without the corresponding shape representation. This presumes a standardised naming algorithm that allows ID mapping from and to system internal representations. Unfortunately, it is most presumably impossible to standardise persistent naming algorithms for CAD system because they strongly depend on the shape representation philosophy, which, in return, strongly varies with the internal shape modeller.

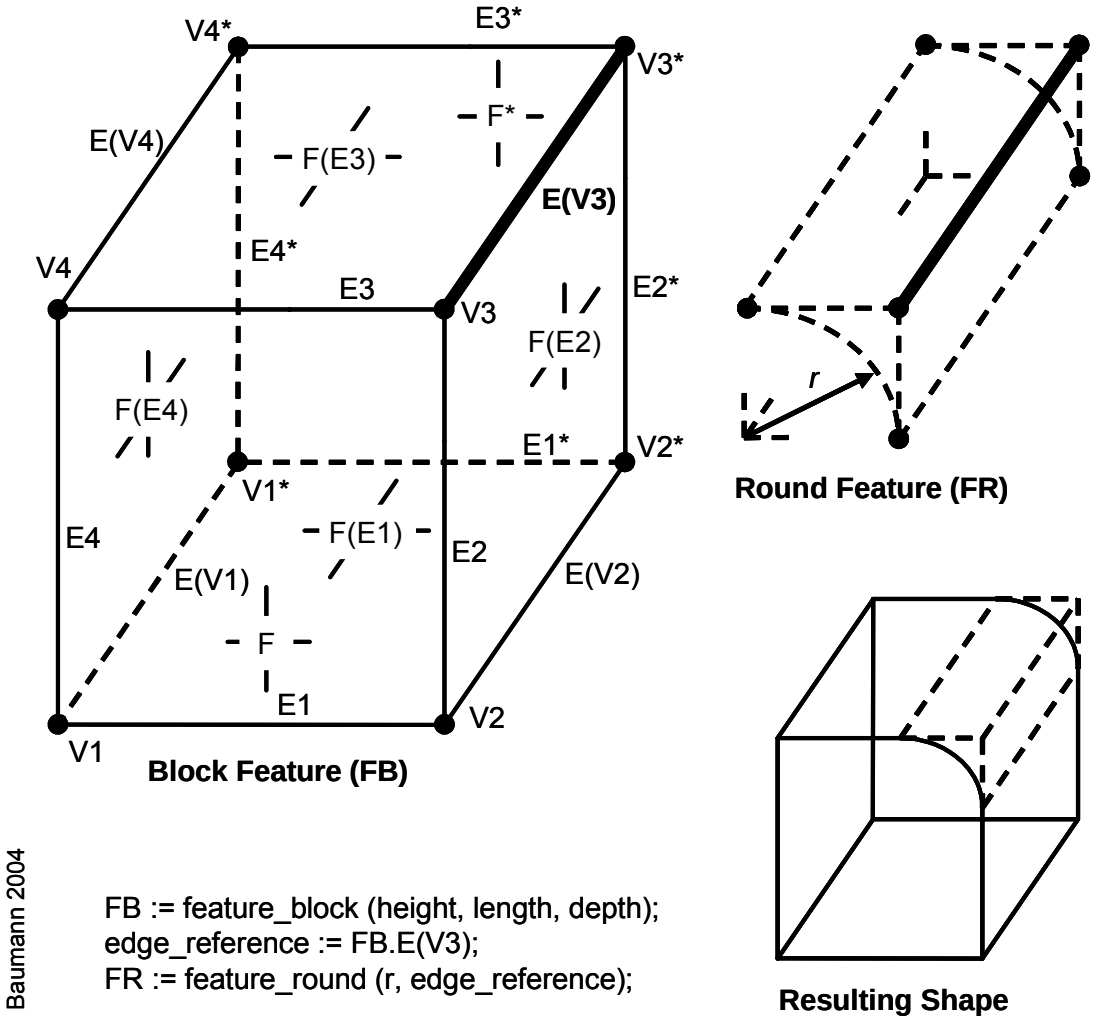


Figure 4-14 Persistent naming example for implicit model representation according to [130]

A third approach to identification of shape elements is coordinate referencing. Given its type and characteristic position, a topological item can be identified directly within a shape model. The characteristics depend on the type of element, e.g. a circle is determined by its centre point, the radius or diameter, and a second point residing on the circle; a straight edge is characterised by its start and end position. This information is sufficient for uniquely defining a topological element within an implicit model representation. Nevertheless, for the receiving system it may not be sufficient for identifying the element within its internal shape model. Due to differences in shape generation or numerical accuracy, the system may not be able to identify a specific type of topological entity at the given position.

Approaches to provide a certain fault-tolerant behaviour are two-fold: The processor may search the shape entity within an adjustable neighbourhood and may apply topology analysis functions. Entity search with adjustable neighbourhood intervals

has been applied successfully in topology healing algorithms for free-formed shape. Topology analysis can help in cases where more than one entity is found at the given position. For example, only outer edges constitute a valid input to a round feature.

This *intelligent coordinate referencing* mechanism will probably not be successful in every possible case. Further research will have to be conducted on fault-tolerance optimisation. On the other hand, an implementation is possible and appears to applicable from the user's perspective.

The following conclusions can be made:

1. In order to support revolve, protrusion and other operations based on explicitly defined shape, the corresponding shape entities have to be represented explicitly.
2. Support of shape manipulating features requires special effort for identification of shape elements that are not explicitly represented.
3. Shape identification based on persistent naming algorithms is not a promising approach, but subject to further research.
4. Instead, an intelligent coordinate referencing mechanism is proposed for shape identification.

4.3.5 User Defined Features Representation

Modern features-based CAD systems allow the user to define his own design elements according to his special requirements and design scope. A designed part may be stored and reused as a user defined feature (UDF). Different from pre-defined feature types for which a standardisation as a unified feature library has been proposed, UDFs must generally be considered as unknown to the receiving system.

Following the proposal of a unified feature library, user defined features could also be subject to standardisation. Various types of features for different design domains could be unified by a permanent working group within a standardisation organisation. The resulting library could be published on a regular basis. The idea is both ineffective and inefficient. On the one hand, system vendors could hardly be motivated to regularly implement additional import and export processor functionality. On the other hand, harmonisation of design domains and corresponding features would soon extend the harmonisation effort for pre-defined features.

If CAD model exchange shall not be limited to pre-defined features, a mechanism must be developed that notifies UDFs to the receiving system. Such a UDF notification mechanism has to represent and transfer the UDF specification that enables the receiving system to instantiate corresponding feature objects.

As described in section 2.1.2, the formal languages PDGL and TEBIS were created to specify features at type level. These languages can also suite as a means for a standardised description of user defined features. A UDF representation is created by a post-processor from the CAD system's internal UDF model. This specification noted in a PDGL dialect is transferred to the receiving system and interpreted by a pre-processor, which generates a system internal UDF representation. PDGL describes feature types at shape modelling level. The application of such a formal language requires harmonised shape modelling operations and representation mechanisms in order to be compatible with a wide range of CAD systems. In principle, a harmonisation can be based on the STEP shape modelling and representation philosophy.

A different idea for UDF notification aims at reproducing the actual procedure of feature definition. A user defined feature usually is a part consisting of pre-defined or available user defined features. The design procedure is no different from regular part design. A UDF consequently consists of a feature model, and it can be specified by a normal feature model structure. In other words, UDF representation is equal to part representation. The mechanism must be recursive as UDFs may be used for further UDF specifications.

Figure 4-15 shows the example of an electric connector modelled with Unigraphics. Variants of the connector shall be designed with different systems. The electric interface is standardised and shall be reused. In order to be unique for all subsequent connectors, the interface is declared a UDF by exporting the corresponding sub-tree of the feature structure.

A UDF notification, i.e. the transfer of a UDF specification to another CAD system, can either be content of a separate data stream or part of the model in which the UDF is applied.

STEP AP214 incorporates two mechanisms for features definition from a shape model: a subset of the shape model is declared a feature, which is similar to the idea of sub-tree definition, or a feature is declared in advance including its shape representation. AP214 provides these mechanisms for the support of shape patters. Analogously they are also meaningful for the application of user defined features, as UDFs are design patterns on feature level.

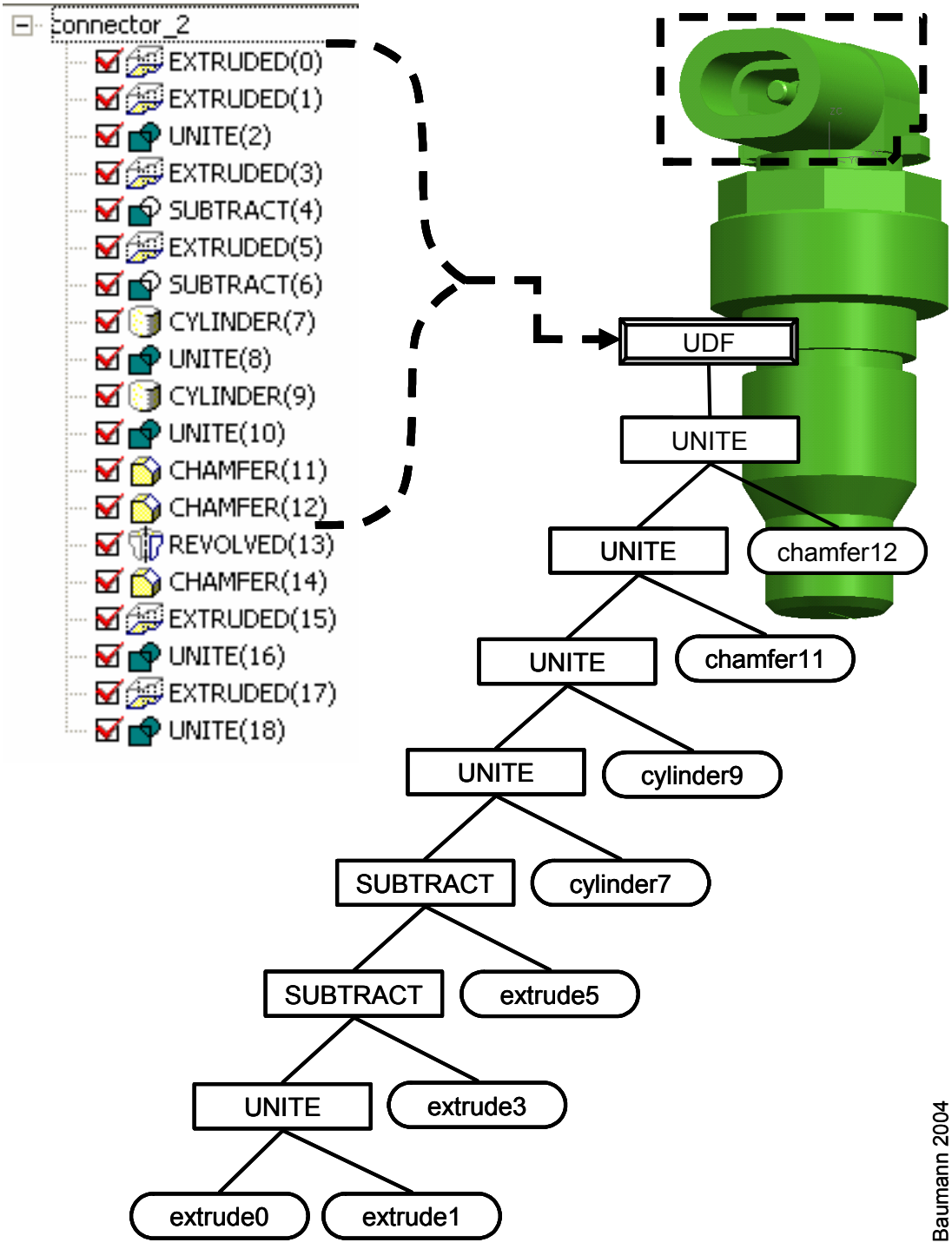


Figure 4-15 UDF specification via feature model sub-tree definition

4.3.6 Parametric Data Representation

Parameters and constraints are important elements of feature-based design. Objects at all levels of the CAD model, assemblies, parts features and shape entities, carry attributes that automatically or on the user's invocation are assigned values. Parameters can also result from the calculation of one or several constraints defined on one or several parameters. The totality of parameters, constraints or constraint system, and resulting values shall be referred to as the parametric model of a part or assembly model.

The parametric model is an important aspect of model alteration. The specific behaviour of a parametric model is strongly dependent on the constraint solving philosophy the CAD system incorporates, namely fully parametric or variational. This behaviour cannot be "transferred". Nevertheless, the parametric model is the modelling foundation and must be represented and transferred as part of a structure-oriented model exchange. Four challenges can be identified:

1. Representation of part and feature parameters and values;
2. Representation of constraints and constraint systems;
3. Constraint system solution representation and instantiation support; considering different constraint solving characteristics;
4. Representation of parametric shape elements and geometric constraints.

The representation of parts and features as parametric entities requires additional representation items. Obligatory, the following parameter information must be represented:

- Unique identifier, i.e. an identifier assigned by the system;
- Name, i.e. an identifier assigned by the user;
- Type, e.g. numeric, Boolean, string, etc.;
- Unit, distinguishing e.g. millimetres from inches;
- Value, i.e. the current result from the last (and consistent) calculation of the constraint system or a simple value assignment.

The unique identifier is specifically important for the representation of constraints. An additional name is part of the design semantic and supports human understanding. Type and unit declaration support constraint plausibility analysis. Alternatively, the system of units can be represented within the part model header. Basic parameter types have already been defined by the STEP methodology. A reference type of feature attributes is needed for the association of shape elements with a shape manipulating feature, e.g. a blend feature will contain a "shape entity list" of edges

and vertices to be blended. Also, revolve or protrusion features will refer to a "sketched curve" as an input their modelling operation.

A constraint defines a formal mathematical interrelationship of variables by formulating a computable term. A constraint system consists of a list of constraints. Variables in a part model are either feature parameters or freely defined variables. A system neutral representation of a parametric model consequently consists of a list of feature independent variables and the list of constraints at part model level. Grammar and syntax for constraint model representation can be defined in accordance with, e.g. STEP part 108 [88]. The mechanisms for referencing shape parameters within a constraint are principally equal to those needed for referencing feature parameters.

The main challenge in exchanging a parametric model is to guarantee that the transferred system of parameters and constraints can be managed by the receiving system. To "manage" in this connection means the ability to represent the imported constraint system and to calculate an appropriate solution. The solution is appropriate when it results in the original values of parameters and variables, and when feature instantiation induces the original resulting shape model.

At this point, the CAD system's parametric modelling philosophy is the crucial aspect. Main characteristics of fully parametric and variational constraint modelling and solving have been introduced as part of the state of the art summary in section 2.1.1. The possibility in variational modelling to define undirected constraints and to compute a numerical result from the complete constraint system is only partly compatible to fully parametric design. In principle, four cases have to be discussed for the exchange of parametric models:

Case 1 – fully parametric to fully parametric:
The constraint solving result is dependent on the order in which constraint are defined and solved. Consequently, model import, constraint representation and model import must retain the original order. Computed results can be compared with the original values to ensure appropriate exchange results.

Case 2 – variational to variational:
It shall be assumed that generation of over-defined constraint systems is inhibited by the variational constraint modeller. Well-defined constraint systems have exactly one solution. In order to ensure that this solution is found by the receiving system, representing the original solution is helpful in the case of different solving strategies. The most common situation is an under-defined constraint system, which has theoretically infinite, but in practice at least a large number of solutions. Here, the representation and instantiation of the original solution to the constraint system is obligatory. A variational constraint solver usually does not search for a new solution to a constraint system if the existing one is currently valid. Therefore, all parameters should be assigned the original values as a local solution to the constraint system. It can be assumed that the receiving system will normally accept this solution.

Case 3 – fully parametric to variational:
From the mathematical perspective, assignments are a trivial case of variational constraint modelling in which each mathematical relation is limited to exactly one variable on the left side; and the relation must be an assignment. It can be concluded that a fully parametric constraint system can generally be handled by a variational modelling engine. Again, transfer of the original parameter values ensures that the variational constraint solver finds the intended solution.

Case 4 – variational to fully parametric:
This is the reversion of the latter case: a set of general mathematical relations must be mapped onto parameter assignments. Science does not provide a closed strategy to this problem. Generally, variational constraint systems cannot be handled by fully parametric CAD systems. In order to prevent a total incompatibility, a workaround solution shall be proposed: The CAD model is regenerated on the basis of the original parameter values. Following, parameter assignments are regenerated, whereas undirected constraints are ignored. Along with these relations, substantial behavioural intelligence of the parametric model is lost. Nevertheless, the imported model is still fully operational, i.e. its resulting shape is conserved and the user can apply arbitrary modifications.

There may be situations in which a constraint solver insists on computing a new solution different to the given one. In addition, a variational constraint solver may not accept a given solution. These inconvenient circumstances are subject to system specific development and cannot be discussed in further detail in this dissertation.

From the perspective of fault tolerance, the original solution to the constraint system should be exchanged in any case. Even if not required for solution finding, the original values may suite for solution comparison. Furthermore, they allow the import processor to regenerate a valid part or assembly model even if restoration of the constraint system fails due to unpredictable reasons.

The described mechanisms for the management of parameters and constraint within feature-based part and assembly models can also be applied to numerical constraints in two-dimensional sketches, whereas geometric constraints require a different solution. Activities described by SHAH / BETTING [90] and the emerging STEP part 108 illustrate that representation of geometric constraints is being worked at. The results can be included into an implicit model representation schema.

The behaviour of shape modelling engines strongly varies among CAD systems. On the other hand, geometric constraints in 2D sketches usually suite as a modelling assistance rather than a necessary attribute of the resulting shape. On the instantiation of feature from these sketches, internal geometric constraints have no effect. Specifically, aspects of resulting shape conservation and model alterability are not interfered. It is therefore recommended to exclude geometric constraints from a structure oriented model representation.

4.3.7 Free-Form Shape Representation

Free-form modelling is an extraordinary discipline even for modern CAD systems, and not all systems provide a free-from modelling engine. Free-form feature representation should be based on a classification of corresponding design elements. According to the implicit approach to structure-oriented exchange, the representation of free-form features should minimise the amount of explicitly represented shape elements. Specialised structural elements for the integration of free-form features into the feature model are also to be discussed.

A conceptual treatment of free-form feature representation requires an appropriate classification of these features, as discussed by AURICH [92] and TÖNSHOFF [93]. A specific free-form feature taxonomy is proposed by STIEL [137] regarding the resulting shape as classifying characteristic. This approach is not optimal for feature representation. Rather, a classification according to free-form feature instantiation is more adequate (Figure 4-16).

