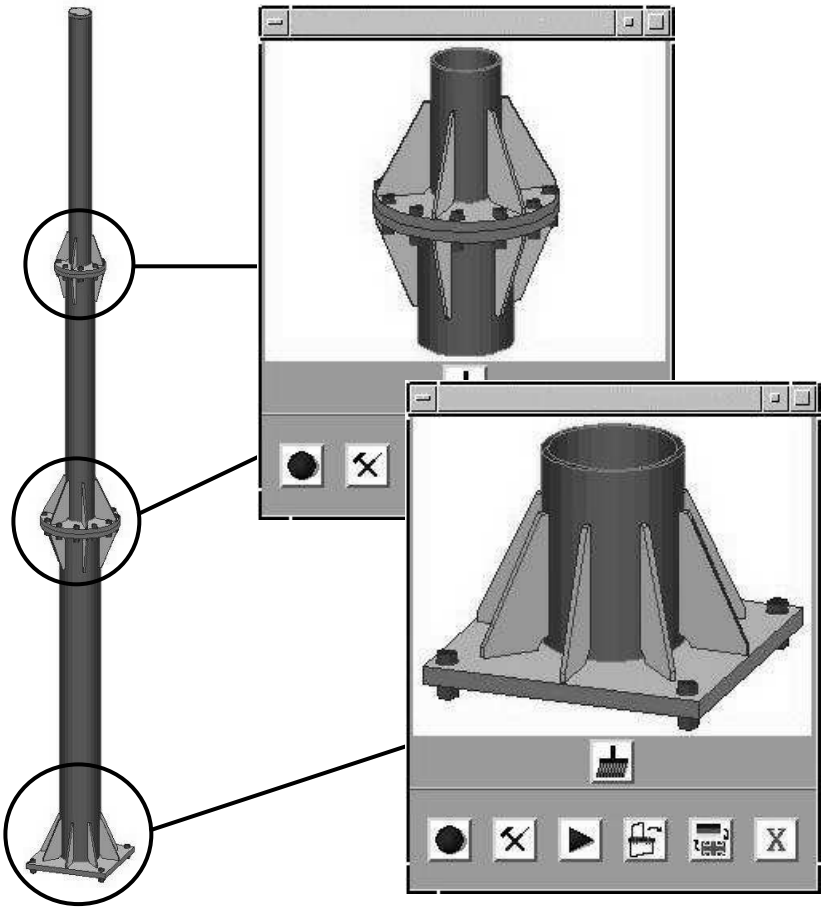
Theoretische Methoden
für systematisches Konstruieren



Dissertation zur Erlangung des Grades Doktor-Ingenieur
des Fachbereichs Bauingenieurwesen der Bergischen Universität Wuppertal

von
Tamás Vadas

Wuppertal 2003

Dissertation eingereicht am: 20.05.2003

Tag der mündlichen Prüfung: 16.12.2003

1. Gutachter: Herr Univ.-Prof. Dr.-Ing. Georg Pegels

2. Gutachter: Herr apl. Prof. Dr.-Ing. Dr. h.c. (SK) Dietrich Hoeborn

Zusammenfassung

In der vorliegenden Arbeit werden die Grundlagen der CAD-Methoden für die systematische Konstruktion und die Aufbauobjekte der Nutzoberfläche erforscht.
Als Zielsetzung wird die Unterstützung eines schnelleren und effizienteren Konstruierens unter Berücksichtigung der Mehrsprachigkeit festgelegt. Dazu wird ein länderunabhängiger Modul, Logikrekorder genannt, mit dem der Konstrukteur mehrfach anwendbare Konstruktionsmethoden selbst in seiner eigenen Sprache erzeugen wird, ingenieurwissenschaftlich entwickelt.
Die Ergebnisse werden ihre Bestätigung im Praxiseinsatz finden.

Summary

In the present work the basis of CAD methods for the systematic construction and the constructional objects of the application surface are explored.
As objective, the support of faster and more efficient engineering with consideration of the multilingualism is specified. A language independent module, named logic-recorder, with which the technical designer will create multiply applicable structural construction methods himself, in his own language, is developed engineer-scientifically.
The results will find their confirmation in practical application.

Résumé

Dans ce travail, les bases des méthodes CAD pour la construction systématique et les sujets de construction de la surface d'utilisation ont été examinés.
Le soutien du bâtit plus rapide et plus efficace est visé en considérant le multilinguisme. Dow arriver à de but, un module – nommé le recorder logique – a été développé à la façon ingénieur- scientifique. Avec ce module c’est le constructeur lui-même qui pourra produire des méthodes de construction à différentes reprises applicables dans sa propre langue.
Les résultats se confirmeront dans les pratiques suivantes.

Vorwort

An dieser Stelle möchte ich mich bei Herrn Univ.-Prof. Dr.-Ing. Georg Pegels bedanken, der mich stets mit Ideen und Ratschlägen bei meiner Arbeit unterstützte, und mich bei ingenieurwissenschaftlichen Problemen hilfreich beriet. Weiterhin möchte ich mich bei Herrn Dr.-Ing. Heinz-Dieter Koch bedanken, der mir bei allen Fragen zum Konstruktionsprogramm *bocad* beratend zur Seite stand. Hiermit möchte ich von ihm zitieren: „Wissen über Konstruieren und Darstellen ist kaum in Büchern, sondern in den Köpfen erfahrener Ingenieure und Konstrukteure gespeichert."

Ich bedanke mich bei meiner Frau Patricia, mit der ich mich in stilistischen Fragen beratschlagen konnte. Mein Dank gilt ebenso Herrn Dipl.- Ing. Krisztián Hegedűs, der mir mit seiner Hilfsbereitschaft und seiner richtungweisenden Kritik eine große Hilfe war. Ferner danke ich allen meinen Kolleginnen und Kollegen, die mich mit ihrer freundlichen Art ermutigend unterstützt.

Inhaltsverzeichnis

1 Hintergründe

1.1 Verbindung des Bauwesens mit der Informatik

1.1.1 Begriff der Bauinformatik

Der in den Lexika noch nicht definierte Begriff Bauinformatik bezeichnet ein in den letzten Dekaden entstandenes Fachgebiet, das aus zwei Bereichen der Ingenieurwissenschaft - dem Bauingenieurwesen und der Informatik - hervorgegangen ist. Die deutschen Professoren für Bauinformatik haben nach [5] die Bauinformatik wie folgt definiert:

Die Bauinformatik „befasst sich mit den wissenschaftlichen Grundlagen beim Einsatz der Informations- und der Kommunikationstechnik und mit der systematischen Weiterentwicklung von Berechnungs- und Simulationsverfahren im Bauwesen."

Sie ist strukturiert nach den folgenden Bereichen:

- Technik der Bauinformatik
- Methoden der Bauinformatik
- Modelle der Bauinformatik
- Prozesse der Bauinformatik

1.1.2 Bedeutung der Bauinformatik

Weiter wird in [5] ausgeführt: „Die Einführung der Computer als multifunktionale Arbeitsplätze im Bauwesen war mit grundlegenden Veränderungen verbunden." „Die Bauinformatik ist eine Grundlage des modernen Bauingenieurwesens und befasst sich mit der Anwendung und Weiterentwicklung der Computerwissenschaft im Bauwesen." So wird auch die Erleichterung der Arbeit des Bauingenieurs im CAD-Bereich mit Hilfe der Werkzeuge der Informatik gelöst. CAD-Programme sind hier Konstruktionsprogramme für die Modellierung und Darstellung von Bauwerken.

„Die Nutzung moderner Computertechnologie ist ein integraler Bestandteil der Bauwirtschaft und der Baupraxis. Bei der Einführung und Umsetzung neuer Techniken und Methoden zur rechnerunterstützten Bearbeitung von Bauvorhaben ist die Bauinformatik unverzichtbar. Ihr kommt damit eine Schlüsselrolle bei der Entwicklung geeigneter Software für die Baupraxis entsprechend dem neuesten Stand der Computerwissenschaften zu. “

1.2 Zielsetzung der Bauinformatik

Pegels definiert in [4] die Ziele der Bauinformatik betont anwendungsorientiert und praxisnah:

„Verschärfter globaler Wettbewerb und anspruchsvollere Bauweisen führen zu Marktentwicklungen von grundsätzlicher Bedeutung für den Bauindustrie-Standort Deutschland. Der schwierigere Markt stellt schwachen Unternehmen fundamentale Risiken. Leistungsstarken Büros und Unternehmen, die anspruchsvolle Bauten schlüsselfertig in kürzester Zeit fehlerfrei und kostengünstig liefern können, bietet er lebhafte Chancen.“

Dazu sind geeignete Werkzeuge neuester Informationstechnologie in Form objektorientierter Datenbanksysteme und Multimedia-Kommunikationstechnik notwendig. Ausgehend von dem bereits heute erreichten technischen Niveau sollen daher folgende Softwarebausteine für die Bauindustrie entwickelt und in Pilotanwendungen erprobt werden:

1. Eine objektorientierte Projektdatenbank, die im vollen Anforderungsprofil der Bauindustrie gleichzeitiges, verteiltes Arbeiten widerspruchsfrei bei hinreichender Schnelligkeit unterstützt.
2. Qualitätsgesicherte, automatisierte Verfahren zur gleichzeitigen Projektbearbeitung, auch von verschiedenen Orten aus, unter Nutzung von Internet- und Intranetverbindungen als Netz.
3. Verfahren für mitlaufenden Informationsaustausch zwischen technischer und

kaufmännischer Bearbeitung.

4. Komplettierung von Bauingenieur-CAD-Systemen um Architekturelemente, damit umfassend von Bauantragszeichnungen bis zu Werkstattplänen durchgängig ohne Systembruch mit einem statt - wie heute üblich - mit mehreren inkompatiblen Systemen gearbeitet werden kann.
5. Entwicklung von CAD-Konstruktionsmethoden (Makros) für das effiziente Entwerfen und Detaillieren anspruchsvoller Bauwerkshüllen in Stahl- und Glasarchitektur, individueller Treppenarchitektur und Metallbau mit modernen Zulieferteilen.
6. Effiziente Projektdurchführung wegen der mehrfachen Makro-Verwendbarkeit (auch in anderen Projekten) und schnelle Neubearbeitung bei Änderungen in laufenden Projekten durch die vom Programmanwender selbst nach Bedarf erstellten Anwender-Makros, siehe *Kapitel 5.*
7. Anforderungen der Mehrsprachigkeit und die Auswahlmöglichkeit der verschiedenen nationalen Normen in dem Programm wegen der Globalisierung und der Verbreiterung der Märkte.

1.3 Ziel dieser Arbeit

1.3.1 Aufbau von CAD-Systemen

Konstruktionsprogramme sind grundsätzlich schalenförmig aufgebaut. Der Kern ist das - z. B. als objektorientierte Datenbank realisierte - dreidimensionale, geometrische Volumenmodell. Es wird durch weitere Eigenschaften zum Produktmodell. Die Methoden, die das konstruktive Wissen bilden, hüllen diesen Kern ein. Die Konstruktionsanschlüsse werden durch CAD-Methoden automatisch erzeugt. Der Benutzer sieht das Ergebnis der Programmierung anschließend an der Benutzungsoberfläche (im weiteren Verlauf „Nutzoberfläche“ genannt, nach Pegels). Siehe *Abbildung 1-1*

.