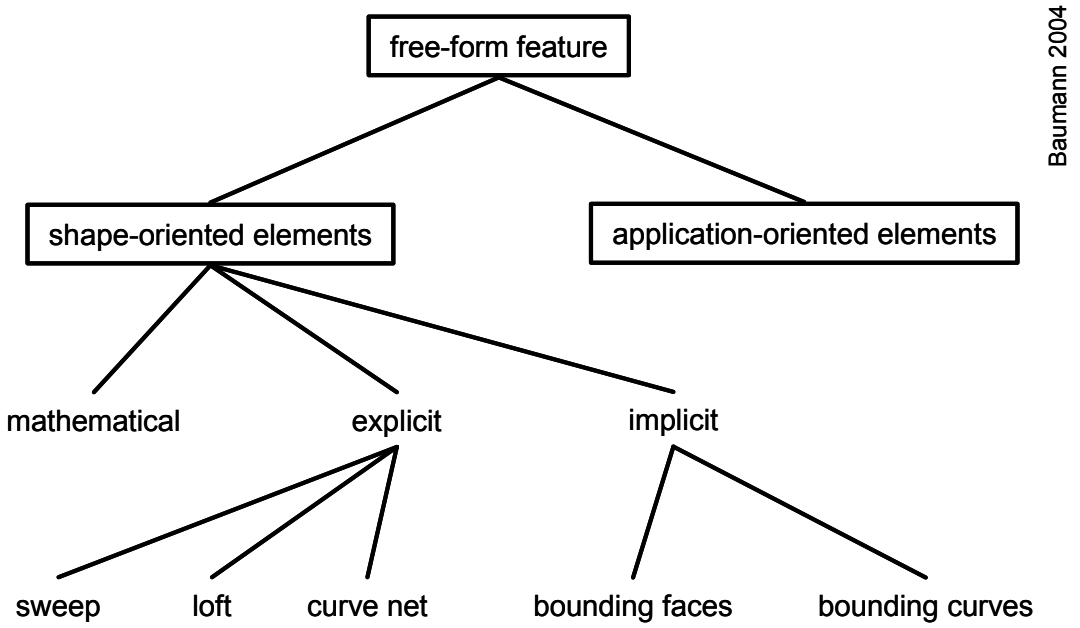


Figure 4-16 Taxonomy of free-form features according to instantiation principles

Presupposing a hybrid shape modelling engine, feature-based CAD systems support two methods for of free-form feature generation: shape-oriented or application-oriented. Shape-oriented free-form feature generation can be distinguished according to the type of input information:

- Mathematical methods generate free-from shape on the basis of mathematical definition, i.e. curves and faces are created from NURBS or Bézier polynomials; the necessary input is more or less directly retrieved from the user.
- Explicit methods generate free-formed curves and faces by applying higher-level modelling functions, such as sweep and loft functions. The necessary input information abstracts from the polynomial definition. The user directly describes the shape modelling result rather than its mathematical content.
- Implicit methods support an indirect generation of free-formed shape. The modelling result is generated from the definition of bounding curves, bounding conditions, e.g. degree of continuity, through blends on adjacent faces and further more.

Application-oriented free-form feature generation apply shape modelling algorithms specific to a certain design domain. These algorithms create shape elements from given technological constraints. The user input defines the modelling result in terms of engineering conditions. For example, a free-formed face is created due to criteria like minimised surface area, optimised mass distribution, minimised air drag coefficient, or a given optical or acoustical behaviour.

An ideal solution to structure-oriented exchange would support all these methods. It must be noticed, however, that mathematical, implicit and application-oriented methods to free-form generation are a specialty to few CAD systems. Unification of corresponding features will assumedly not be supported by most system vendors.

The great majority of free-form shape generating features are of the class explicitly shape-oriented. From the algorithmic point of view, these features behave similarly to a revolve or protrusion feature: explicit shape is handled to create the intended free-formed shape. Figure 4-17 shows the example of a variational sweep feature that creates a solid from three given planar sketches as illustrated in a procedural notation. Analogously to protrusions features, the three closed loops have been explicitly defined by the user and must be explicitly represented.

To avoid cases, in which a free-form generating feature is not standardised, and thus cannot be imported, an explicit representation of the resulting free-form shape can be contemplated. Of cause, this is not a preferable solution and it quasi foils the idea of implicit data exchange, but it still provides an option to prevent the complete loss of information due to incompatibility.

The mathematical foundation of free-form shape representation are parameterised polynomials like Bézier, Lagrange, B-splines and NURBS. NURBS have the advantage of representing both rational and non-rational faces and are therefore considered the most universal representation [138]. The resulting free-formed curve

or surface explicitly represented by a NURBS will consequently consist of the necessary shape parameters, like sampling nodes, weights, degree of the polynomials etc. A corresponding representation schema has been developed by Stiel [137] and can be adopted.

Like all other design elements, free-from features are combined to existing shape by structural nodes within the feature structure. In the case the resulting shape is a solid, the usual Boolean operations can be applied (see section 4.3.3). For the combination of solids and free-formed faces, additional node types are required that allow trimming of solids and provide the possibility to use these curves and faces as an input to protrusion and revolve features.

The application of free-form features within the scope of implicit structure-oriented exchange generally aims at, directly or indirectly, generating a solid. In the area of pure surface modelling, several other questions arise, which shall not be addressed in this dissertation.

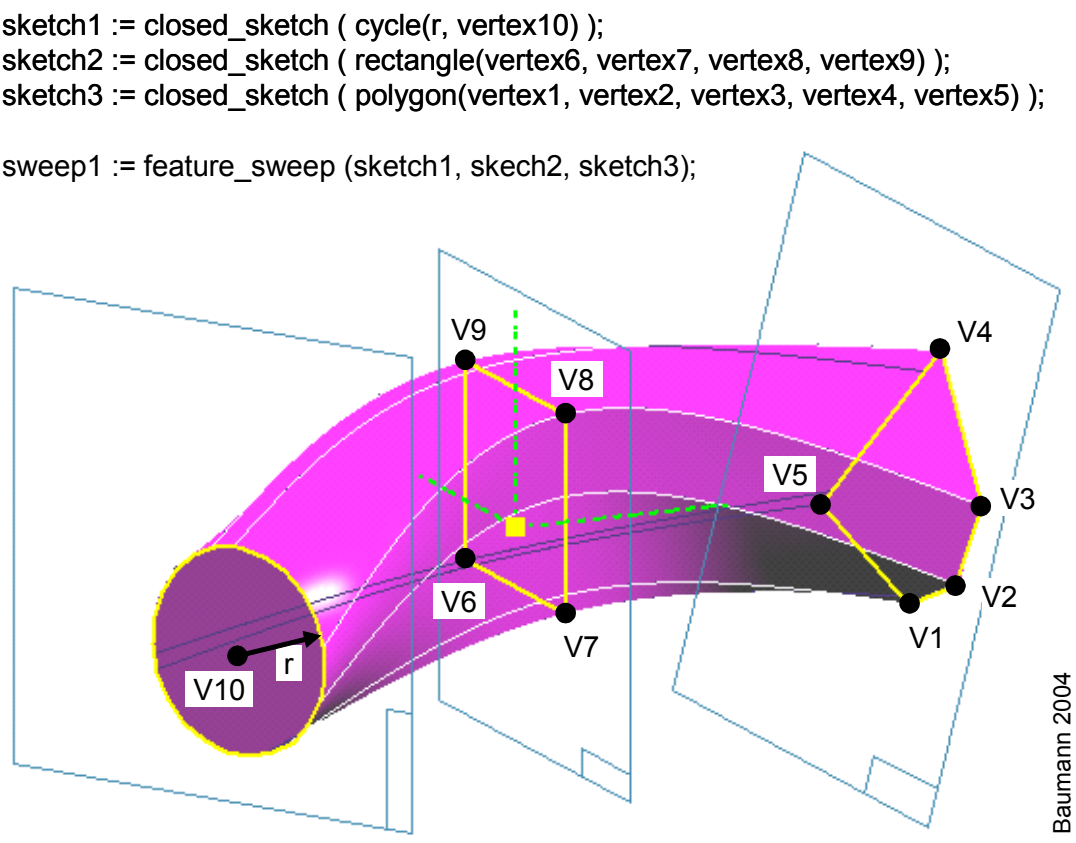


Figure 4-17 Variational sweep from explicitly represented sketched curve loops

4.3.8 Assembly Models Representation

Part and assembly modelling is usually performed by the aid of different systems or system modules. Consequently, representation of part and assembly models should be specified by different schemas. The STEP assembly representation schema is one example. An implicit assembly representation has to

- Identify the elements of an assembly model;
- Define the structural interrelationships within the assembly part structure; and
- Express positioning and parametric information.

A generalised view on assembly models has been proposed in section 2.1.1. An assembly consists of a hierarchical tree of sub-assemblies, with parts as leaf nodes. For some assembly modellers the definition of a master part is mandatory. A master part is the first part onto which other parts are assembled. The definition of a master part constitutes no loss to generality. Modellers that do not demand a master part may ignore the corresponding attribute. In the opposite case, the first part entry of an assembly representation can be interpreted as a master part.

The occurrence of independent features within an assembly model is the universal case, which is not supported by all assembly modellers. An example for an independent feature interfering with two assembled parts is a tolerated bore hole through the hubs of two flanged shafts. The occurrence of features in assemblies does not require a specific representation schema.

Parts and features are parameterised objects. In analogy to a parametric part model, additional variables and constraints may be defined in order to determine parameter values according to the assembly context. Generally, parametric modelling at assembly level does not differ from parametric modelling at part or feature level. The proposed concepts can be adopted.

Interrelationships between assembly model elements are different from those between features in a part model: Parts are assembled without intersection of their physical borders. Therefore, the required structural combinations basically aggregate parts and assemblies to higher-level objects. The structural node “is assembled to” can be applied at all hierarchy levels of the assembly. For independent features, a structural node is needed that expresses the application of a shape generation, e.g. SUBTRACT in case of a bore hole, or the association of a shape neutral or shape independent feature to the assembly, e.g. a neutral COMBINE node in case of a semantic feature.

Two different positioning mechanisms for assembly elements are common:

1. Positioning via absolute or relative coordinate referencing,
2. Positioning via mating conditions.

Positioning via coordinate referencing, also referred to as discrete positioning, is the method usually applied by standard exchange formats, such as IGES or STEP. A part is positioned relative to a world coordinate system or to a master part. Corresponding representation schemas can be adopted for the implicit exchange.

Problems with numerical tolerances, which may lead to mispositions, are occasionally discussed in literature. This is a general effect that cannot be "cured" by the implicit exchange approach. Fortunately, different from the case of shape manipulating features, the exact position of a part within an assembly is not crucial to a successful model import. Deviation in part positions usually occur at decimal places and may result in minimal overlaps of adjacent part. These collisions can be identified and eliminated by the user. The major objections of the structure-oriented exchange, shape conservation and model alterability, are not compromised.

Positioning via mating conditions is normally provided as add-on functionality to discrete positioning. Mating conditions specify the relative position of shape elements, i.e. collinear axes, coplanar faces, coincident vertices, and further more. For example, a bolt may be assembled into a flange by defining its middle axis as collinear with the middle axis of the bore hole, and the bottom face of the bolt head as facing and touching the face of the flange. More convenient modelling capabilities enable the user to define assembly characteristics for each part. When assembled, these parts automatically orientate themselves according to these pre-defined mating conditions [139]. This functionality requires special structural elements for the assembly part structure.

An applicable approach to elegantly representing adjacency interrelationships of mating parts is realised in the FEAMOS system: Mating conditions are represented in shape neutral adjacency features. Mating shape elements are referenced by corresponding feature attributes, which constitutes another case of explicit shape representation, as already discussed in section 4.3.4. The proposed solution, i.e. identification of shape elements by type and coordinate referencing, is also applicable for identification of mating shape entities.

4.3.9 Miscellaneous Model Content

The above described elements of part and assembly models contain the essential information of a CAD model. Their representation is essential for a structure-oriented exchange method. There are further information objects of lesser importance or originally defined as out of the scope of this dissertation, but nevertheless worth mentioning.

A CAD model carries organisational information, such as date of creation and last change date and time, model version, system of units, type of constraint system, and other meta-data, that may e.g. be used for product data management purposes. Other information has already been addressed, e.g. coordinate systems, auxiliary geometric elements, like symmetry axes, geometric points, and other more. Established

exchange methods have already found a suitable representation for these objects; thus, they do not have to be discussed in detail here.

Modern CAD systems provide many more part and assembly modelling capabilities that have not been addressed yet, but would still need a representation concept. Variant and complex part modelling, kinematic mechanisms, tolerances and semantic annotations of any kind, bill of material generation, and 2D technical drawings, are examples, which must be left open to future research.

Apart from model exchange among CAD systems, the so-called vertical exchange along the process chain of product creation comprises several consumers of CAD model data. Along this process chain, an implicit CAD model provides easy and even selective access to design data and may suite as a container for data aggregation. For example, manufacturing operations are based on a mapping of design features to manufacturing features, as addressed by BAUM [140] and ULBRICH [141]. These algorithms can selectively extract the feature information from an implicit CAD model. Time consuming regeneration of the complete model, and of the shape model in particular, becomes obsolete.

Knowledge-based engineering (KBE) methods aim at providing context specific information to the designer. KBE algorithms often analyse CAD model content as an input to their functionality. In many cases, the structural information of the CAD model is essential, whereas the resulting shape is of minor interest. The realisation of such KBE algorithms can therefore be enhanced by the application of structure-oriented exchange. For example, an algorithm for a knowledge-based generation of welding sequences is based on the welding feature information within the CAD model and on the positions of these features in the design part [142]. This information can efficiently be extracted from an implicit model. Complete model regeneration and analysis, including feature recognition algorithms, as would be required for STEP models, can be omitted.

4.4 Definition of Representative Exchange Scenarios

After discussing conceptual approaches to an implicit representation of feature-based product model data, this section will address the area of model transfer between different CAD systems. At first, relevant exchange scenarios shall be identified, which define the functionality of a transfer model strategy.

In chapter 1 the situation of engineering companies on today's markets has been reflected that have led to three product creation strategies: reduction of development time, increasing cooperation and efficient IT strategies. Summarising the introductory illustration of these strategies, three main scenarios for product data exchange can be identified with varying, but altogether representative exchange conditions (Figure 4-18):

A) Product data exchange based on pure file transfer;

B) Product data exchange in a world-wide web environment;

C) Product data exchange in a direct system integration environment.

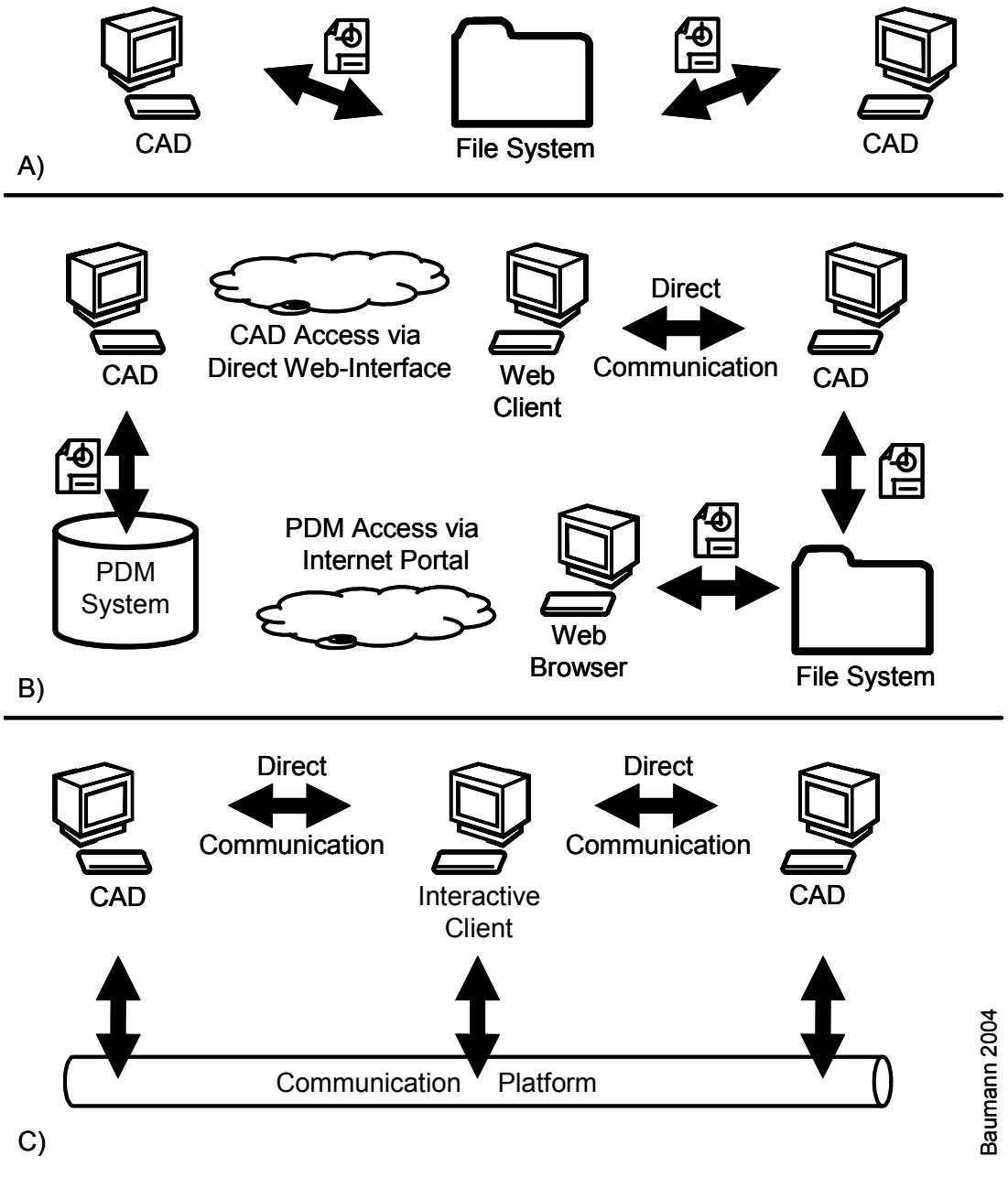


Figure 4-18 Three representative scenarios for product data exchange

The established and most common practice for the transfer of product models among different CAD systems is the exchange of data files. Different from the information flow to and from a data base management system, data files are comparatively small in size and easy to store. Required functionality for data file transfer is ubiquitous and inexpensive, which makes file-based data exchange the first choice for small and medium sized companies. The transfer of an implicit model file does not differ from the transfer of, e.g. a STEP physical file: the export data stream is stored to the local hardware, transferred by any means of file transportation, copied to the receiving hardware system, and finally imported into the receiving CAD system.