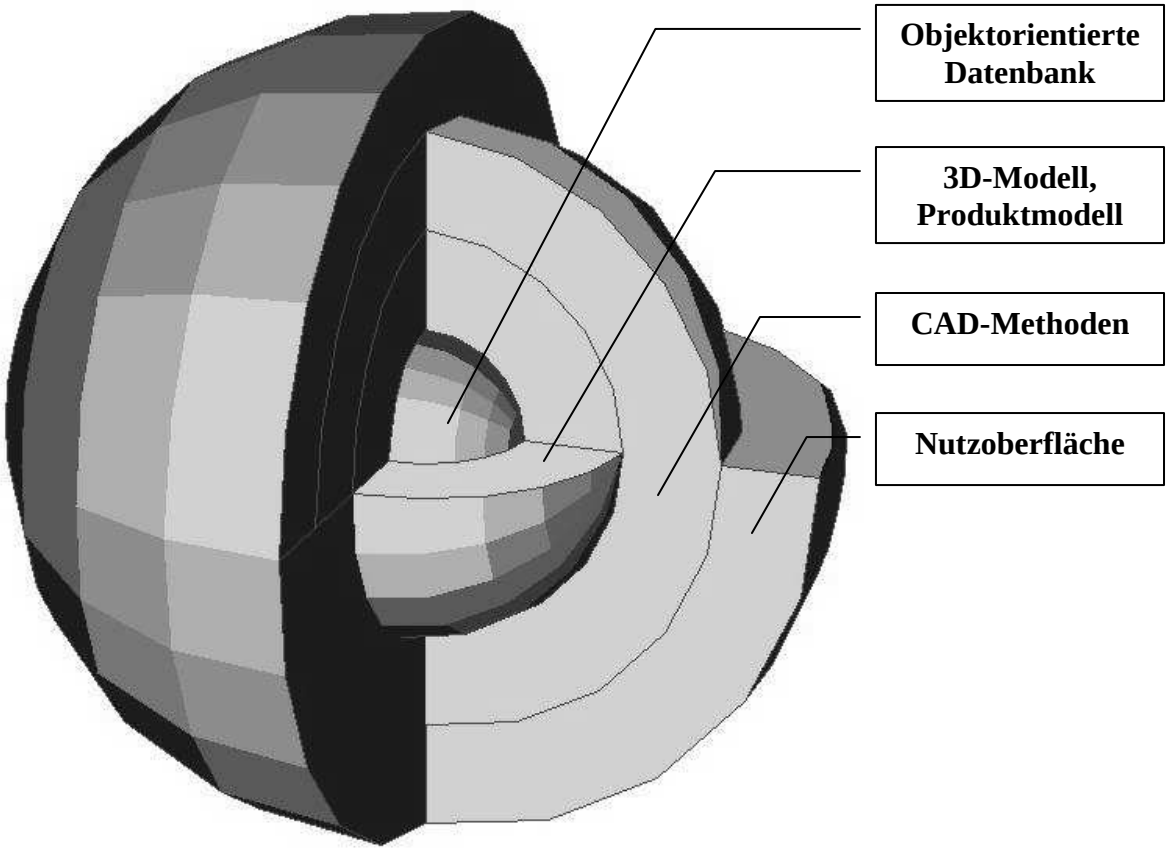


Abbildung 1-1: Aufbau von CAD-Systemen

Zeitgemäße CAD-Systeme müssen so aufgebaut sein, dass sie unabhängig von Plattformen – z. B. sowohl unter Windows als auch unter Linux - laufen. Linux hat den Vorteil, dass die Benutzung dieses Betriebsystems kostenlos ist. Windows hingegen ist in der Bauindustrie wesentlich stärker verbreitet.

1.3.2 Vorgehensweise dieser Arbeit

In dieser Arbeit werden zuerst die Grundlagen der CAD-Systeme aus Sicht der Informatik und des Bauingenieurwesens erklärt, auf denen nach folgend aufgebaut wird. In dem zweiten Teil werden die erarbeiteten, neuen Forschungsergebnisse - die oben genannten CAD-Methoden für Anwender-Makros - behandelt.

Anhand der in der *Abbildung 1-1* dargestellten zwei oberen Schalen werden zunächst die Grundlagen erläutert:

- Basiselemente für individuelle CAD-Konstruktionsmethoden (*Kapitel 2*) und
- Werkzeuge für Nutzoberflächen *(Kapitel 3*).

Den aktuellen Stand der CAD-Systeme - untersucht an dem für diese Arbeit zur Verfügung gestellten CAD-System BOCAD-3D der bocad Software GmbH in Bochum - bilden systemeigene Methoden (Makros), die im nächsten Schritt behandelt werden:

- Theoretische Methoden der Konstruktion für neue Märkte (*Kapitel 4.1*) und
- Redesign von Konstruktionsmethoden (*Kapitel 4.2*).

Die diskutierten Werkzeuge dienen zur Erstellung anspruchsvoller CAD-Anwendungen mit automatisierten Makros. Die Anwender sollen mit den Ergebnissen dieser Arbeit die Möglichkeit erhalten, eigene Makros firmenspezifisch zu entwickeln:

- CAD-Methodenrecorder (*Kapitel 5.1*) und
- dessen Weiterentwicklung, der Logikrekorder (*Kapitel 5.2*).

Abschließend wird über die Erkenntnisse reflektiert. Es werden Prognosen über mögliche und wünschenswerte Zukunftsperspektiven aufgestellt:

- Zusammenfassung und Ausblick (*Kapitel 6*)

2 Basiselemente für individuelle CAD-Konstruktionsmethoden

2.1 Programmiersprachen

Bevor die Einzelheiten und Besonderheiten der CAD-Entwicklungswerkzeuge näher erläutert werden, sind die wichtigsten Programmiersprachen vorzustellen und einzuordnen.

Die Klassifikation der Programmiersprachen:

- Maschinenorientierung: Assembler, Maschinensprache
- Problemorientierung: COBOL, PASCAL, C, Fortran, *FFEIN7*
- Logikorientierung: Prolog, LISP
- Objektorientierung bzw. Objektbasierung: *C*++, Visual Basic, VBA, Delphi, Object COBOL

Von den oben klassifizierten Programmiersprachen werden in diesem Kapitel die Sprachen FFEIN7 und C++ detaillierter erläutert, da sie Basis der experimentellen Nachweise dieser Arbeit sind.

Die Realisierung der Idee des zukunftweisenden Programmmoduls, des Logikrekorders, wird im letzten Teil dieser Arbeit durch Beispiele veranschaulicht. Dabei wird die Programmiersprache FFEIN7 angewendet. FFEIN7 (**F**ormat **F**reie **Ein**gabe Fortran **7**7) ist eine Interpretersprache für das Konstruieren im Bauwesen, die ursprünglich aus Fortran 77 entwickelt wurde.

Die moderne, objektorientierte Programmierung, wie z. B. die Anwendung der C++ Programmiersprache genügt den heutzutage gestellten Ansprüchen. Deshalb werden die Programmiersprache FFEIN7 und C++ verglichen und dadurch wird gezeigt, dass die bei den Beispielen zum Logikrekorder angewandten Programmierungselemente – wie z. B. die Steueranweisungen – auch auf andere, moderne und heutzutage verbreitete Programmiersprachen übertragbar sind.

2.2 Kommentare

Während der Programmierung Erläuterungen und Kommentare in den Quelltext (Source Code) einzufügen ist sehr sinnvoll, um nachträgliche Änderungen, Ergänzungen oder Korrekturen in dem Programm ohne langen Suchvorgang und ohne Verständnisprobleme vornehmen zu können.

Kommentare sind in den Beispielsourcen für die Nutzoberfläche mit /* eingeleitet, sofern es sich um Konfigurationsdateien handelt.

Das Einfügen von Kommentaren wird in der Sprache C++ mit einem doppelten Slash // am Zeilenanfang ermöglicht. In FFEIN7 werden Kommentare mit @ eingeleitet.

2.3 Variable

Symbole des weit verbreiteten Programmablaufplans (PAP) werden auch in dieser Arbeit wie nach [7] benutzt, „um den Ablauf eines Algorithmus und damit auch den eines Programms besser darstellen und dokumentieren zu können“. „Diese grafische Darstellung existiert losgelöst von der Programmiersprache und beschreibt Ablauf und Funktion eines Algorithmus universell.“ Mit einer solchen Darstellung lässt sich ein Algorithmus in einer beliebigen Programmiersprache verwirklichen.

2.3.1 Anweisungen

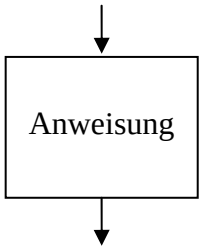


Abbildung 2-1: Anweisung

Anweisungen steuern die Ausführung eines Programms. Sie „sagen dem Rechner, was er tun soll“. Soll z. B. in der Sprache FFEIN der Variablen *a* der Wert *7* zugewie-

sen werden, ist nach dem optionalen Zuweisungsbeschleuniger *%z* wie folgt zu formulieren:

%z a=7;

Der Variablen der linken Seite der Anweisung wird nach dem Gleichheitszeichen (hier Zuweisungszeichen) der Wert der rechten Seite zugewiesen. Das Ende der Anweisung ist wegen der Formatfreiheit der Sprache durch ein Semikolon anzuzeigen. Die Zuweisungen können, um die Interpretation durch Vorgabe der Befehlsart zu beschleunigen, am Beispiel in FFEIN im Gegensatz zu Sprachen wie C++ oder Java mit *%z* eingeleitet werden. Wegen dieser Interpretationsbeschleunigung werden %f vor den Funktionen, %a vor den Anweisungen und %s vor den Standardanweisungen ähnlich angebracht. Der Programmablauf kann dadurch bei der Interpretation beschleunigt werden, siehe Quelltext am Beispiel in *Kapitel 3.6.4*.

Um den Speicherbedarf so gering wie möglich halten zu können, werden passende Datentypen z. B. Ganzzahlen als *long int*, *short int* in den Programmiersprachen angeboten.

Die Zahlen werden bei FFEIN nicht wie in der C++-Programmierung mit Float- oder Double-Befehlen unterschieden, sie sind stets doppelt genau.

Neben Zahlen müssen auch Texte verarbeitet werden können. Jeder char- Variable entspricht genau einem Zeichen. Der Typ String ist ein Array (Feld), das mehrere Zeichen speichern kann. String bildet eine Klasse und die string- Variablen entsprechend den Objekten dieser Klasse. (Über Objekte und Klassen siehe mehr in *Kapitel 2.5*).

Eine Textkonstante wird - wie auch die Zahlen - in FFEIN einfacher definiert als in C++. Er wird von zwei Hochkommata umschlossen, z. B. *txt='text'*.

2.3.2 Globale und lokale Variable

Vor allem bei größeren Entwicklungsprojekten, die von mehreren Personen bearbeitet werden, ist es empfehlenswert, alle Variablen innerhalb von Funktionen zu schützen. (Encapsulation) Die lokalen Variablen sind nur innerhalb der aktuellen Funktion gültig und sichtbar. Wenn die Funktion verlassen wird, werden die Speicherplätze der lokalen Variablen freigegeben. Die Vorteile der globalen und lokalen Eigenschaften gelten auch für die Vererbung der Klassen, siehe *Kapitel 2.9.2*.

Wenn in der Sprache FFEIN das Zeichen *&* den zweiten Buchstaben einer Variablen (heißt dann Commonvariable) bildet, dann überschreibt ein zugewiesener Wert auch die höher liegende Ebene. Parameter ohne dieses Zeichen können nur Werte importieren, sie dienen also zur Kapselung. Für Parameter mit Zeichen *&* ist der Export des Parameterwerts in die höhere, aufrufende Ebene gestattet.

2.4 Funktionen

Vor einer Funktion steht in FFEIN als Beschleuniger das Zeichen *%f*, dem eine Funktion folgt, wie beispielsweise *%fSIN(25)*.

Die Funktion *main()* ist das Kernstück jedes C++-Programms, mit der das Programm beginnt. Der Funktionsinhalt wird in den nachfolgenden geschweiften Klammern definiert.

Die vor der main()-Funktion definierten Funktionen können in der main()-Funktion aufgerufen werden.

Mit dem Schlüsselwort void wird ein Funktion eingeleitet.

```
void Funktion()
{
Befehlfolge;
}
```

C++ erlaubt Funktionspolymorphie oder Vielgestaltigkeit der Funktion, das heißt,

dass Funktionen, die sich in ihren Parametern unterscheiden, unter demselben Namen erzeugt und aufgerufen werden. Der Compiler entscheidet anhand der Art und Menge der Parameter, welche Funktion aufgerufen wird.

2.5 Klassen und ihre Objekte

Die Zusammenfassung von Variablen und Funktionen zu einer Einheit wird Klasse genannt. Die beschriebene Zusammenfassung ihrerseits heißt Kapselung.