World-wide web or simply Web technology is increasingly applied in automotive and other industries as a means for supplier integration. Within a virtual private network, access is granted to data storages utilising a Web user interface, i.e. client-based, or through a Web portal, i.e. sever-based. The actual product data may reside on a product data management system or any other Web-enabled storage. A structure-oriented model may be transferred either via a direct Web interface or via a Web-enabled storage facility, e.g. a PDM system. A Web client establishes direct communication to a distant CAD system and may directly communicate with an in-house CAD system. A PDM system provides Web access through a portal that may be interfaced with any Web browser. Data is transferred as a byte stream and locally stored as a file before it is imported into the receiving CAD system.

For the establishment of integrated process chains, a direct system interaction is often necessary or at least desirable. Tools for calculation, simulation and other engineering tasks profit from direct communication to the CAD system. The computer-aided conceptual design tool FOD (Function-Oriented Design), which has emerged from the iViP project [3], [143] can be named as one example. The FOD system generates a parametric and constrained component structure as part of a product concept. This part structure is transferred to the CAD system via a direct CAD interface. The transferred data is little different from the content of an implicit part model. System integration is achieved by direct interfacing or via a communicational platform. An interactive client suits as a user interface for data transfer and may provide additional services, such as visualisation of model content, selection of single model elements or compliance analysis.

4.5 Development of a Model Transfer Strategy

4.5.1 Model Transfer Requirements and Functionality

With regard to the representative exchange scenarios classified above, a model transfer strategy shall be developed in the following sections. Detailed conception of necessary components is preceded by a discussion of technical requirements and general transfer functionality that fulfils all three scenarios.

Requirements on the new exchange method have been set up from a business perspective in section 3.3 (see Table 3-1 on page 31). Model transfer related aspects can be translated into technical requirements as listed in Table 4-4.

Business requirements	**Technical requirements**
Support data management: Support small and large scale data distribution and management strategies	Provide flexible data format: Arrange data flow to be applicable for established interfacing technologies
Support collaboration: Support modern Web-based exchange mechanisms	Support Web-based exchange: Ensure compatibility with Web data formats and transfer protocols
Support CA systems interaction: Enable / enhance direct information exchange at systems run-time	Support direct system communication: Ensure data format to enable both off-line and on-line communication and data flow
Provide fault tolerance: Ignore or compensate information incompatibilities even for different CAD architectures	Provide parametric compatibility: Ensure parametric model import regardless solving strategy
	Support fragmentary model regeneration: Regenerate model despite incomplete or incompatible model entities (as far as possible)
Provide applicable strategy: Ensure feasibility for an industrial application	Ensure commercial implementation: Ensure realisation of interfaces, processors and infrastructure with commercial CAD systems at feasible implementation effort

Table 4-4 Requirements on a new product model transfer strategy from business and technical perspective

Considering these requirements, basic model transfer functionality for the three exchange scenarios can be derived as arranged in Table 4-5. Principle components and information flow for model export is illustrated in Figure 4-19. A processing engine retrieves the native CAD model from the system's API. The native model is post-processed into an implicit run-time representation and transcribed into a data stream, which can be stored in a physical file or communicated by means of Web- or other technology. An external client may analyse and present the implicit model, i.e. provide a graphical view, and support user interaction for, e.g. sub-model definition. For model import the corresponding information flow is analogue. The functional disjunction of model processing and transcription allows the installation of multiple transcription engines generating different data formats. The selection of an appropriate format and conceptual aspects of CAD interfacing, export and import processing and transfer communication shall be discussed in the following sections.

A) File-based	B) Web-enabled	C) System integrating	Function	Description
X	X	X	Model post-processing	Conversion of system internal model into system neutral implicit representation
X	X	X	Model pre-processing	Regeneration of a native CAD model from a system neutral implicit representation
	X	X	Implicit model presentation	Provide view on implicit model content to human user
	X	X	Implicit sub-model definition	Enable the user to select subsets of the implicit model
	X	X	Implicit model consistency analysis	Perform consistency check on implicit model structure
X	X	X	Export transcription	Translation of the implicit representation into a transferable data stream
X	X	X	Import transcription	Translation of transferred data stream into implicit representation
X	X	X	CAD direct interfacing	Establishment of communication to CAD system according to the system's interfacing capabilities
	X		Web-based interfacing	Establishment of communication to engineering applications utilising standard Web technologies
		X	Platform-based interfacing	Establish communication according to platform specifications
		X	Platform-based communication	Provide means for unified communication of engineering applications
X	X	X	File storage interfacing	Storage and recovery of byte stream to and from a physical file

Table 4-5 Basic model transfer functionality

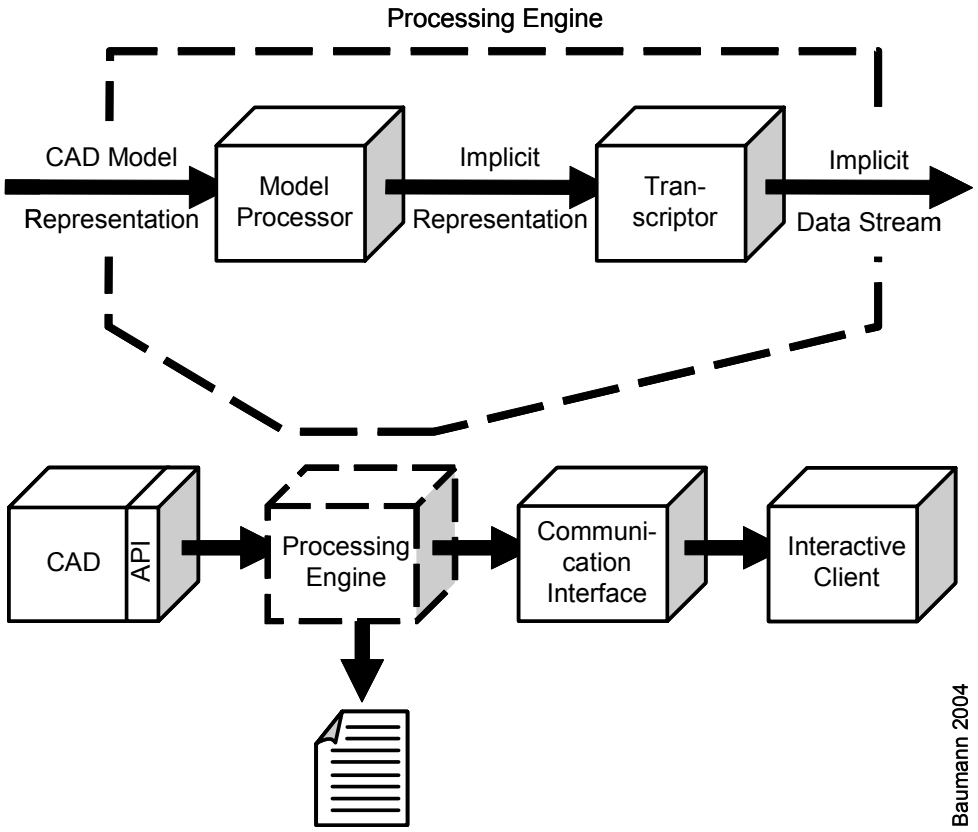


Figure 4-19 Principle components of a model transfer strategy and information flow for model export

4.5.2 Exchange Format

The purpose of a physical format is to formalise a data stream into a well-defined sequence of data items, each of them specified in type and structure. A physical format provides the envelope for the transfer of the implicitly represented model data. It is duty of the transcription function to transform the implicit representation into a byte stream compliant to the physical format. The conceptual discussion of all other components is dependent on the specification of the data format.

Some approaches and existing solutions for physical data formats have been discussed in section 2.2. Of those technologies, two are preferable for the transfer of implicitly represented CAD model data.

The STEP part 21 physical file specification was designed to transfer EXPRESS data models as text files. EXPRESS entities are arranged in an enumerated list; the entries themselves are symbolised independent from a specific representation schema. Interrelationships between entities are established by references on their number position. This mechanism can also be used to established tree-like structures of the

implicit representation. Resulting files are compact and communicable as files online data stream. Within Web environments, STEP physical files are not directly supported. The data stream can be transferred as an anonymous text stream, a content specific treatment is not provided. Contained data has to be converted to HTML or XML in order to be presentable by a Web browser. Due to the limited capability of representing parametric model content, specific representation constructs will have to be added. A fragmentary or aggregating transfer of model content is not supported. For application interface development, a limited number of software tools is available. The application of STEP physical files as a transfer format for implicit models requires an implicit model representation schema in EXPRESS or EXPRESS-G.

The XML technology has been developed to be applied within the scope of world-wide Web. Mechanisms as configured view provision in a Web browser are technology immanent. In this area, XML can be considered the best solution for data transfer. Due to its hierarchical entity structure, the XML format automatically corresponds to the hierarchical CAD model structure. The format can also support selective model representation. Independent schema definition and schema mapping capabilities, e.g. by utilising XSLT mapping descriptions, guarantee for an uncomplicated transcription into native application formats. An XML schema or DTD for implicit model representation has to be defined. The effort is similar to a schema specification in EXPRESS-G. XML schemas are specified in XML, which provides certain flexibility for the extension of the unified feature library. Feature types can be changed or added without implementation effort for an adaptation of the transcription engines. Availability of a large variety of tools and development environments optimally facilitates the implementation of XML-based solutions. The PDT-net project has proven that the XML technology is sufficiently applicable for data transfer between engineering and CA systems in an industrial environment. Communication can be based on Web protocols like SOAP. [126]

Comparing the characteristics of STEP physical files and XML technology, the XML approach appears more flexible, and less laborious in realisation. It shall therefore be chosen as the technical basis for the transfer of implicitly represented product model data.

Adapted to an XML document type definition, an implicit part model representation is shown in Figure 4-20.

```
<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT  part (unit?, featureNode*)>
<!ATTLIST  part path CDATA #IMPLIED>
<!ELEMENT  unit EMPTY>
<!ATTLIST  unit value CDATA #REQUIRED>
<!ELEMENT  featureNode
           (origin?, direction?, parameter*, curveData*)>
<!ATTLIST  featureNode name CDATA #REQUIRED
           ttype (BLEND | BLOCK | CHAMFER | CYLINDER |
                  EXTRUDE | REVOLVE | SPHERE | SUBTRACT |
                  UNITE | INTERSECT | CONE) #REQUIRED
           isAlive CDATA #REQUIRED
           isSuppressed CDATA #REQUIRED>
<!ELEMENT  parameter EMPTY>
<!ATTLIST  parameter
           name CDATA #REQUIRED
           value CDATA #REQUIRED>
<!ELEMENT  curveData (form, istOffen, firstVertex,
           secondVertex, center, radius, point1, point2,
           point3, nornalVec)>
<!ELEMENT  firstVertex ( x, y, z)>
<!ELEMENT  secondVertex (x, y, z)>
<!ELEMENT  poin1 (x, y, z)>
<!ELEMENT  point2 (x, y, z)>
<!ELEMENT  point3 (x, y, z)>
<!ELEMENT  center (x, y, z)>
<!ELEMENT  form (#PCDATA)>
<!ELEMENT  ist_open (#PCDATA)>
<!ELEMENT  direction (x, y, z)>
<!ELEMENT  origin (x, y, z)>
<!ELEMENT  radius (#PCDATA)>
<!ELEMENT  nornalVec (x, y, z)>
<!ELEMENT x (#PCDATA)>
<!ELEMENT y (#PCDATA)>
<!ELEMENT z (#PCDATA)>
```

Figure 4-20 XML DTD for implicit part model representation

4.5.3 CAD Interfacing

Realisation of standardised data exchange is best supported when based on standardised system interfaces. This perception has lead to several activities of standardising CAD programmable interfaces, as summarised in section 2.2. Unfortunately, none of the available results provides the necessary capabilities of handling feature-based models and associated information. For that reason, a

standardised CAD API with special functionality for structure-oriented model exchange is proposed, referred to as the *implicit CAD API*.

Necessary capabilities of an implicit CAD API can be derived directly from the implicit model representation concept. All model information essential for the generation of an implicit model representation must become retrievable from the interface. In return, functionality is needed that enables the instantiation of a native CAD model from the implicit representation. Main API characteristics address:

- Unified feature library support: all feature types must be applicable;
- Establishment of inter-system communication;
- Administrative data exchange: retrieve of system type and release; retrieve, load and store available models, etc.
- Assembly and part model analysis and retrieval: navigation within the model structure; request feature types, parameter values, parametric model information; retrieve associated model elements, i.e. get shape model from feature and get features from shape elements; request for auxiliary geometry and inactive shape elements, e.g. meanwhile deleted or replaced entities, split edges and faces; etc.
- Assembly and part model generation: instantiation of feature objects; variables and constraint definition; parameter value assignment; invocation of constraint solving.
- Shape modelling: generation of sketches and auxiliary geometry elements.

Communication mechanisms have to be specified in an abstract form. According to the OMG MDA paradigm, specific communication technology bindings can be derived regarding CAD specific capabilities [128]. Realisation of inter-system communication itself is no functionality specific to structure-oriented data exchange. By choosing XML as the exchange format, prerequisites are fulfilled to support all three exchange scenarios. Communication may be based on a specific middleware, such as CORBA, RMI, or Web-services, or on a higher-level platform concept, such as J2EE or the iViP integration platform.

The application of an implicit CAD API enables the realisation of a standardised processing engine for a structure-oriented model exchange (Figure 4-21).

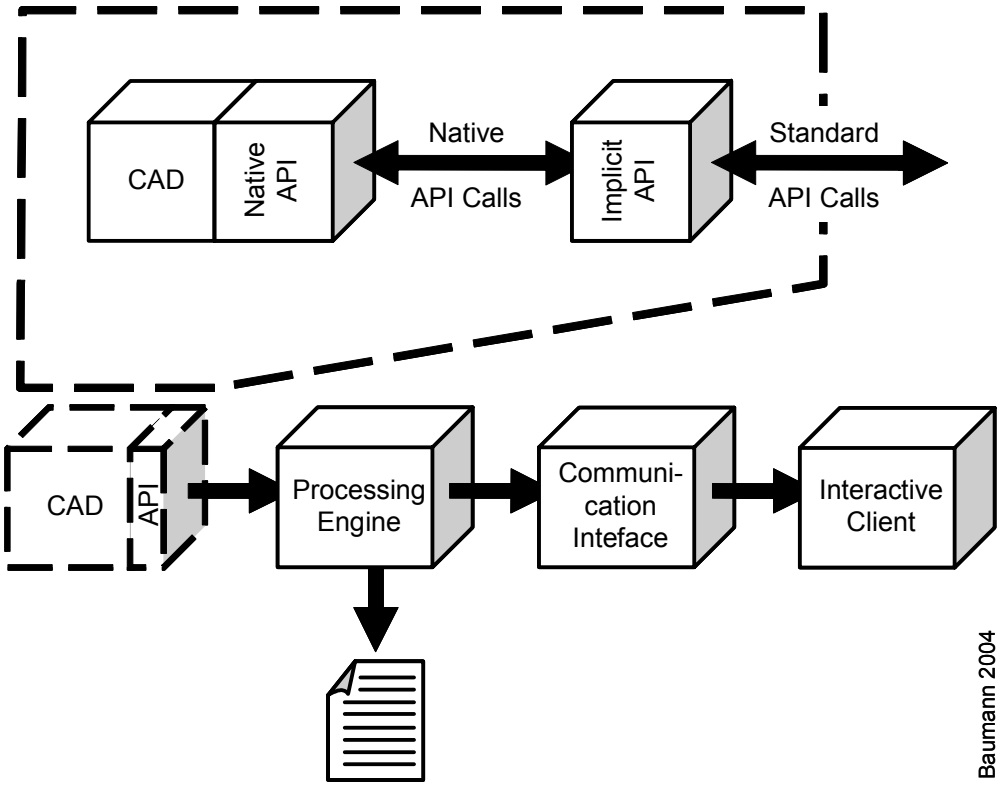


Figure 4-21 Abstraction of native modelling capabilities through an implicit CAD API

Apart from a generalised communication establishment and retrieval of available models at run-time, standardisation of CAD interfaces at ISO and OMG have not developed as far as required to support implicit model exchange functionality. Nevertheless, the proposed functionality can be adopted. As a tribute to compatibility to the OMG CAD-Services specification version 1.1, two new methods for the class CadSystem are proposed, covering the complete implicit model import and export (Figure 4-22).

```
public String export_cadModel (String modelName)
    throws ModelNotFoundException;

public void import_cadModel   (String xmlFile)
    throws XMLDocNotValidException;
```

Figure 4-22 Proposal of import and export method for OMG CAD Services

As a proposal to future versions of the OMG CAD Services specification, a set of detailed functions are proposed that cover important export and import functionality for a structure-oriented model exchange (see annex 1). Once included, the CAD Services specification would provide the basis for the realisation of an implicit CAD

API through which – in a long-term vision – separate model processing and transcription engines could become obsolete.

4.5.4 Model Processing

The choice of XML as a format for the structure-oriented transfer strategy leads to specific main functions of the processing engine. Figure 4-23 shows these functions for the case of model export: A CAD model analysis function provides model information to a DOM generation function. This XML document object model constitutes the implicit run-time representation of the CAD model. The transcription function generates an XML data stream according to the XML schema, as defined in Figure 4-20. Model import information flow is orientated conversely: An XML data stream is transcribed into an implicit run-time representation, i.e. an XML DOM, which is passed on to a CAD model regeneration algorithm.

In the following, mechanisms of CAD model analysis and regeneration shall be discussed in detail. These mechanisms can be derived from the characteristics of the implicit representation for CAD model data.

For model export, the part structure at assembly level can be mapped directly. Adjacency, relationships are either converted into coordinate references or transformed into corresponding feature objects. If not explicitly identified, the first part object is declared the master part.

At part level, the feature tree is mapped into a binary DOM tree. The representation of feature objects may require a mapping of position and direction according to the unified feature library specification. Explicitly generated or referenced shape elements are retrieved from the CAD system and represented by type, name, and characteristic position that guarantees for unique identification. The constraint model is represented in a separate list of constraints, free variables and the corresponding local solution.

Model regeneration mechanisms operate analogously. At assembly level, the master part and further parts and features are instantiated and positioned. A part model is generated by means of feature instantiation according to the unified feature library specification. The corresponding shape model is generated automatically by the CAD system. Instantiation of structural feature nodes results in a Boolean operation of the associated shape models. Explicitly generated shape, i.e. 2D sketches and auxiliary geometry like rotation axes, is created by means of the implicit CAD API.

Explicitly referenced shape elements will have been created at the time they are referenced from a shape manipulating feature. These entities are retrieved from the shape modeller by the aid of characteristic coordinate references. To achieve higher fault tolerance, shape entities can be searched for in an adjustable neighbourhood of the given position. If necessary, additional plausibility analysis checks on the near-by topology can be applied (intelligent coordinate referencing; see section 4.3.6).

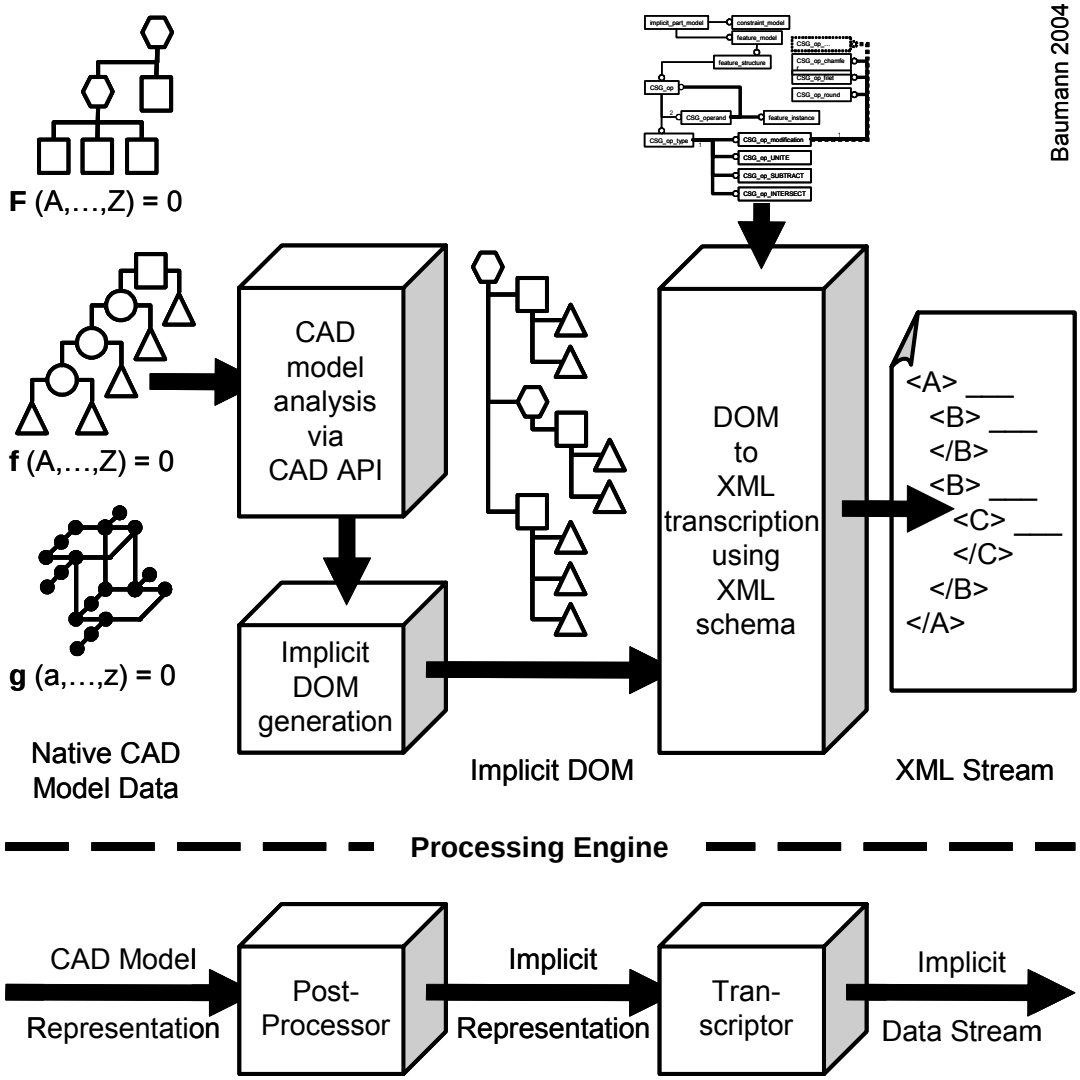


Figure 4-23 Main functions of an XML-based model processing engine and information flow for model export

The parametric model is instantiated after the complete assembly or part model is re-generated with explicit parameter values. As discussed in section 4.3.6, the installation of the parametric model depends on constraint system compatibility. Fully parametric constraint systems must be re-installed in the exact order of their definition. Variational constraint systems have to be re-installed completely before a re-calculation is invoked. For a fully parametric solver, a variational constraint system is ignored.

Two additional aspects of fault tolerance shall be mentioned. The question of geometric accuracy is combined with the problem of mismatching topological shape entities that arise from computation of corresponding geometry entities – which is

subject to errors. Generally, this is a minor problem to the implicit exchange because, as the complete shape model is generated by the receiving system, the accuracy by which these mismatches are handled is not “transferred” from one system to another. It is also less a problem for 2D than for 3D shape [89]. Nevertheless, for regeneration of explicitly exchange shape elements it still necessary to consider computed geometry as in curves and free-from faces. For that purpose, a tolerance dimension can be specified that enables the import processor to adjust the accuracy criteria of the receiving system. This presupposes the availability of such a functionality through the system’s API.

If identification of explicitly referenced shape entities fails completely, the import processor may skip instantiation of the concerned feature and try to continue model regeneration at the next structural level. In many cases, shape manipulating features, like blends and chamfers, are included with no dependencies to later feature objects. Ignoring these features will still result in a stable and consistent model.

4.6 Synthesis of the Structure-Oriented Exchange Method

In the preceding sections, alternative solutions for various conceptual aspects of an implicit model representation and transfer strategy have been examined. Summarising the conceptual discussion, the final methodical approach shall be outlined, which is proposed as the new structure-oriented exchange method.

Main characteristic of the implicit model representation is its utilisation of feature technology capable of representing semantic and shape model information. As the object-orientation of parameterised features completely abstracts from shape modelling, features are the fundament of the implicit approach.

A unified feature library is proposed that standardises design objects common to all CAD systems. Preferably, sub-libraries should be standardised according to different fields of design applications.

The feature model structure is represented as a system-neutral binary tree combining features and structural nodes to a design structure. User defined features are represented as a sub-model of the feature structure using the same modelling operations as provided for part modelling.

The CAD shape model is only represented, when explicitly defined by the user, i.e. 2D sketches, swept curves and auxiliary geometry, or explicitly referenced from shape manipulating features. The identification issue is resolved by an intelligent coordinate referencing strategy that makes use of plausibility assumptions on the topology adjacent to the searched element. Different from current opinion in literature, a persistent naming mechanism is not necessary.

In order to assure constraint solving compatibility, the parametric model is instantiated preferably after the feature model has been re-generated. Both fully parametric and variational modelling is supported. For mathematical reasons, variational constraint systems cannot be transformed into fully parametric constraint systems. However, structure-oriented model exchange is still operational if the parametric model is skipped in this case.

For free-from modelling, a taxonomy is proposed according to feature instantiation principles. Free-from features are classified into application- and shape-oriented elements; the latter are distinguished into mathematical, explicit and implicit free-form features. Explicit and implicit free-from shape generating features are represented analogously to their regular shape generating counterparts. For mathematically defined free-form features, NURBS are chosen as external representation mechanism.

Assembly part structures are represented as n-dimensional trees of part and feature elements. Positioning is realised by absolute or relative coordinate referencing. Adjacency relationships are represented by shape neutral placement features. Main characteristic of implicit model representation is the application of one structuring mechanism for the definition of parts, assemblies and user defined features.

As a basis for a model transfer strategy, three representative scenarios have been defined which characterise today’s situation in product model exchange: pure file transfer, exchange within a web-based environment, and direct system integration. With regard to these scenarios, XML is identified as the optimal choice of a physical format for structure-oriented product model data. An XML document type definition is proposed for implicit model representation. For the generation of corresponding XML data streams, separation of implicit model representation and transcription into the physical format is identified as an important characteristic.

The Implicit CAD API is proposed as a standardised functionality of application programmable interfaces to CAD systems. As a contribution towards its realisation, a set of interface functions essential for structure-oriented model exchange is proposed to be included in the OMG CAD Services specification. Furthermore, of import and export processing functionality is specified in detail.

5 Realisation, Proving and Evaluation

5.1 Realisation of Representation Principles

In the last chapter, an implicit approach to structure-oriented data exchange has been developed, consisting of an implicit representation method and a corresponding transfer strategy. Realisation of these concepts has taken place and is going to be described in two stages: Firstly, principles of implicit model representation have been realised as part of a nationally funded research project. The implementation was based on exploratory feature modelling systems in order to achieve a proof of concept. Secondly, the concepts have been adapted to commercial CAD systems considering industrial exchange situations.

Following, both realisation activities and applied use cases are outlined. Results are evaluated in consideration of business requirements and with regard to the thesis statement in section 3.4. It shall be verified whether the implementation shows feasibility of the proposed concepts for a structure-oriented model exchange. Industrial statements are quoted assessing the findings from their company's perspective. Finally, investment costs and possible benefits are analysed.

Realisation of an implicit exchange mechanism was subject of a research project titled "Exchange of Semantic Information on the Basis of Implicit Feature-oriented Product Descriptions" ("Austausch semantischer Informationen auf der Basis impliziter Feature-orientierter Produktbeschreibungen"), funded by the Deutsche Forschungsgemeinschaft (DFG). The DFG project KR 785/11-1,2 was located at the Institute for Machine Tools and Factory Management (IWF) at the Technical University of Berlin, Germany, under the supervision of Professor Dr.-Ing. F.-L. Krause. Mr. M. Wang was involved in the implementation. Cooperation was established with the Institute of Production Engineering and Machine Tools (IFW) at the University of Hannover, Germany. The corresponding DFG project To 56/134-1,2 was conducted by Mr. P.-O. Wölk under the supervision of Professor em. Dr.-Ing. Dr.-Ing. E.h. H. K. Tönshoff.

The project had emerged from experiences in feature-based modelling and corresponding feature modelling systems FEAMOS and EMOS at both institutes. On the basis of the formal feature representation languages PDGL and TEBES, the project aimed at realising a new method for the exchange of semantic information across both systems [136]. Distinct views upon the product development process and corresponding product model data were assumed: exchange of pure product design information among CAD systems (horizontal exchange) was distinguished from the exchange of product models along the process chain of design to manufacturing planning (vertical exchange). In the project, IWF concentrated on horizontal, whereas IFW was concerned with vertical exchange. Unless otherwise noted, the following realisation summary is limited to findings on horizontal exchange the author was responsible for at IWF.

Implicit representation principles for feature-based models have been implemented for the FEAMOS and EMOS systems. The system neutral representation consists of two substantial parts: a feature library specifies available design elements and corresponding instantiation methods in a generic form; and a structural description represents feature instances and feature model structure. The representation format is derived from the PDGL and TEBES languages (see also section 2.1.2) which have been harmonised to be compatible with both feature modellers. The resulting format has been published under the name *Feature-based Definition and Exchange Language* (FEADEL) [84].

FEADEL defines syntax and grammar for the system neutral specification of feature types as well as for the definition of a CAD model at instance level. FEADEL representation capabilities include parameterised semantic and form-features that can be specified by a large variety of feature attributes and shape aspects, Boolean operations on feature instances, constraint definitions following the variational design philosophy, free-form features based on mathematical, i.e. NURBS-based, representation. Representation of assembly structures and additional multi-medial information has been added by IFW. Another capability is the exchange of modelling functionality through representation of additional feature methods describing higher-level modelling algorithms. This is achieved by implementing a run-time interpreter for these modelling functions. In this connection, FEADEL also suits as a scripting language for CAD modelling functionality.

Implementation has included adaptation of the existing PDGL language and the corresponding interpreting engine to the FEADEL concept and realisation of additional CAD model pre- and post-processors (Figure 5-1). CAD models are transferred as human readable text files; Web-based and procedural system integration has not been considered.

FEADEL is based on a set of CAD API functions harmonised for both feature modellers. As both systems incorporate the same ACIS shape modelling kernel, the CAD API is basically an abstracted subset of the ACIS library suitable for hybrid shape modelling. Additional instructions were added for feature-based and parametric modelling. As FEADEL emerged from the harmonisation of PDGL and TEBES, the support of user defined features is language immanent but limited to the capabilities of the ACIS API. UDFs are represented as a modelling result on shape level and insofar different from the concept proposed in section 4.3.5.

The FEADEL processor for FEAMOS has been implemented using the FLEX and BISON. These interpreter development tools allow for a comparably comfortable realisation of lexical scanning and grammar parsing modules for formal languages. Implementation has been carried out on an IRIS system environment on a Silicon Graphics workstation. Programming language is C / C++.

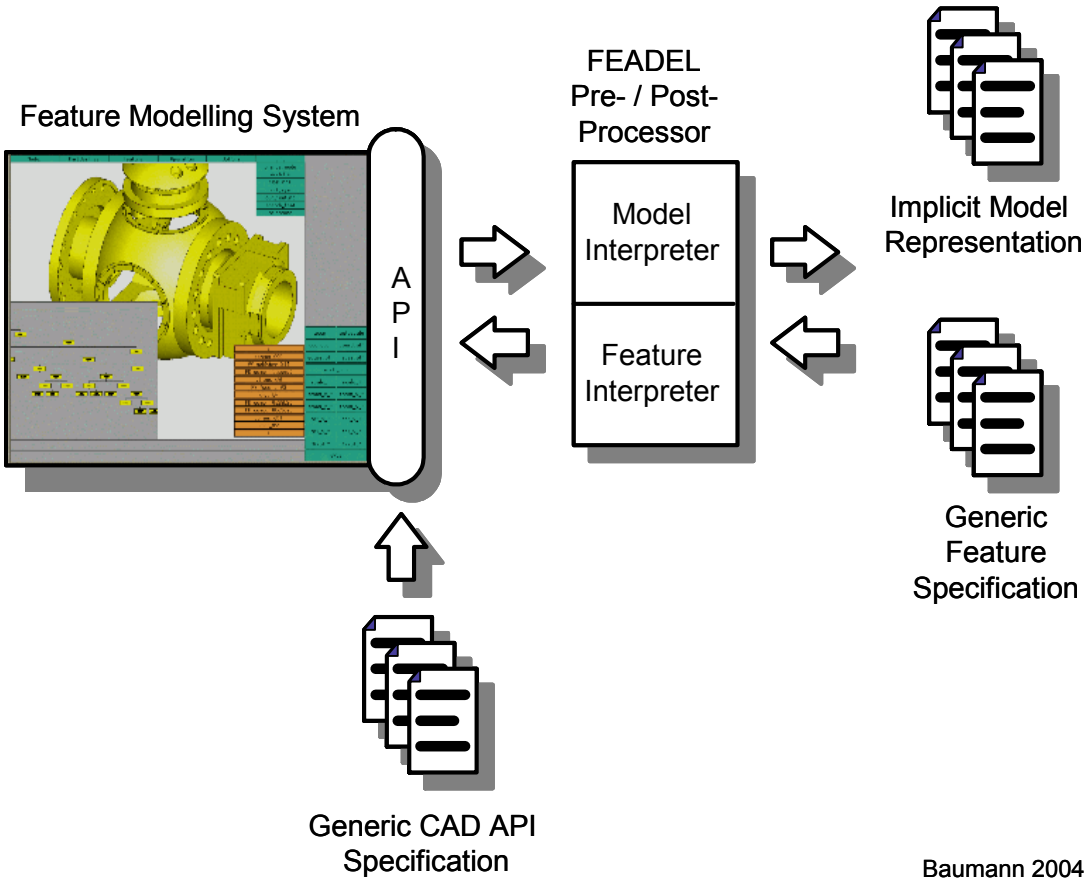


Figure 5-1 Implicit exchange using FEADEL for Feature Modelling System

5.2 Realisation of a CAD Model Exchange Environment

The second stage of realisation has aimed at proving conceptual adaptability to commercial CAD systems considering industrial exchange situations. Cand. ing. Masoud Gholchin participated in the implementation of major software modules.

Representative for the variety of commercial feature-based CAD systems, two modellers from different vendors have been chosen: Unigraphics version 18 by UGS PLM Solutions and I-Deas Version 8 by SDRC[1]. The systems have been selected according to availability at the institute and system openness through a well-documented API. Unlike the first realisation stage for the university feature

1 At the time writing, SDRC has been taken over by UGS, and both systems are being migrated. Nevertheless, realisation and evaluation as described in this dissertation, is based on the system versions available when both systems and companies were totally disjoint.

modellers FEAMOS and EMOS, which both are based on the ACIS, the second phase should prove feasibility for commercial CAD systems operating on different shape modelling kernels. I-Deas utilises a native kernel, whereas Unigraphics is based on the Parasolid engine, which, similar to ACIS, is available as a commercial stand-alone shape modeller.

For both systems, implicit part representation and XML-based model transfer via an interactive client has been realised (Figure 5-2). For implicit representation, a number of parameterised shape generating and shape manipulating features have been harmonised, creating blocks, cylinders, cones, spheres, blends, filets, chamfers or blind holes. Additionally, extrude and revolve features support solid generation from 2D sketches. Protrusions generate linear sweeps from a closed edge loop with given protrusion angle and distance; a revolve operation creates a rotational part with given rotation angle and axis.