Eine Klasse ist grundsätzlich eine Schablone, von der Abbilder erschaffen werden. Ein Objekt ist das Abbild einer Klasse. Beliebig viele Objekte können aus derselben Klasse gebildet werden.

Mit dem Punktoperator ´.´ kann auf die Methoden und Elemente eines Objektes zugegriffen werden, ähnlich wie in der bekannten Programmiersprache Visual Basic.

2.6 Operatoren

2.6.1 Mathematische Ausdrücke

Neben den wichtigsten Grundrechenarten (+,-,*,/) gehört auch die Modulo- Operation (%), die bei einer Ganzzahlen-Division den Rest zurückliefert, zu den mathematischen Operatoren.

In Programmen werden Variable oft in- und dekrementiert, deswegen die Anweisung *C=C+1* als *C*++ und die Anweisung *C=C-1* als *C*-- vereinfacht wurden.

2.6.2 Logische Ausdrücke

Ein Vergleichsausdruck mit Vergleichsoperatoren gibt den Wahrheitswert - wahr oder falsch - als Ergebnis an.

Sowohl das Negationszeichen als auch die Verknüpfungszeichen sind in logischen Ausdrücken verfügbar.

2.7 Felder

Ein Feld, das Variablen desselben Typs speichern kann, ist eine ein- oder mehrdimensionale Liste. Felder werden in der Fachliteratur häufig Array genannt.

Felder haben zwei Typen: die Werte- und die Textfelder. Bei Feldern hat der Variablenname einen Index für das entsprechende Feldelement.

In FFEIN enthält das Feldelement mit dem Index „0“ (Kopfwort) die Anzahl der Feldelemente insgesamt.

z. B.: a0=4; a1=7; a2=1; a3=-6; a4=12;

In C++ belegt ein Array mit *i* Elementen die Adressen von *0* bis *i-1*.

2.8 Kontrollanweisungen

2.8.1 Bedingungen

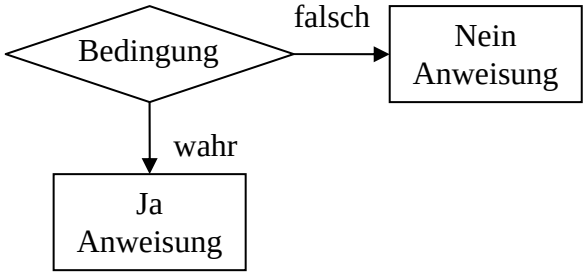


Abbildung 2-2: Bedingung

Der Rückgabewert der Bedingung ist entweder wahr oder falsch. Wenn eine Bedingung wahr ist, werden die nachfolgenden Anweisungen des Zweiges „wahr“ ausgeführt. Bedingungen können beliebig ineinander verschachtelt werden.

2.8.1.1 Auswahl

In C++:

- if (Bedingung) {Anweisungen;}
- if (Bedingung) {Anweisungen;} else{Anweisungen;}

In FFEIN:

- #WENN, Bedingung, Anweisung;
- #WENN, Bedingung; #DANN, Anweisungen; ##;
- #WENN, Bedingung; #DANN, Anweisungen; ##; #ODER, Anweisungen; ##;

2.8.1.2 Mehrfachauswahl

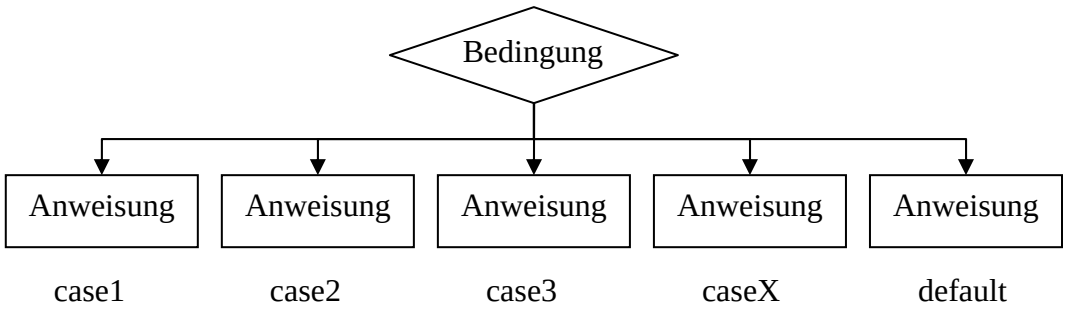


Abbildung 2-3: Mehrfachauswahl

Der Wert hinter *switch* (in FFEIN: *AUSWAHL*) wird mit den einzelnen Werten hinter *case (FALL)* verglichen. Wenn sie übereinstimmen, werden die Anweisungen hinter dem Doppelpunkt - andernfalls hinter dem *default:* - ausgeführt.

In C++:

```
switch (Wert)
{
case Wert1: Anweisungen;
case Wert2: Anweisungen;
case Wert3: Anweisungen;
case WertX: Anweisungen;
default: Anweisungen;
}
```

In FFEIN:

```
#AUSWAHL, Wert;
#FALL, Wert1; Anweisungen; ##;
#FALL, Wert2; Anweisungen; ##;
#FALL, Wert3; Anweisungen; ##;
#FALL, WertX; Anweisungen; ##;
#FALL, ; Anweisungen; ##;
##;
```

2.8.2 Wiederholungen (Schleifen)

Programmteile können mit einer Schleife wiederholt ausgeführt werden.

Solange die Bedingung in der Schleife wahr ist, wird der Schleifenkörper wiederholt. Ist die Bedingung falsch, verlässt das Programm den Schleifenkörper und springt zur nächsten Zeile hinter der Schleife.

Eine Schleife, deren Bedingung immer wahr ist, wird Endlosschleife genannt.