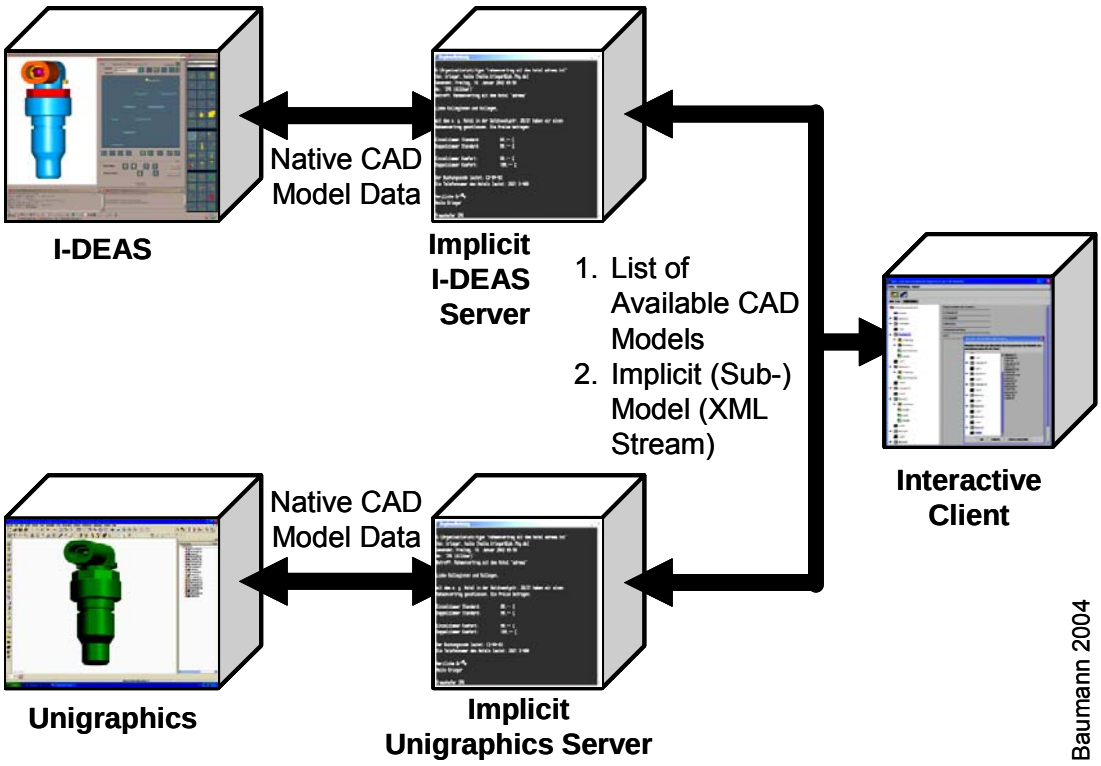


Figure 5-2 Implemented components and information flow for commercial CAD model exchange

As curve elements within 2D sketches, straight lines, circles and circular arcs are supported. The coordinate referencing approach has been implemented for shape entity identification. Explicitly sketched elements are located by a necessary number of points characterising start, end, centre and additional positions. For the export of

round and chamfer features, the original edges had to be calculated, as corresponding API functions were not available. For model regeneration, these edges can be identified in a neighbourhood of the calculated position. The search interval is chosen according to blend radius or chamfer width respectively.

The part model feature structure is represented as a binary tree of features and structural nodes. Combinations include unite (join), subtract and intersect operations.

Implemented aspects of the model transfer strategy comprise one processing server for each of the systems, incorporating an implicit run-time representation engine and a XML transcription unit based on the implicit XML representation schema. In order to support the three exchange scenarios, an implicit CAD server for I-Deas and Unigraphics provide file-based as well as direct communication.

The interactive exchange client communicates with both CAD servers and suites as a user interface to exchange functionality (Figure 5-3). The user may request a list of available models from the sending CAD server and may invoke the model export procedure on a selected model. The implicit model is presented to the user. Feature structure, feature type, name and identifier, attributes, position, parameter values, and further information are shown in a liner list. The client also indicates, whether the feature is suppressed, i.e. has not effect on the modelling result.

The complete or a sub-set of the part model can be transferred to the receiving CAD server. In order to prevent the user from generating an inconsistent sub-model, the client performs a validity analysis on the user's entity selection, as shown in the smaller dialog window in Figure 5-3. This guarantees that structural rules of the implicit representation are obeyed. Furthermore, all feature types are checked for support through the import processor of the receiving system.

According to the application integration exchange scenario, a transfer infrastructure has been realised. Figure 5-4 illustrates the resulting architecture. For I-Deas, the OpenIdeas API was used which provides a CORBA-based server for implementation purposes. The UGopen interface to Unigraphics consists of a C library. Import and export engines for Unigraphics have been realised in separate executables written in C. For DOM generating and XML transcription modules, the XERCES C library version 1.2.0 has been applied [144]. The processing engines for I-Deas are integrated in the implicit I-Deas CAD server; the modules have been implemented in Java. The DOM generating and XML transcription modules have been realised using the DOM4J library version 1.1 [145].

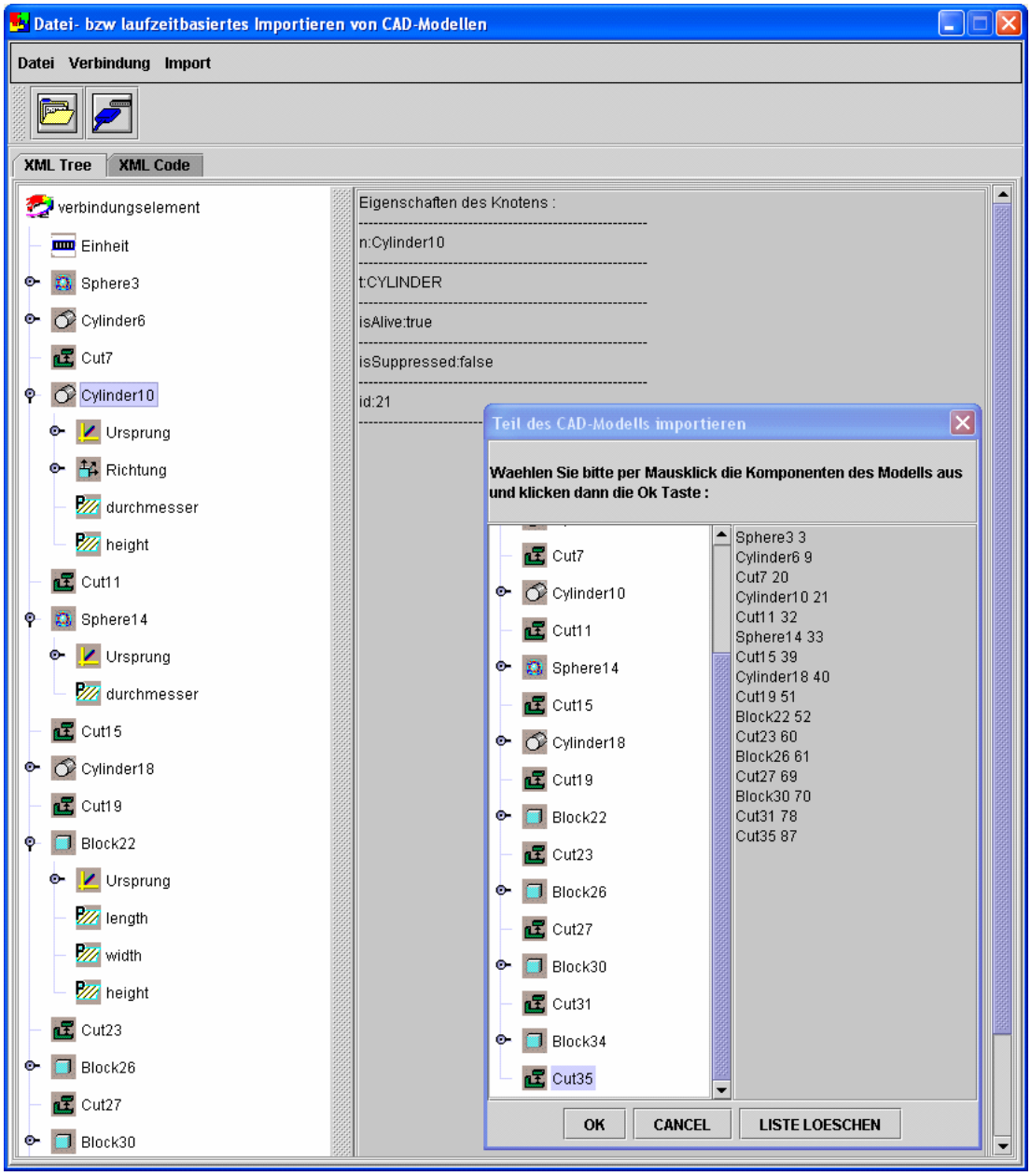


Figure 5-3 The interactive exchange client enables selective model exchange

For the application integration scenario a CORBA-based middle-ware has been chosen, since OpenIdeas communication is already based on CORBA. The implicit CAD servers and the interactive client have been equipped with their own Orbacus ORB version 4.1 [146]. Communication is established via usual CORBA object calls and referencing within a local network (intranet).

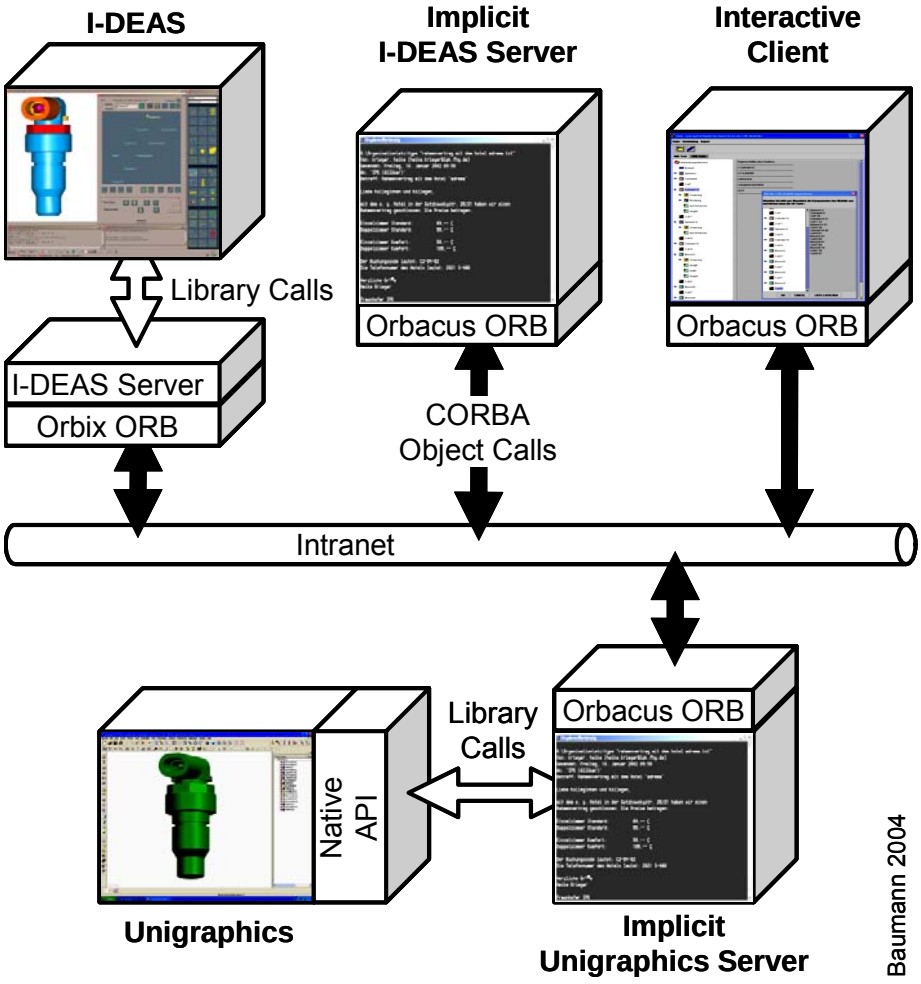


Figure 5-4 Communication of implemented components in an application integration scenario

According to the Web-based exchange scenario, the exchange components are equipped with corresponding communication units (Figure 5-5). An implicit CAD servers servlet reside on a Tomcat Web-server. Tomcat is a component of the Sun Java Web-Services Development Pack, applied in version 1.2 [147]. Being implemented as a Java application, the interactive client can either be installed as a stand-alone component or be executed as an applet within a Web-browser. The client communicates via SOAP messages transferring implicit CAD models as XML streams from and to the CAD servers.

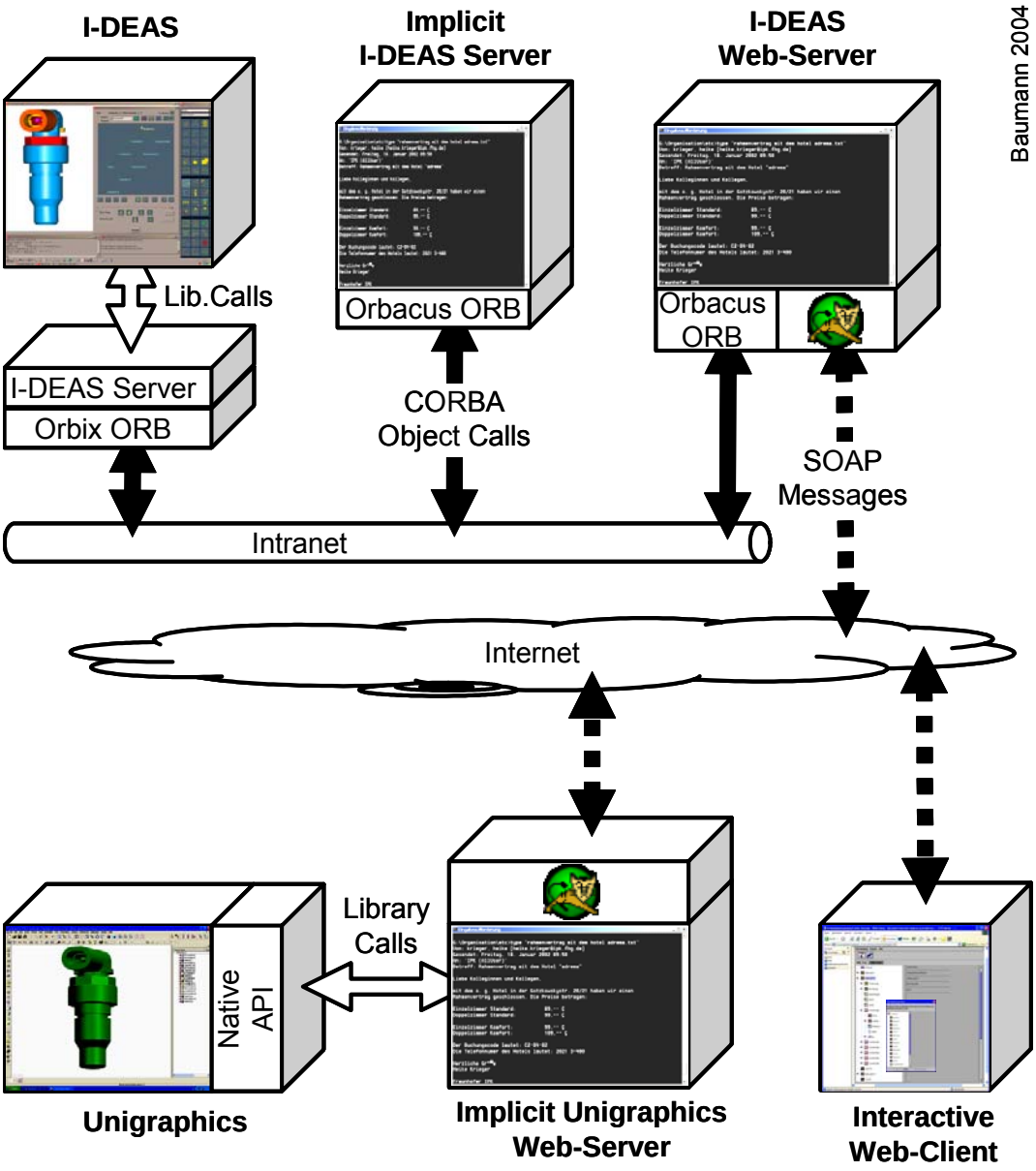


Figure 5-5 Communication of implemented components in a Web-based exchange scenario

5.3 Use Case Description

Implementation results of both realisation phases have been evaluated considering two use cases. For verification of representation principles implemented in phase one, a friction gearbox has been schematically designed in FEAMOS and EMOS (Figure 5-6). The primary emphasis has been put on a balanced representation of different information types within the modelled parts and features. An application-oriented design was considered less important for the evaluation purpose.

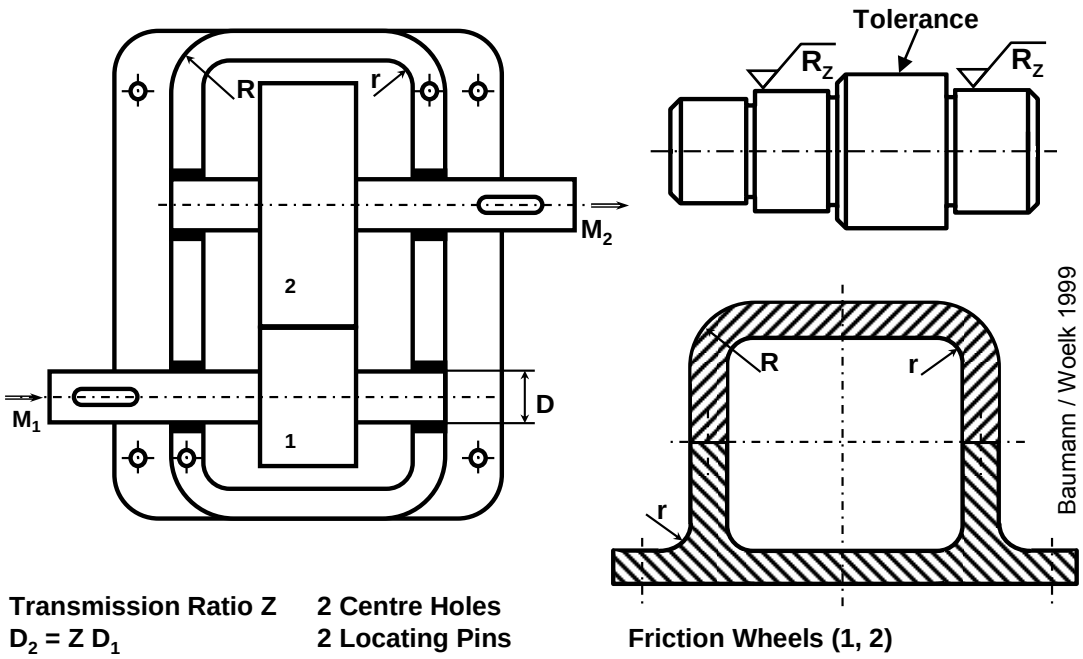


Figure 5-6 Evaluation use case: friction gearbox and shaft

Form-features, such as blocks and cylinders, and shape manipulating features, such as blends and chamfers, are used in the gearbox model. Semantic features represent technology information, such as surface roughness of the bearing carrier and tolerated shaft diameters. An additional semantic feature groups design spanning variables, e.g. transmission ratio *Z*, fillet radius *R* and *r*, torsional moment *M*, and connecting diameters D_1 and D_2. Each part has been stored as a user defined feature and applied for gearbox design.