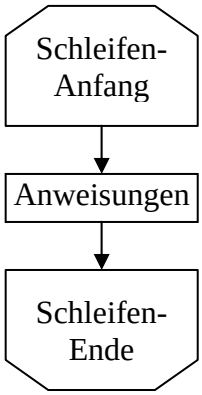


Abbildung 2-4: Wiederholung

2.8.2.1 Kopfgesteuerte Wiederholungen

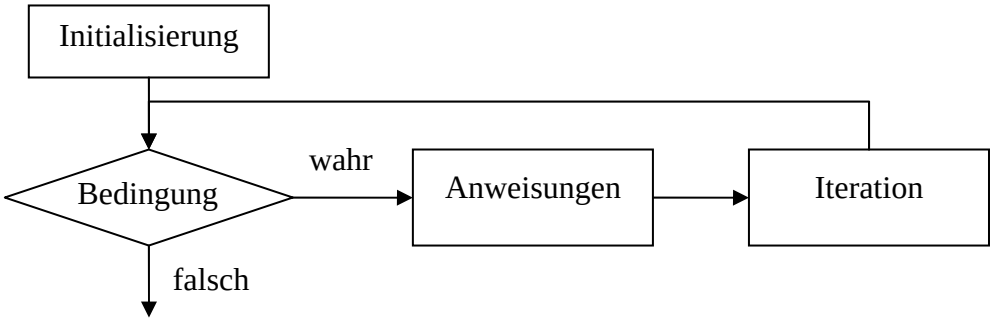


Abbildung 2-5: Kopfgesteuerte Wiederholung

Als Steuerungselement einer Schleife wird die Zählvariable benutzt, die zuerst initialisiert wird. Die nächsten Schritte - die wiederholte Überprüfung, das Ausführung der Anweisungen und die ständige Inkrementierung bzw. Dekrementierung der Variable – werden nach der Auswertung der Bedingung wiederholt oder abgebrochen.

In C++:

```
for (Initialisierung; Bedingung; Iteration)
{
Anweisungen;
}
```

In FFEIN:

```
Initialisierung;
#SCHLEIFE, Bedingung, Iteration;
Anweisungen;
##;
```

2.8.2.2 Fußgesteuerte Wiederholungen

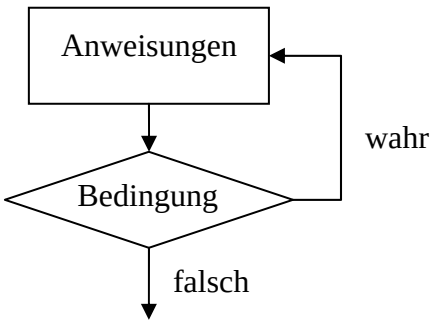


Abbildung 2-6: Fußgesteuerte Wiederholung

Die Schleife wird durch die Bedingung erst am Ende getestet, wenn das Programmziel erfordert, dass der Schleifenkörper mindestens einmal ausgeführt wird.

In C++:

```
Initialisierung;
do
{
Anweisungen;
Iteration;
}
while (Bedingung);
```

In FFEIN:

```
Initialisierung;
#WIEDERHOLE, Bedingung;
Anweisungen;
Iteration;
##;
```

2.9 Programmaufbau

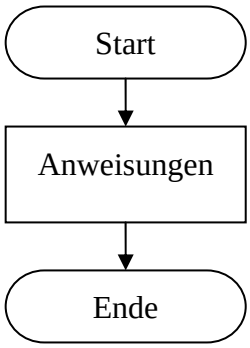


Abbildung 2-7: Start und Ende der Methode

2.9.1 Methodenverschachtelung

Die Methoden eines Programms stehen nicht als Kettenglieder hintereinander. Sie bilden Verschachtelungsstufen, d.h. Methodenkeller. Eine Methode *(name.met)* ruft eine andere auf zweierlei Weise auf.

2.9.1.1 Gleiche Ebene

Die erste Möglichkeit ist, dass die Methode ihre Daten durch einen Puffer eingefügt bekommt, siehe *Abbildung 2-8*. (Quelltext: #PUFFER, Methodenname;) Die erste Zeile eines Puffers ist eine Standardanweisung: %sCONTINUE;. Alle Puffer haben diesen gleichen „formalen Kopf“.

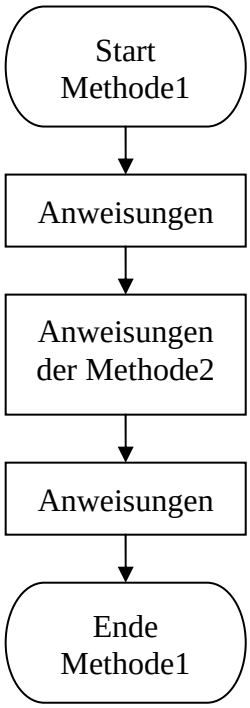


Abbildung 2-8: Puffer in der Methode

2.9.1.2 Unterprogrammaufruf

Die zweite Möglichkeit ist der klassische Unterprogrammaufruf gängiger Programmiersprachen, siehe *Abbildung 2-9.*

Listenelemente einer Parameterliste von Unterprogramm bestehen aus einem Parameternamen, dem nach dem Zuweisungszeichen „=“ ein Voreinstellungswert zugewiesen wird. Bei Einhaltung der Reihenfolge können die Parameternamen weggelassen werden.

z. B.: XYZ(l=1,k=’JA’,m=0) oder XYZ(1,’JA’,0)

Der Präprozessor in C++ sucht zuerst den gesamten Source Code nach Anweisungen, die mit einem Doppelkreuz eingeleitet werden, durch. Die von dem Hashzeichen # beginnende Anweisung *include* bewirkt, dass der Quellcode der angegebenen Datei eben hier in dem entsprechenden Programm eingefügt wird.