Functional associations within and between the parts are represented by parametric references. For example, the blend radii *R* and *r* are applied for all blends and fillets features; and the inner friction wheel and bearing bush diameters are calculated from outer shaft diameters and given tolerances. The complete design consists of 280 parts and features. By application of complex user defined features for housing top and bottom case, shafts, friction wheels, location pins, and design spanning variables, the resulting assembly is reduced to less than 30 elements (Figure 5-7).

User defined features and assembly design have been exchanged as FEADEL files. A multi-parsing strategy is needed that firstly installs unknown feature types to the system and secondly re-generates feature instances, model structure and parametric model. As the feature modellers have been open to the developers by source code, processor development has been completed until the entire model content could be exchanged. All design elements are instantiated by native modelling functionality and thus are fully alterable. Dissimilarities in the resulting shape model cannot be perceived.

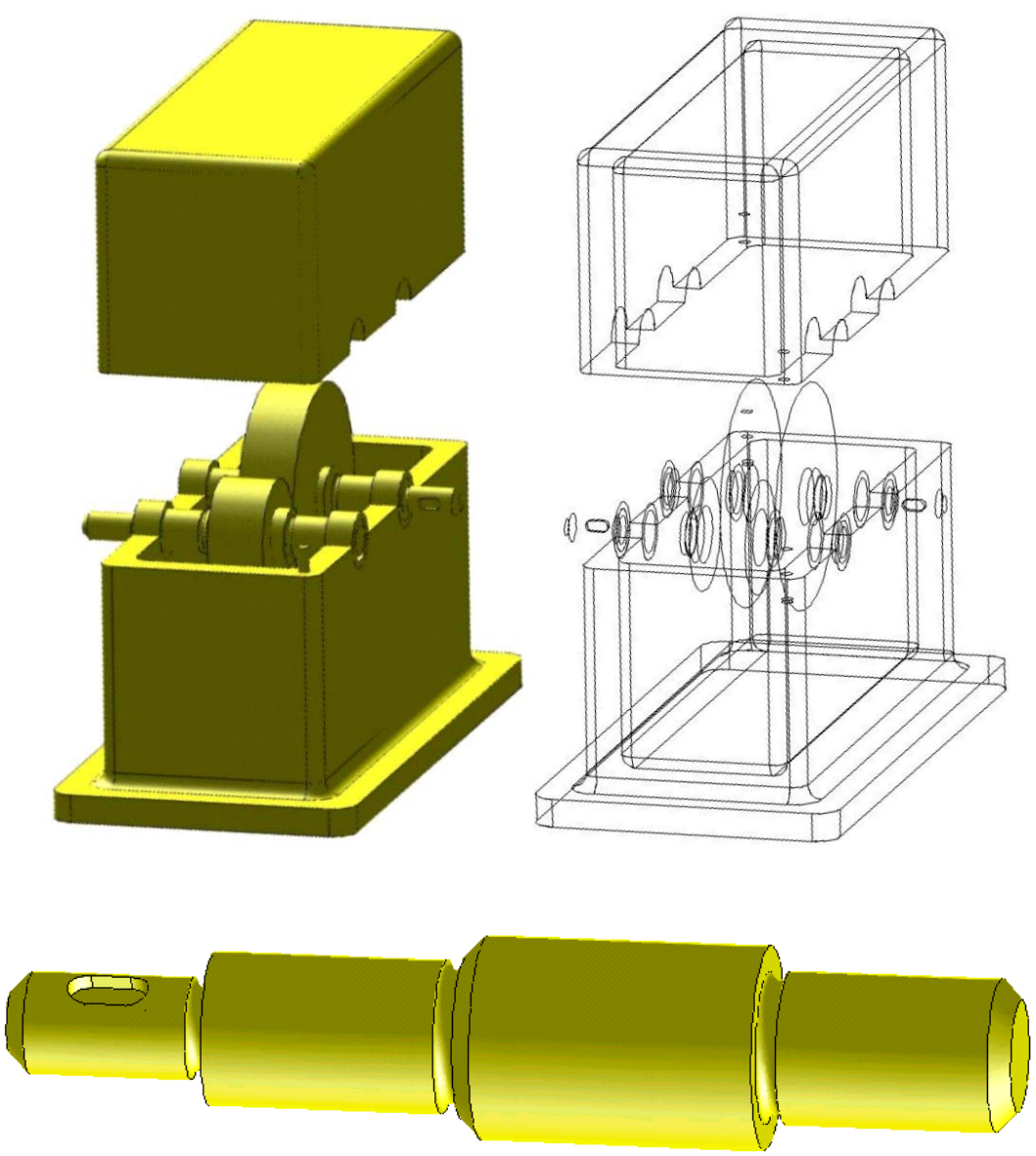


Figure 5-7 Friction gearing and shaft (FEAMOS model)

The use case for results evaluation of the second implementation phase includes exchange of numerous models. Most of the exchanged parts have been recruited from the seventh and eighth STEP AP214 processor benchmarks [134], [135]. These benchmarks are performed by the ProSTEP-iViP Association on a regular basis. The author likes to express his thanks for the friendly provision of the I-Deas and Unigraphics native files. The tested components are parts of a vehicle pump assembly originally provided by Volkswagen.

All parts are successfully transferred from and to both CAD systems. All features and attributes are accessible, models can be altered by normal modelling functions of the receiving systems. Features can be suppressed, reordered and replaced within the feature structure. Altogether, part models behave natively even after several exchange cycles during which numerous modifications have been made to the designs. Comparison of the resulting shape models reveal no deviations.

For an evaluation of the interactive client, the java bean has been tested in a CORBA-environment and within a Web-browser. Model transfer assessment has included establishment of run-time connections to both CAD servers according to OMG CAD Services regulations, requests for available CAD models, export and graphical presentation of selected CAD models, sub-model definition, consistency validation, and import to the receiving CAD system.

5.4 Evaluation and Thesis Proving

After examining the implemented software components in two use cases, all conceptual aspects of structure-oriented CAD model exchange have been addressed. Focusing on model representation, general feasibility of the implicit approach can be ascertained. Exchanged models appear equal in shape and provide further alterability through all native modelling capabilities.

Due to different feature specifications provided by the different CAD APIs, the processors have to perform shape mapping during model re-generation. Implementation and run-time computing effort for these algorithms emerged as unexpectedly high. This underlines the conceptual demand for a unified feature library as an element of an internationally standardised CAD API.

The coordinate referencing strategy for identification of explicitly referenced shape elements worked out rather stable. This proves that implicit model representation is not limited to shape generating features. Comparatively long computing time for shape identification in I-Deas demands for specific API functionality.

Implementation of UDF representation and transfer using FEADEL for university feature modelling systems has proven feasible at least for the case of identical shape modellers. For a general application of this concept, a harmonisation of shape modelling capabilities is required. The sub-tree definition concept, which specifies a user defined feature as a sub-set of a part model feature structure, has been realised for I-Deas. It constitutes the more general approach and thus is preferable.

Feature instances can be generated including parameters and their values. The exchange of semantic features has been realised for the university feature modellers using FEADEL as a specification language for features at type level. As corresponding semantic feature modelling capabilities in commercial CAD systems are comparably rudimentary, the evolution of a unified feature library is probably the more promising approach. The exchange of assembly models and variational constraint systems among the feature modellers has also been successful.

The resulting size of transferred XML files has been compared with corresponding STEP physical files generated from Unigraphics AP214 processors. Although the XML structure is principally less compact than a STEP physical files, the latter averaged out to a factor of five to ten larger that the XML files. The actual difference in size depends on the amount of explicitly represented shape elements.

With respect to user requirements defined in section 3.3, the evaluation of achieved results can be summarised as shown in Table 5-1.

Business requirements	**Assessment**
Retain design results	Fulfilled
Enable model alterability	Fulfilled
Shorten engineering cycle times	Fulfilled: No additional effort for manual reword
Reduce data traffic	Fulfilled: Reduction by 50–90 %
Support multi-CAD part library	Fulfilled
Maintain design systematics	Fulfilled
Support multi-disciplinary design	Not implemented
Support iterative process chains	Fulfilled for horizontal exchange
Support selective model analysis	Fulfilled
Support data management	Enabled
Support collaboration and modern WEB based exchange mechanisms	Fulfilled
Provide fault tolerance	Fulfilled
Support CA systems interaction	Fulfilled
Provide applicable strategy	Fulfilled

Table 5-1 Evaluation of structure-oriented exchange method with respect to user requirements

Implementation has also demonstrated that structure-oriented model exchange can be organised to effectively facilitate industrial exchange situations even for commercial

CAD software. Choosing XML as a physical format for data transmission allows for a synergetic conception of processor components. Separation of model processing and communication functionality has led to efficient implementations for different communicational environments.

The application of an interactive client provides additional exchange functionality to the user. Specifically, selective model exchange and consistency validation are valuable and prototypically demonstrate the integration of other engineering applications based on the structure-oriented exchange method.

Regarding run-time behaviour, short export and import cycles demand for specific CAD API functionality and direct library access. The provision of a CORBA-based CAD server for I-Deas is somewhat comfortable for realisation of client-server communication, but for implementation of model processing engines insufficiently slow.

In section 3.4, a thesis for this dissertation has been postulated in tree statements:

1. The model structure is the essential information responsible for creation as well as for modification of the CAD model, which leads to the introduction of structure-oriented model exchange.

2. Two possible approaches to a structure-oriented exchange method can be identified, referred to as the implicit and the explicit approach.

3. The implicit is suitable and the preferable approach.

The first aspect has been developed as part of the characteristics definition for the new exchange method in section 4.1. The implicit approach has been identified as the superior alternative in consideration of technical requirements (section 4.2.4). By conception and realisation of an implicit approach, central characteristics have been achieved: The implicit representation method is shape preserving, i.e. it retains the resulting shape of the product. It is also structure preserving, i.e. it transfers a structural model that allows later alteration within the receiving system. According to the definition in Figure 4-3 on page 37, the implicit approach fulfils the characteristics of a structure-oriented exchange, which proves the thesis statement.

Furthermore, evaluation of implementation results has demonstrated the fulfilment of requirements on the structure-oriented exchange, which verifies that an applicable strategy has been developed and realisation. Suitability for industrial application has been a subject of exploratory discussions with various engineering companies. Following, some of these industrial opinions shall be quoted.

UGS PLM Solutions[1] as a representative CAD system provider evaluates the implicit approach to a structure-oriented exchange of CAD model data to be of interestingly

1 The author thanks Mr. Uwe Stach, UGS PLM Solutions, Germany, for his friendly engagement in discussing the project's goals and achievements.

high potential. The prototype implementation proves the feasibility of central mechanisms for part model exchange like sketch- and primitive-based modelling. The solution found for identification of shape elements for blend and chamfer features appears promising. Crucial for commercial appliance appears a rather high tolerance level regarding features and modelling operations applied by the user. Industrial acceptance will strongly depend on the independence of the exchange format from the model content. A "design for exchange" situation must be avoided by any means. The concept for supporting user-defined features is a helpful step in this direction.

CADsys GmbH, Chemnitz, Germany, specialises in development of core modelling functionality for product development tools such as the feature modelling kernel of the *CADdy ++* mid-range CAD system and the CACD tool *FOD*. According to CADsys[1], the proposed method is the only and long awaited alternative to established exchange strategies. Even in comparison to various direct CAD-to-CAD interfaces which usually ignore important structural model content the structure-oriented exchange promises to be an achievement. Nevertheless, for the industrial application further feature representation details need to be implemented. Widely applied are features that do not generate a stand-alone shape instance, such as blocks or cylinders, but directly add or subtract shape to or from a selected face, such as a bore-hole feature, need to represent the input shape entities at instantiation time. This is similar to blend or chamfer features directly referencing to the corresponding edges or vertices they operate on. Since the mechanism for referencing to faces or datum planes seems little different from referencing to edges and vertices the proposed exchange method, in principle, appears to be meeting this requirement.

The European Airbus consortium[2], being one of the largest users of CAD technology, is using data exchange procedures for company internal processes as well as in cooperation with suppliers. As part of the migration efforts to CATIA V5 many design models originally created in CADDS5 or Pro/Engineer need to be converted to CATIA V5. Furthermore, much of the design experience modelled in user-defined features impends to get lost during that migration. From this perspective, the structure-oriented method for CAD data exchange is expected to improve the migration results substantially. Model alterability and transfer of UDF definitions after import are functionalities that can help reduce migration expenses. For industrial application the identification of referenced off-set faces is considered an important capability. The solution found for identifying referenced curves in blend and chamfer operations shows that the proposed exchange method is generally applicable.

1 The author thanks Mr. Frieder Swoboda, managing director and head of CADsys central development, for his time and interest in the subject and for the fruitful discussion on special representation and transfer issues.

2 The author thanks Mr. Arnd Rothe, project manager in the Airbus CAD department for his kind support and interest in the research results and for the helpful discussion about crucial aspects of their industrial application.

Finally, Bosch[1] stresses the possible impact of the structure-oriented method on data exchange to other than CAD systems. Every product component is evaluated in one or several simulation processes, such as FEM and multi-body simulation, thermal and acoustic analysis or others. Specialised simulation engineers spend a significant percentage of their efforts on the generation of digital mock-ups, i.e. analogues models suitable for various simulations purposes, out of the corresponding CAD model. In order to increase simulation quality, the original product shape is modified. For instance, shape complexity is reduced by eliminating details like blends and chamfers and other elements irrelevant for the physical effect addressed by the specific simulation routine. Today's simulation systems are usually provided with a shape model from which the simulation model is derived. A simulation specialist has to import a shape model exported from CAD and adjust it to his requirements by means of low-level shape modelling mechanisms. Alternatively, a corresponding change request has to be directed to the original designer.

At Bosch, several hundred experts are engaged in simulation activities spending a significant amount of their time on regenerating shape models or on waiting for adequately adjusted models. In principle, product model adjustments on structural level are by circa 70 to 90 percent more efficient than on shape level. Assuming that a simulation engineer had access to a structure-oriented CAD model that could be passed on to a corresponding import mechanism for simulation tools, an immense benefit would be achievable.

5.5 Cost-Benefit Analysis

5.5.1 Investment Cost Estimation

Realisation of structure-oriented exchange as a commercial software solution necessitates initial as well as periodic implementation. These efforts have to be calculated per CAD system applied in a company, as implementation will presumably be taken over by the corresponding CAD system vendors.

Realisation of basic exchange functionality includes implementation of a model analysis and regeneration engine and of a XML-based transcription processor. Presupposing adequate programming experiences, this functionality can be realised in 1.5 to 2 person-years. Periodically, model exchange functionality has to be updated with the constant development of the CAD system. Usually, system vendors publish new system revisions every 6 to 12 months. The estimated effort for exchange processor adaptation is 2 to 4 person-months a year.

1 The author tanks to Dr. Ralf Mendgen, Section Manager CAD/CAM Techniques; Corporate Research and Development at Robert Bosch GmbH for his interest in the subject and for the friendly provision of corresponding statistics.

Comfortable Web-based or system integration functionality implementation requires further 6 to 12 person-months. Adaptations to new versions of applied communication standards, such as CORBA or Web-services, have to be considered although reliable cost estimation cannot be given. Additionally, license fees may be granted for XML development, communication and other libraries.

A commercial version of the interactive exchange client can be realised in circa eight person-months. Periodical updates to future CAD system releases will not exceed four person-months per year.

These realisation efforts will probably lead to a periodical licence agreement. Most practically, a structure-oriented exchange processor and interactive exchange client will be part of an exchange functionality package that also includes STEP, IGES and other exchange processors. The license agreement will then cover proportionate initial and periodical expenses.

Considering the comparably high complexity of STEP AP214 compliant processors, implementation of a structure-oriented model exchange is less costly and will reach stable software revisions far earlier. It can be estimated that license cost for a structure-oriented exchange processor may constitute less than 50 percent of a STEP AP214 processor.

Additional hardware investments will not be necessary.

5.5.2 Benefit Estimation

Benefit analysis for companies that want to install structure-oriented exchange technology is complex, and reliable statistics on key parameters are not publicly available. The main difficulty results from the fact that industrial product development processes are arranged to avoid direct model exchange between different CAD systems. Reengineering of these processes is usually subject to mid-term projects; corresponding project volumes are company specific.

For benefit estimation, a product development process is assumed in which shape models are exchanged among several CAD systems. A known number of engineers is engaged in shape model adaptation and in regeneration of other model content that is not exchange by conventional processors. The benefit mainly results from 70 to 90 higher efficiency of features-based and parametric modelling compared to pure shape modelling. The following parameters have to be considered:

- N – Number of CAD engineers;
- p_1 – Percentage of engagement in model adjustment or regeneration (in %);
- p_2 – Percentage of time saving by feature-based and parametric modelling compared to pure shape modelling (in %);
- C – Average annual personnel cost (in Euro per person);

b – Annual benefit per person (in Euro per person);

B – Total annual benefit (in Euro).

The annual benefit per CAD engineer calculates as follows:

$$b = 0.0001 \cdot C \cdot p1 \cdot p2.$$

The total annual benefit results to:

$$B = 0.0001 \cdot N \cdot C \cdot p1 \cdot p2.$$

For a product developing company an arbitrary number of 1’000 CAD engineers shall be assumed who spend at least 10 percent of their time with shape model adaptation and model regeneration. Considering average annual personnel cost of 200’000 Euro, the annual benefit results to 14’000 to 18’000 Euro per engineer or to a total of 1.4 to 1.8 Million Euro per year.

For product simulation processes similar to the situation at Bosch (see above), the possible benefit can be estimated analogously. It shall be assumed that Europe-wide a number of N = 400 to 800 simulation experts are spending at least $p_1 = 20$ percent of their time on regenerating shape models or on waiting for adequately adjusted models. Implementation of structure-oriented exchange would enable these engineers to transfer the feature-based CAD model to a corresponding import mechanism for simulation tools. The economisation results to 28’000 to 36’000 Euro per engineer or to a total of 11.2 to 28.8 Million Euro per year.

6 Summary and Outlook

Product developing companies react on increasingly high market pressure with strategies for raising efficiency and effectiveness in product development processes. A special optimisation focus is placed on the exchange of product model data as a time consuming but non-productive task. Product engineering demands higher quality data exchange along the product creation process and for flexible support of different exchange scenarios reaching beyond pure file exchange mechanisms.

Exchange of product model data between different CAD systems is most carefully considered. Specific demand is expressed for exchange technologies not only capable of transmitting the exact product shape across system borders but also supporting model alterability within the receiving system. These exchange technologies should be standardised to avoid costly development of native interfaces and to assure investments.

A survey on the state of the art in product model representation and exchange reveals that neither established standardised exchange technologies and formats, primarily the STEP technology, nor other research activities offer adequate solutions to these demands. Starting from this divergence of industrial needs and available technology, the objective of this dissertation is defined as the development of a new method for the exchange of product model data. This method shall consist of a system neutral model representation and a model transfer strategy. The scope of application is set to the exchange among feature-based CAD systems.

Discussion on main requirements leads to the definition of *structure-orientation* as structure and shape preserving characteristics of model representation. Two methodical approaches to structure-oriented model representation can be identified. The *explicit approach* adds structural information to a hierarchy of sub-models of shape representation. The *implicit approach* takes advantage of the fact that the product model structure incorporates the information, which is essential for model generation and modification. For that matter, apart from distinct exceptions, shape model representation is obsolete altogether.

Regarding technical requirements, the implicit approach is preferable. A *system neutral feature-based model representation* is developed. A unified feature library is recommended as a standardised basis for the exchange method. Representation concepts for part model structure, user defined features, parametric data and constraints systems, assembly model structure and miscellaneous model content are discussed in detail. A *free-from features taxonomy according to instantiation principles* is proposed.

A widely discussed issue concerning feature-based modelling is the identification of shape elements, known as the *persistent naming problem*, which currently is subject to research. Contrary to opinions in literature, an *intelligent coordinate referencing*

mechanism is proposed instead of assigning persistent identifiers to explicitly referenced shape elements.

As a fundament for a structure oriented transfer strategy, *three exchange scenarios are defined* that – in addition to classical file-based transmission – describe CAD data exchange situations in a Web-based and in an application integrating environment. Regarding these exchange scenarios, central *transfer functionality and components* are identified. XML is chosen as the most suitable technology and physical transfer format, and a corresponding functional concept is developed for model processing, transcription and CAD interfacing. Characteristics and functionality of an *implicit CAD API* are proposed, as a prerequisite for a fully operative and efficient implementation. Integration of this functionality into the OMG CAD services is recommended. A functional specification is presented for that purpose.

Realisation of the proposed concepts has taken place and is described in two phases. Realisation principles have been subject of a research project. Processors are implemented for the university feature modelling systems FEAMOS and EMOS. A formal *Feature Definition and Exchange Language* (FEADEL) provides constructs for the representation of feature types and feature-based models based on the ACIS shape modelling library.

The second realisation phase has addressed further conceptual development of the exchange method and adaptation to commercial CAD systems and industrial exchange situations. Implementation is accomplished for the Unigraphics and I-Deas system. The processing components are installed into a Web-based environment and into an application integration infrastructure. An *interactive client* is presented as a user interface to advanced exchange functionality, like selective model exchange and consistency analysis. Implemented mechanisms have been tested in two different *use cases*. Part models applied for exchange evaluation between the commercial CAD systems are recruited from ProSTEP AP214 processor benchmarks. Achieved results are successfully assessed in consideration of prior defined business requirements.

Realisation and evaluation demonstrate the implicit approach to structure-oriented exchange as a new and qualified solution that meets the demands for CAD model exchange technology applicable to commercial systems and industrial environments.

Future developments may address the application of the proposed concepts for other CAD systems, further specification of a unified feature library and standardisation of a harmonised CAD API.

Additional applications can be perceived in those fields of product creation that have been identified introductory but have been excluded from the scope of this dissertation. Structure-oriented exchange may suite for the integration of early design phases by enabling information flow from conceptual design to embodiment design and production planning. Specifically, the new exchange method may facilitate data exchange between CACD and CAD systems. Realisation of interdisciplinary design may be supported by an integration of, e.g. ECAD and MCAD modellers.

References

[1] Ross, D. T: *The early days of CAD*. In: Proceedings of the IFIP WG 5.2/GI International Symposium on Advanced Geometric Modelling for Engineering Applications, Berlin, Germany, November 1989, pp. 25–34. IFIP/GI 1990

[2] Verband der Automobilindustrie: *Jahresbericht Auto 2003.* http://www.vda.de/de/service/jahresbericht/auto2003/auto+maerkte/g_50.html; VDA; 2003, last called: 2004-05-06

[3] Krause, F.-L.; Baumann, R.; Böttge, U.; Jansen, H.; Kaufmann, U.: *Ergebnisse des iViP-Projektes.* In: Proceedings of Virtual Product Creation 2002 International Congress, 2.–4. July 2002, Berlin

[4] Tang, T.: *Kollaborative Lösungen auf einer Integrations- und Kommunikationsplattform – Basisvoraussetzung für die integrierte virtuelle Produktentstehung.* In: Proceedings of Virtual Product Creation 2002 International Congress, 2.–4. July 2002, Berlin

[5] Keil, H.-S.: *PDM- and CAD-Integration in Cross-brand Fahrzeugprojekten.* In: Proceedings of the ProSTEP iViP Symposium 2004, ProSTEP iViP Association, Darmstadt, 2004

[6] Brunnermeier, S. B.; Martin, S.A.: *Interoperability Cost Analysis of the U.S. Automotive Supply Chain, Final Report.* Research Triangle Institute for U.S. NIST, Gaithersburg, March 1999

[7] Sander, R.: *Neutraler WebClient zur Partnerintegration in Entwicklungsprozessen.* In: Engineering to Engineering Collaboration, Proceedings of ProSTEP Symposium, Wolfsburg 04./05. April 2001, ProSTEP Association, Darmstadt, 2001, pp. 97–104

[8] Bielohlawek, D.: *Business meets IT – Gemeinsame Zielsetzung für die virtuelle Fahrzeugentwicklung.* In: Proceedings of Virtual Product Creation 2002 International Congress, 2.–4. July 2002, Berlin

[9] Spur, G.: Krause, F.-L.: *CAD-Technik: Lehr- und Arbeitsbuch für die Rechnerunterstützung in Konstruktion und Arbeitsplanung.* Hanser Verlag, München, Wien, 1984

[10] Spur, G.; Krause, F.-L.: *Das virtuelle Produkt: Management der CAD-Technik*. Carl Hanser Verlag, München, Wien, 1997

[11] Spatial Technologies: *CATIA CAA product description*. 2003, http://www.spatial.com/products/CAA/?LV3=Y, last called: 2004-05-05

[12] Parametric Technology Corporation: *Pro/ENGINEER product information*. http://www.ptc.com/appserver/it/icm/cda/icm01_list.jsp?group=201&num=1&show=y&keyword=403, last called: 2004-05-05

[13] UGS PLM Solutions: Unigraphics Version 18 API Reference Manuel UGOPEN. Part of product documentation, 2002

[14] UGS PLM Solutions: I-DEAS product description. http://www.eds.com/products/plm/ideas/; last called 2004-05-05

[15] Spatial Technologies: ACIS Product Description, 2003 http://www.spatial.com/products/3D/modeling/ACIS.html?LV3=Y; last called 2004-05-05

[16] Ricoh Co.: DESIGNBASE V12 User's Guide. 2004

[17] UGS PLM Solutions: Parasolid Product Description. http://www.eds.com/products/plm/parasolid/index.shtm; last called 2004-05-06

[18] Weiler, K.: The Radial Edge Structure: A topological representation for non-manifold geometric modelling. In: Wozny, M.J.; McLaughin, H. (Eds.): Geometric Modelling for CAD Applications, Elsevier Science Publishers, Amsterdam 1988, pp. 3–66

[19] Conrey, J.; Lim, T.: 3D Modelling with ACIS. Saxe-Coburg Publications, 2001

[20] University of Rochester: I-DEAS shape model. http://www.lle.rochester.edu/pub/support/SDRC/ms8/SDRCHelp/LANG/English/library.htm; last called: 2004-05-05

[21] UGS PLM Solutions: *Unigraphics Version 18 API Reference Manuel UGOPEN.* Part of product documentation, 2002

[22] Spatial Technologies: *CAA Version 5 R7 API Reference Manual.* Part of online product documentation.

[23] ISO TC184/SC4: ISO 10303 Part 42 IS edition2: *Industrial automation systems and integration – Product data representation and exchange – Part 42: Geometric and topological representation.* ISO TC184/SC4; 2000

[24] Pratt, M.J.: *A hybrid feature-based modelling system.* In: Proceedings of the IFIP WG 5.2/GI International Symposium on Advanced Geometric Modelling for Engineering Applications, West Berlin, Germany, November 1989, pp. 189–201. IFIP/GI 1990

[25] Hoffmann, C.M.: *Geometric and solid modeling*. San Mateo, CA: Morgan Kaufmann, 1989

[26] Lee, K.: *Principles of CAD / CAM / CAE Systems.* Addison-Wesley, 1999

[27] Foley, J.D.; van Dam, A.; Feiner, S.K.; Hughes, J.F.: *Computer Graphics: Principles and Practice (2nd Ed.).* Addison Wesley, 1990

[28] Schroeder, W.; Martin, K.; Lorensen, B.: *The Visualization Toolkit – An Object-Oriented Approach To 3D Graphics (3rd Ed.)*. Kitware, Inc. Publishers, 2001

[29] Lüddemann, J.: Virtuelle Tonmodellierung. Dissertation, Technical University of Berlin, 1996

[30] Shah, J.J.: *Conceptual Development of form features and feature modelers.* In: Research in Engineering Design (1991) 2; Springer-Verlag, New York, 1991, pp. 93–108

[31] van Houten, F.J.A.M.: *PART: A computer Aided Process Planning System*. Dissertation an der Universität Twente 1991.

[32] Klauck, C.; Bernardi, R.; Legleitner, R.: *Feat-rep: Representing features in CAD/CAM*. Konferenz-Einzelbericht: International Symposium on Artificial Intelligence. Cancum, Mexico, November 13-15, 1991 (1991) Balderas: Editorial Limusa, Seite 245-251, ISBN 968-18-4133-6.

[33] Brun, J.M.: *From Characteristic Shapes to Form Features*. In: Proceedings of the IFIP international conference "Feature modeling and recognition in advanced CAD/CAM-Systems", Valenciennes, Mai 1994, pp. 315–326.

[34] Weber, C.: *What do we call a "feature" and what is ist use? – Results of FEMEX working group I "Feature Definition and Classification"*. International Symposium on Tools and Methods for Concurrent Engineering 1996 (TMCE 96), Budapest/Hungary 29.–31.05.1996. Proceedings pp. 377–385

[35] Ovtcharova, J.: *A framework for feature-based product design – Fundamental principles, system concept, applications*. Fortschrittsbericht VDI Reihe 20 Nr. 241, VDI-Verlag, Düsseldorf, 1997

[36] Podehl, G.; Vajna, S.: *Durchgängige Produktmodellierung mit Features.* CAD-CAM Report Nr. 3 (1998); S. 48–53

[37] Weber, Chr.: *Neue VDI-Richtlinie über Feature-Technologie.* In: Konstruktion 51 (1999), Springer Verlag, Berlin, 1999; S. 31–32

[38] Verein Deutscher Ingenieure: VDI-Richtlinie 2218: *Informationsverarbeitung in der Produktentwicklung – Feature-Technologie*. VDI-Gesellschaft Entwicklung Konstruktion Vertrieb, Düsseldorf, März 2003

[39] Shah, J.J.: *Philosophical Development of Form Feature Concepts and Functional Requirements*. In: Proceedings of the CAM-I Features Symposium, Boston, August 1990.

[40] Salomons, O.W.; van Houten, F.J.A.M.; Kals, H.J.J.: *Review of Research in Feature-Based Design*. In: Journal of Manufacturing Systems Volume 12/No.2, 1993.

[41] Rieger, E.: *Semantikorientierte Features zur kontinuierlichen Unterstützung der Produktgestaltung*. Dissertation TU-Berlin, Reihe Produktionstechnik Bd. 158. Hanser Verlag, München, 1995.

[42] Shah, J.J.: *Understanding parametric modelling terminology: Parametric vs. variational; procedural vs. declarative.* Technical Brief ASU/DAL/engen 95-01. Design Automation Laboratory, Arizona State University, Tempe, AZ 85287-6106, October 1995

[43] Owen, J.C.: *Algebraic solution for geometry from dimensional constraints.* In: J. Rossignac and C. Turner (Eds.): Proceedings of the Symposium on solid Modelling Foundations and CAD/CAM Applications, pp. 397–407, ACM New York 1991

[44] Verroust; Schonek, F.; Roller, D.: *Rule-oriented method for parameterized computer aided design.* Computer-Aided Design 24(10):531–540, October 1992

[45] Jampel, M.; Freuder, E.; Maher, M. (Eds.): *Over-Constraint Systems.* Lecture Notes in Computer Science No. 1106, Springer-Verlag, Berlin, Heidelberg, New York 1996

[46] Bouma, W.; Fudos, I.; Hoffman, Chr.: *A geometric constraint solver.* Report CSD-TR-93-054, Department of Computer Science, Courant Institute, New York, NY 10012

[47] Lee, K.-Y.; Kwon, O-W.; Lee, J.-Y.; Kim, T.-W.: *A hybrid approach to geometric constraint solving with graph analysis and reduction.* In: Advances in Engineering Software 34 (2003) pp. 103–113, Elsevier Science Ltd.; 2003

[48] Frühwirth, T.: *Constraint-Programmierung: Grundlagen und Anwendungen.* Springer-Verlag, Berlin, etc. 1997

[49] Guesgen, H.W.; Herztberg, J.: *A perspective of constraint-based reasoning – An introductory tutorial.* Springer-Verlag, Berlin, Heidelberg, 1992

[50] Spur, G.: *Die Genauigkeit von Maschinen*. Carl Hanser Verlag, München, Wien, 1996

[51] Müller, G.: *Rechnerorientierte Darstellung beliebig geformter Bauteile*. Dissertation TU Berlin 1980, Reihe Produktionstechnik – Berlin, Band 8, Carl Hanser Verlag, München, Wien, 1980

[52] Dokken, T.; Skytt, V.; Ytrehus, A.M.: *The role of NURBS in geometric modelling and CAD/CAM.* In: Proceedings of the IFIP WG 5.2/GI International Symposium on Advanced Geometric Modelling for Engineering Applications, West Berlin, Germany, November 1989, pp. 107–114. IFIP/GI 1990

[53] Dombrowski, P.; Gauss, C.F.: *150 Years after Gauss' „Disquisitiones Generales Circa Superficies Curvas"*. Paris Societe Mathematique de France, 1979

[54] De Boor, C.: *A Practical Guide to Splines*. Springer, New York, 1987

[55] Farin, G.: *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, Boston, 1993

[56] Dierckx, P.: *Curve and Surface Splitting with Splines*. Oxford Clarendon Press, 1995

[57] Salzmann, P.: *Trends der Freiformmodellierung*. CAD/CAM Report. Nr. 1 Januar 2001

[58] Gross, N.: *Applications of Subdivision Techniques in Product Development.* Dissertation, Technical University of Berlin, 2003; http://edocs.tu-berlin.de/diss/2003/gross_nele.pdf; last called: 2004-06-18

[59] Krause, F.-L.; Stiel, Chr.: *Featuremodellierfunktionalität für Freiformgeometrien*. In: VDI-Tagung "Features verbessern die Produktentwicklung", Berlin, pp. 119–134. VDI-Verlag Düsseldorf, 1997

[60] Sha, J.J.; Mäntylä, M.: *Parametric and feature-based CAD/CAM, concepts, techniques, and applications.* John Wiley & Sons, USA, 1995

[61] Kramer, G.A.: *Using degrees of freedom analysis to solve geometric constraint systems.* In: Rossignac, J.; Truner, J. (Eds.), Proceedings of First ACM Symposium on Solid Modelling Foundations and CAD/CAM Applications, June 5–7, ACM Press, 1991, pp. 371–378

[62] Noort, A.; Hoek, G.F.M.; Bronsvoort, W.F.: *Integrating part and assembly modelling.* In: Computer-Aided Design 34 (2002), Elsevier Science Ltd., pp. 899–912

[63] Bloor, M.S.; Owen, J.: *Product Data Exchange.* University College London Press; 1995

[64] ISO TC184/SC4: *ISO 10303 Part 1 IS: Industrial automation systems and integration – Product data representation and exchange – Part 1: Overview and fundamental principles*. ISO TC184/SC4, 1994

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

[66] Eversheim, W.; Baumann, M.; Marczinski, G.; Leber, M.: *STEP als Integrationskern für die Produktdatengenerierung*. In: VDI-Z 135 (1993), Nr. 7, S. 63-66

[67] ISO TC184/SC4: *ISO 10303: STEP Overview*. http://www.tc184-sc4.org/SC4_Open/SC4_Work_Products_Documents/STEP_(10303)/; last called: 2004-03-30

[68] ProSTEP iViP Association: *STEP Portal Starting Point.* http://www.prostep.org/en/stepportal/ausgangssituation/; last called: 2004-03-30

[69] SCRA: *ISO 10303 STEP application handbook version2.* SCRA, Charleston, USA 2001;Available at https://www.uspro.org/; last called: 2004-03-30

[70] Kemmerer, S. J.: *STEP – The Grand Experience*. National Institute of Standards and Technology, Special Publication 939, U.S. Washington, July 1999

[71] Mäntylä, M.; Opas, J.; Puhakka, J.: *A prototype system for generative process planning of prismatic parts.* In: Kusiak, A. (editor), Modern Production Management Systems, Proc. AMPS '87; North-Holland Publ. Co. Amsterdam, 1987, pp. 599–611

[72] Krause, F.-L.; Ulbrich, A.; Vosgerau, F.H.: *Feature based approach for the integration of design and process planning systems.* In: IFIP 5.2 Workshop on Geometric Modelling, Rensselaerville, NY, June 17–21, 1990, North-Holland

[73] Solano, L.; Brunet, P.: *Constructive constraint-based model for parametric CAD systems.* In: Computer-Aided Design 26 (1994) 8, Butterworth-Heinemann Ltd.; 1994

[74] Shah, J.J.: *Procedural languages for geometry definition: ASU testbed procedural language.* Technical Brief ASU/DAL/engen 95-02. Design Automation Laboratory, Arizona State University, Tempe, AZ 85287-6106, October 1995

[75] Wilson, P.R.; Faux, I.D.; Ostrowski, M.C.; Pasquill, K.G.: *Interfaces for data transfer between solid modeling systems.* IEEE Computer Graphics and Applications 5 (1985) 1; pp. 41–51

[76] ISO TC184/SC4: *ISO 10303 Part 48: Industrial automation systems and integration – Product data representation and exchange – Part 48: Form features.* ISO TC184/SC4; 1992