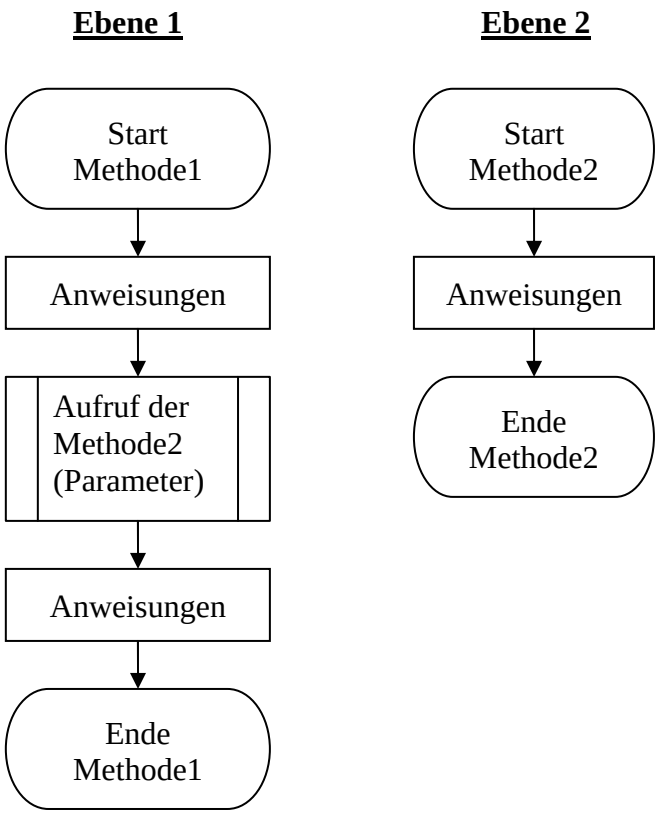


Abbildung 2-9: Prozeduraufruf

2.9.2 Vererbung und Ableitung in C++

Vererbung innerhalb einer Programmiersprache bedeutet, dass Klassen ihre Eigenschaften, also ihre Methoden und Datenelemente, auf andere Klassen übertragen, d.h. vererben. Die Vererbung aus der Sicht der neuen Klasse wird Ableitung genannt. Die neue Klasse wird von einer bereits bestehenden Klasse (Basisklasse) abgeleitet. Durch Veränderung einzelner Eigenschaften wird diese speziellen Bedürfnissen angepasst.

C++ hat drei verschiedene Schlüsselworte, um die Rechte von Datenelementen und Methoden zu definieren:

- public: Der Klassenbestandteil ist komplett sichtbar und kann von abgeleiteten Klassen gelesen werden.
- protected: Der Klassenbestandteil ist für abgeleitete Klassen komplett sichtbar, ansonsten aber privat.
- private: Der Klassenbestandteil kann nur von den Methoden der eigenen Klasse genutzt werden.

2.10 Blockstrukturen

Befehle können in FFEIN in Blockstrukturen geklammert werden.

Beispiele für verschiedene Blockanweisungen:

- Punktanweisungen:

```
    %aPUNKTE:
    %aMITZ;
        Befehlfolge;
%aENDE:PUNKTE;
```

- Schweißverbindungen:

```
    %aNAEHTE:
        Befehlfolge;
    %sENDE;
```

- Schraubenbindungen:

```
    %aSCHRUBEN:
        Befehlfolge;
    %sENDE;
```

- Strukturelle Verbindungen:

```
    %aVERBINDUNG:
    WERKSTATT;
    Teileliste;
    %sENDE;
```

Ein Block dient auch zum Datenaustausch zwischen den verschiedenen Ebenen.

z. B. :

```
ZIELBLOCK:BLOCK(0)-1;
KOPIERE: 'feld'<>, '\angabename'<>;
ZIELBLOCK;
```

Die Funktion der Blockstrukturen wird in C++ mit den vorher bereits vorgestellten Klassen gelöst.

2.11 Generierung stabförmiger Bauelemente

Das *$* Zeichen leitet in FFEIN als Schlüsselzeichen das Generierungsstatement ein. Die Glieder der Spezifikationsliste des Statements bestehen aus der Startpositionsnummer (POS=1), der Benennung (BEN='Stütze'), dem Profilnamen (PR='IPE360'), dem Material (QPL= S235JRG2), den Punkten (P1=1, P2=2), denen das Stabelement zugeordnet wird, und den Lageparametern (QV=RECHTS, PA=VORDER, TV=HINTEN).

z. B. : $POS,BEN,PR,QPL,P1-P2,QV,PA,TV;

An der Nutzoberfläche werden die Teile wie in *Abbildung 2-10* dargestellt. Für die Startpositionsnummer gilt dabei nach [8]: „die Startpositionsnummer ist die niedrigste Positionsnummer, die ein Teil nach dem Detaillieren bei der automatischen Gleichteileerkennung erhalten kann. Ist diese Nummer durch ein anders aussehendes Teil besetzt, erhält es automatisch die nächsthöhere, freie Nummer. Über die Startpositionsnummer können also Teile zweckmäßig organisiert werden, indem man systematisch z. B. Stützen und Pfosten ab 1, Riegel ab 100, Wandriegel ab 200, Verbände ab 300 etc. nummerieren lässt, was der firmenspezifischen Ablauforganisation hilft.

Punkt 1 in der Vorschau *(Abbildung 2-10)* ist der Startpunkt des Profilstabes, der beim Verlegen zuerst gewählt wird, Punkt 2 ist der Zielpunkt. Punkt 1 und die beiden kräftigen Pfeile in *Abbildung 2-10* zeigen die Arbeitsebene an. Senkecht darauf wird mit einem Doppelpfeil die Blickrichtung angedeutet."

Das in *Abbildung 2-10* dargestellte Muster enthält die folgenden Eigenschaften:

- Der Optionsknopf der Querlage QV ist auf RECHTS eingestellt.
- In der mittleren Spalte (Profilansicht PA) ist VORDERansicht - Ansicht auf die hohe Seite (Stegseite) eines Profils - ausgewählt,
- und die Tiefenlage TV ist auf HINTEN eingestellt.