[77] ISO TC184/SC4: *ISO 10303 Part 214 IS: Industrial automation systems and integration – Product data representation and exchange – Part 214: Core data for automotive mechanical design processes.* ISO TC184/SC4; 2001

[78] ISO TC184/SC4: *ISO 10303 Part 224 2nd Edition IS: Industrial automation systems and integration – Product data representation and exchange – Part 224: Mechanical product definition for process planning using machining features.* ISO TC184/SC4; 2000

[79] ISO TC184/SC4: ISO 10303 Part 111 WD: *Industrial automation systems and integration – Product data representation and exchange – Part 111: Construction history features.* ISO TC184/SC4/WG12 N2006; 2003

[80] Hoffmann, C.M.; Juan, R.: *EREP: an editable high-level representation for geometric design and analysis.* In: Wilson, P.R.; Wozny, M.J.; Pratt, M.J. (editors): Geometric modeling for product realization. North-Holland Publishing Co., 1992

[81] Krause, F.-L.; Kramer, S.; Rieger, E.: *PDGL – A Language for Efficient Feature Based Product Gestaltung*; Cirp Annals, Vol. 40,1, pp. 135–138, 1993

[82] Tönshoff, H.K.; Aurich, J.C.; Baum, T.: *Configurable feature-based CAD/CAPP system*. In: Proceedings of the IFIP international conference "Feature modeling and recognition in advanced CAD/CAM-Systems", Valenciennes, Mai 1994, pp. 757–771

[83] Tönshoff, H. K.; Aurich, J. C.; Hamelmann, S.: *Formale Elementbeschreibung für Konstruktion und Arbeitsplanung*; VDI-Z, 135, Nr.11/12, 1993

[84] Tönshoff, H. K.; Krause, F.-L.; Baumann, R.; Woelk, P.-O.: *On the Implicit Exchange of Feature-based Product Model Data. Production Engineering – Research and Development.* Vol. VI/1, pp. 125–128. WGP, Berlin, 1999

[85] Mun, D.; Han, S.; Kim, J.; Oh, Y.: *A set of standard modeling commands for the history-based parametric approach.* Computer Aided Design 35 (2003), pp. 1171–1179; Elsevier Ltd. 2003

[86] Anderson, W.; Ansaldi, S.: *ENGEN data model: a neutral model to capture design intent*. In: Jacucci, G.; Olling, G.J.; Preiss, K.; Wozny, M.J. (editors): CDROM Proceedings of IFIP PROLAMAT'98 Conference, Trento, Italy, September 1998, Kluwer Academic Publishers, 1998

[87] Shah, J.J.; Khan, N.; Solkhan, S.: *Geometric constraint specification in declarative modelling.* Technical Brief ASU/DAS/engen 95-03. Design Automation Laboratory, Arizona State University, Tempe, AZ 85287-6106, October 1995

[88] ISO TC184/SC4: *ISO 10303 Part 108 CD: Industrial automation systems and integration – Product data representation and exchange – Part 108: Parameterization and constraints for explicit geometric product models.* ISO TC184/SC4/WG12 N940; 2001

[89] Pratt, M.J.; Anderson, B.D.: *A shape modelling applications programming interface for the STEP standard.* In: Computer-Aided Design 33 (2001), Elsevier Science Ltd., 2001, pp. 531–543

[90] Bettig, B.; Shah, J.: *Derivation of a standard set of geometric constraints for parametric modeling and data exchange.* In: Computer-Aided Design 33 (2001), Elsevier Science Ltd.; 2003, pp. 17–33

[91] Choi, G.H.; Mun, D.; Han, S.: *Exchange of CAD part models based on the macro-parametric approach.* International Journal of CAD/CAM 2 (2002) 2, pp. 13–21. http://www.ijcc.org, last called 2004-03-03

[92] Aurich, J.C.: *Werkstückmodellierung mit Technischen Freiformelementen.* Dr.-Ing. Dissertation, Universität Hannover, VDI Verlag, Reihe 20, Nr. 183, 1995

[93] Tönshoff, H.K.; D'Agostino, N.: *Technische Elemente für die Modellierung von Bauteilen mit Freiformgeometrie.* In: VDI-Tagung “Features verbessern die Produktentwicklung”, Berlin, pp. 99–118. VDI-Verlag Düsseldorf, 1997

[94] Lee, K.; Gossard, D.C.: *A hierarchical data structure for representing assemblies: part I.* In: Computer-Aided Design 17 (1985) 1; pp. 15–29; Elsevier Science Ltd.; 1985

[95] Henson, B.W.; Baxter, J.E.; Juster N.P.: *Assembly representation within a product data framework.* In: Advances in Design Automation 1993; vol 1. New York: ASME, 1993; 10–22 Sept. 1993, pp. 195–205

[96] Baxter, J.E.; Juster, N.P.; de Pennington A.: *A functional data model for assemblies used to verify product design specifications.* Proceedings IMechE, Part B, IMechE 1994; 208(B4); pp. 235–244; ISSN/ISBN 095-4054

[97] Sugimura, N.; Moriwaki, T.; Kakino, T.: *Study on assembly model based on STEP and its application to assembly process planning.* In: Proceedings of the 1996 Japan–USA Symposium on Flexible Automation, ASME 1996; pp. 791–794

[98] Oh, Y.; Han, S.-H.; Suh, H.: *Mapping product structure between CAD and PDM systems using UML.* In Computer-Aided Design 33 (2001), Elsevier Science Ltd. 2001, pp. 521–529

[99] Zha, X.F.; Du, H.: *A PDES/STEP-based model and system for concurrent integrated design and assembly planning.* In: Computer-Aided Design 34 (2002) pp. 1087–1110, Elsevier Science Ltd., 2002

[100] Heissemann, J.; Mattikalli, R.: *Representing relationships in hierarchical assemblies.* In: CD-ROM Proceedings of the 1998 ASME Design Engineering Technical Conferences and Computers in Engineering Conference; ASME, 1998

[101] ISO TC184/SC4: ISO 10303 Part 11 IS: *Industrial automation systems and Integration – Product data representation and exchange – Description Methods: The EXPRESS language reference manual*; ISO; 1994

[102] ISO TC184/SC4: ISO 10303 Part 21 FDIS edition 2: *Industrial automation systems and integration – Product data representation and exchange – Part 21: Clear text encoding of the exchange structure.* ISO TC184/SC4; 2001

[103] ProSTEP-iViP Association: *STEP description methods – EXPRESS, EXPRESS-G, EXPRESS-I, EXPRESS-X.* http://www.prostep.org/en/stepportal/was/beschreibung/. last called: 2004-05-28

[104] ISO TC184/SC4: ISO 10303 Part 22 FDIS : *Industrial automation systems and integration – Product data representation and exchange – Part 22: Standard Database Access Interface (SDAI).* ISO TC184/SC4; 1997

[105] STEP Tools, Inc.: *Introduction to Standard Data Access Interface (SDAI).* http://www.steptools.com/library/standard/sdai.html; last called: 2004-05-28

[106] World Wide Web Consortium: *Extensible Markup Language (XML) 1.1.* http://www.w3.org/TR/xml11/, W3C Recommendation 04 February 2004; last called: 2004-05-28

[107] Microsoft: *Understanding XML.* http://msdn.microsoft.com/XML/Understanding/default.aspx?pull=/library/en-us/dnxml/html/understxml.asp; last called 2004-05-28

[108] Goldfarb, C.; Prescod, P.: *The XML Handbook.* In: C. Goldfarb (Ed.), Open Information Management, Prentice Hall, Upper Saddle River, NJ, 1998. ISBN 0-13-081152-1.

[109] Coverpages.org: *Product Data Markup Language (PDML).* Technology Reports, http://xml.coverpages.org/pdml.html, Last modified: June 22, 1999; last called: 2004-05-28

[110] Burkett, W.C.: *Product data markup language: a new paradigm for product data exchange and integration.* In Computer-Aided Design 33 (2001), Elsevier Science Ltd. 2001, pp. 489–500

[111] Roller, D.; Ruess, H.: *An approach to an open CAD system architecture.* In: Proceedings of the IFIP WG 5.2/GI International Symposium on Advanced Geometric Modelling for Engineering Applications, West Berlin, Germany, November 1989, pp. 365–378. IFIP/GI 1990

[112] CAM-I: *Application Interface Specification (AIS), Version 2.1*. Technical Report R-94-PM-01, Consortium for Advanced Manufacturing International, Inc., Bedford, TX, 1994

[113] Pavey, S.G.; Hailstone, S.R.; Pratt, M.J.: *An automated interface between CAD and process planning*. In: McGeough, J.A. (editor): Computer aided production engineering. Mechanical Engineering Publications Ltd, Bury St Edmunds, UK, 1986

[114] Schlechtendahl, E.G.: *Transporting geometric models. Results of the CAD*I project.* In: Proceedings of the IFIP WG 5.2/GI International Symposium on Advanced Geometric Modelling for Engineering Applications, West Berlin, Germany, November 1989, pp. 143–155. IFIP/GI 1990

[115] Middleditch, A.E.; Reade, C.M.P.: *A kernel for geometric features*. In: Hoffman, C.M.; Bronsvoort, W.F. (editors): Proceedings of the Fourth ACM Symposium on Solid Modeling and Applications, Atlanta, GA, May 1997, New York: Association for Computing Machinery, 1997, pp. 131–140

[116] Middleditch, A.E.; Reade, C.M.P.; Gomes, A.J.: *Set-combinations of mixed-dimension cellular objects of the Djinn API*. Computer Aided Design 31 (1999) 11, Elsevier Science ltd. 1999, pp. 683–694

[117] Armstrong, C.; Bowyer, A.; Cameron, S.; Corney, J.; Jared, G.; Martin, R.; Middleditch, A.; Sabin, M.; Salmon, J.; Woodwark, J.R.: *Djinn: specification and report.* Winchester, UK: Information Geometers Ltd, 2000

[118] Design and Modeling Advisory Council: *What is OLE for Design and Modeling?* At http://www.dmac.org/

[119] ISO TC184/SC4: ISO 13584 IS: *Industrial Automation Systems and Integration – Parts Library*. ISO TC184/SC4; 1998

[120] Open CADCADE S. A.: *Open CASCADE Technology, 3D modeling & numerical simulation*. http://www.opencascade.org/, last called: 2004-05-19

[121] Object Management Group: *CAD Services 1.1 RTF Report – Updated Convenience document*. http://www.omg.org/cgi-bin/apps/do_doc?dtc/03-01-05.pdf, last called: 2004-03-24; OMG, 2003

[122] Object Management Group: *CORBA 3.0 Specification.* http://www.omg.org/technology/documents/formal/corba_2.htm; 2004; last called: 2004-05-28

[123] World Wide Web Consortium: *SOAP Version 1.2 Part 0: Primer.* W3C Recommendation 24 June 2003; http://www.w3.org/TR/2003/REC-soap12-part1-20030624/#intro; last called: 2004-05-28

[124] Kappel, G.; Pröll, B.; Reich, S.; Retschitzegger, W. (Hrsg.): *Web Engineering – Systematische Entwicklung von Web-Anwendungen.* Dpunkt Verlag, 2004

[125] Dumke, R.; Lother, M.; Wille, C.; Zbrog, F.: *Web Engineering*. Pearson Studium, 2003

[126] PDTnet: *Product Data Technology and Communication in the Network of Automotive Manufacturers and Suppliers.* http://www.pdtnet.org/en/; last called: 2004-05-06

[127] Krause, F.-L.; Tang, T.; Ahle, U. (Hrsg.): *Leitprojekt integrierte Virtuelle Produktentstehung (iViP) – Fortschrittsbericht II.* Fraunhofer IRB Verlag, Stuttgart, März 2002

[128] Object Management Group: *Model driven architecture – MDA, frequently asked questions.* http://www.omg.org/mda/mda_files/MDAFAQfinal1.pdf; last called: 2004-05-05

[129] Chen, X.P.; Hoffmann, Chr. M.: *On editability of feature-based design.* Computer-Aided Design 27 (1995) 12; pp. 905–914

[130] Capoyleas, V.; Chen, X.P.; Hoffmann, Chr. M.: *Generic naming in generative, constrain-based design.* In: Computer-Aided Design 28 (1996) 1, pp. 17–26

[131] Raghothama, S.; Shapiro, V.: *Boundary representation deformation in parametric solid modelling.* ADM Trans on Computer Graphics 17 (1998) 4; pp. 259–286

[132] Wu, J.; Zhang, T.; Zhang, X.; Zhou, Ji.: *A face based mechanism for naming, recording and retrieving topological entities.* In: Computer-Aided Design 33 (2001), Elsevier Science Ltd.; 2001, pp. 687–698

[133] Kripac, J.: *A mechanism for persistently naming topological entities in history-based parametric solid models.* In: Computer-Aided Design 29 (1997) 2; pp. 113–122

[134] ProSTEP Association: *Report on the 7th Benchmark held by ProSTEP.* HTML Publication is available to members only; ProSTEP Association, Darmstadt, November 2001

[135] ProSTEP Association: *Report on the 8th Benchmark held by ProSTEP.* HTML Publication for Members; ProSTEP Association, Darmstadt, September 2003

[136] Krause, F.-L.; Baumann, R.: *Austausch parametrischer Informationen auf der Basis impliziter Produktbeschreibungen.* In: VDI-Tagung “Features verbessern die Produktentwicklung”, Berlin, pp. 215–232. VDI-Verlag Düsseldorf, 1997.

[137] Stiel, Chr.: *Featurebasiertes Gestalten von Produkten mit Freiformgeometrien.* In: Berichte aus dem Produktionstechnischen Zentrum Berlin, Fraunhofer IPK, Berlin 1998

[138] Piegl, L.; Tiller, W.: *The NURBS book.* Springer Verlag, Berlin, Heidelberg 1995

[139] Autodesk: *Autodesk Inventor 7 online reference manual*. Part of online product decumentation, 2003

[140] Baum, T.: *Grafisch-interaktive Arbeitsplanung mit Technischen Elementen.* Fortschritt-Berichte VDI, Reihe 20, Nr. 250, VDI-Verlag, 1997

[141] Ulbrich, A.: *Featureintegrierte Fertigungsplanung*. In: Berichte aus dem Produktionstechnischen Zentrum Berlin, IPK Berlin, TU-Berlin, 1996

[142] Krause, F.-L.; Baumann, R.; Dreher, S.: *Wissensbasierte Schweißplanung mit Fuzzy-Logik*. In: Schweißen und Schneiden 52 (1999) 6, DVS Verlag Düsseldorf

[143] Leemhuis, H.; Baumann, R.; Kaufmann, U.; Swoboda, F.; Kühn, T.; Ragan, Z.: *Function Oriented Product Modelling Based on Feature Technology and Integrated Constraint Management.* In: Proceedings PDT Europe 2002 Executive Summary, Turin, Italy 7th–9th May 2002, S. 173–180

[144] N.N.: *Xerces: XML parsers in Java and C++ (plus Perl and COM).* http://xml.apache.org/xerces-c/index.html; last called: 2004-03-11

[145] N.N.: *Project: dom4j: flexible XML framework for Java: Summary*. At Sourceforge.net, http://sourceforge.net/projects/dom4j; last called: 2004-03-11

[146] IONA Technology: *Orbacus ™ your CORBA source.* IONA Technology 2004, at http://www.orbacus.com, last called: 2004-03-11

[147] Shin, S.: *Java Technology and Web Services*. At http://java.sun.com/webservices/, Sun Microsystems, Inc. 1994-2004, last called: 2004-03-11

Annex – Proposal of Additional Functionality for OMG CAD Services Specification

The following classes and methods are proposed for future versions of the OMG CAD Services specification. The proposal comprises basic functionality necessary to establish structure-oriented exchange of feature-based CAD models.

```
Class CadServer
CadSystem connect (String CadSystemName, String pw, ...);

Class CadSystem
CadModel         openModel           (String modelName);
ModelCastFactory getModelCastFactory ();

Class CadModel
Node     getRootNode    ();
Assembly createAssembly (String name, ...);
Part     createPart     (String name, ...);

Class Node
boolean isAssembly ();
boolean isPart     ();
boolean isFeature  ();

Class ModelCastFactory
Assembly castToAssembly (Node);
Part     castToPart     (Node);
Feature  castToFeature  (Node);

Class Assembly
AssemblyNode getRootNode    ();
Assembly     createAssembly (String name, ...);
Part         createPart     (String name, ...);

Class AssemblyNode
boolean      isAssembly    ();
boolean      isPart        ();
AssemblyNode getLeftChild  ();
AssemblyNode getRightChild ();
```

```
Class Part
PartCastFactory getPartCastFactory ();
Feature         getRootNode        ();
Feature         getLeftChild       ();
Feature         getRightChild      ();

Feature performOperation (Feature f1, Feature f2,
                          Type operationType, ...);

Feature  createFormFeature (String name, Type type, ...);
Face[]   getFaces          ();
Edge[]   getEdges          ();
Vertex[] getVertices       ();

Class Feature
Boolean isOperationFeature ();
Boolean isFormFeature      ();
String  getName            ();

Class PartCastFactory
OperationFeature castToOperationFeature (Feature);
FormFeature      castToFormFeature      (Feature);

Class OperationFeature
boolean IsAddOperation       ();
boolean IsSubtractOperation  ();
boolean IsIntersectOperation ();
Feature getFirstFeature      ();
Feature getSecondFeature     ();

Class FormFeature
FormFeatureCastFactory getFormFeatureCastFactory ();
boolean                isCylinder                ();
boolean                isBlend                   ();
boolean                isChamfer                 ();
boolean                commit                    ();
Face[]                 getFaces                  ();
Edge[]                 getEdges                  ();
Vertex[]               getVertices               ();

Class FormFeatureCastFactory
Cylinder castToCylinder ();
Blend    castToBlend    ();
Chamfer  castToChamfer  ();

Class Blend
Edge[] getBlendedEdges();

Class Chamfer
Edge[] getChamferedEdges ();
```

```
Class Cylinder
CylinderParameter getClinderParameter  ();
void              setCylinderParameter (CylinderParameter p);

Class CylinderParameter
double getRadius ();
Void   setRadius (double r);

Class ExtrudedParameter
CurveCastFactory  getCurveCastFactory ();
Curve[]           getCurves           ();
void              setCurve            (Curve[] curves);

Class Edge
CurveCastFactory  getCurveCastFactory ();
Curve             getCurve            ();

Class Curve
boolean isLine();

Class CurveCastFactory
CircularCurve castToCircularCurve (Curve c);

Class CircularCurve
Point3D[]  Get3Points     ();
Point3D    getStartPoint  ();
Point3D    getEndPoint    ();
Point3D    getMiddlePoint ();
```