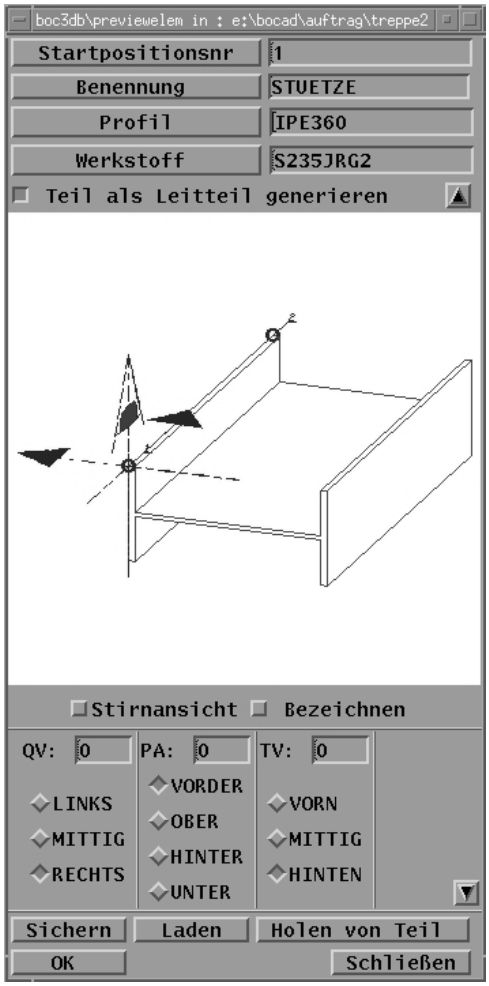


Abbildung 2-10: Verlegen eines stabförmigen Bauelements

2.12 Basis-Methoden aus systemeigenen Methodenbibliotheken für die Entwicklung von Makros

Um Lösungsschritte für Konstruktionsaufgaben formulieren zu können, sind anwendungsspezifische Basisbefehle hilfreich. Die folgende Aufstellung gibt einen Eindruck, welche Basisbefehle typisch sind:

- PUTPKTCST: erzeugt Punkte an der Nutzoberfläche.
- DET_SCHN: schneidet ein Teil entlang einer Ebene ab.
- ANPT_TEIL: passt ein Teil einem anderen Teil an.

- DRAULANG: wird bei Schraubenverbindungen angewendet. Sie erzeugt Langlöcher.
- DRAUSCR: erzeugt Schraubenverbindungen.
- SCHWANA: erzeugt Schweißnahtverbindungen.
- GETELM: liest Daten aus standardisierten Speichern aus.
- SETELM: schreibt Daten in standardisierte Speicher.
- AINR: wandelt einen ASCII-Text in einem Zahlenwert.
- RINA: macht aus den entsprechenden Textwert einen entsprechenden Real-Wert, sofern möglich.
- ELEMELI: löscht überflüssige Hilfsteile.

2.13 Lesen und Speichern von Daten

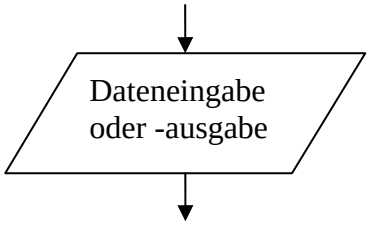


Abbildung 2-11: Dateneingabe und -ausgabe

In der Interpretersprache FFEIN wird das Lesen und Speichern von Daten mit einem Methodenaufruf gelöst. Die entsprechende Methode zum Lesen lautet: GETELM() (aus Eng.: get element), und zum Schreiben lautet: SETELM() (aus Eng.: set element).

Daten, z. B. die Eigenschaftsnamen und Eigenschaftswerte eines Objekts werden mit SetElm in Dateien *name.rsc* als Text im ASCII-Format geschrieben. Die einzelnen Zeilen dieser Dateien enthalten je eine Datenmenge von maximal 96 Bytes, wovon jeweils die ersten 16 Bytes als Kopfwort der Identifizierungsschlüssel: key1 (=4 Bytes) + key2 (=4 Bytes) + bkey (=8 Bytes) ist. Auf derartige Kombinationen von Ob-

jekteigenschaften wird also über einen Schlüssel lesend und schreibend zugegriffen.

Solche *name.rsc* Dateien dienen zum Speichern für Nutzoberflächedaten. Die Eingabefelder z. B. importieren Daten von diesen Speicherdateien und exportieren Daten in die RSC-Dateien, siehe *Kapitel 3.2.4*. So sind auch die Voreinstellungen der Eingabefeldervariablen, die auch in der gewünschten Landessprache an der Nutzoberfläche erscheinen sollen, hier gespeichert. Aus den oben genannten Gründen werden die Dateien mit der Erweiterung *.rsc* bei der Übersetzung beachtet.

Die Methode get() ist in C++ die einfachste Art, etwas aus einer Datei zu lesen, und put() ist beim Schreiben das Äquivalent zur get()-Methode.
C++ bietet eine Art Lesezeichen, das die aktuelle Stelle in der Datei anzeigt. Das Lesezeichen steht an der Stelle des letzten Zugriffs. Mit der Methode seekg() ist ein Lesezeichen zu suchen, das ein Kriterium erfüllt. Die Methode tellg() gibt die aktuelle Position des Lesezeichens wieder.

2.14 Fehlerbehandlung

Wenn der verwendete Interpreter oder eine CAD-Methode einen Fehler oder eine unlösbare Situation im Programmablauf erkennt, muss dem Anwender eine Meldung zur Information und Begründung im Ablaufprotokoll-Fenster ausgegeben werden, siehe *Kapitel 3.3.* Diese Meldung wird mit Hilfe des Ausgabebefehls *DRUCKE* erzeugt. Die Meldungstexte inklusive der Werte von zugehörigen Variablen werden im Fenster sichtbar.

Zur Globalisierung des CAD-Systems mit automatischer Ausgabe von Fehlermeldungen in den verschiedenen Landessprachen und Schriftzeichen wird hier ein neuer Ansatz vorgeschlagen: Meldungen sollten in einem *.inp* File gepuffert und dann von diesem File automatisch übersetzt in das Ablaufprotokoll ausgegeben werden. So erscheinen auf der Nutzoberfläche stets übersetzte Textobjekte. Nur eine einzige, aktuelle Sprachdatei ist dann beim Sprachenwechsel zu beachten. Einige Megabytes an Programmgröße werden so gespart.

Ein praktisches Beispiel mit der Meldung „Plattenhöhe 250 ist kleiner als die zul. Plattenhöhe“ ist in der Interpretersprache wie folgt zu formulieren:

```
DRUCKE: %g1, 571, TEXT(HPL);
```

wobei:

- %g1 die Fremdsprachendatei *name.inp* aufruft,
- 571 sich auf die Nummer der Fehlermeldung in der oben genannten Datei bezieht,
- TEXT(HPL) den Wert der Variable *HPL* als Text aufruft.

In der aufgerufenen Fremdsprachendatei (hier deutsche Sprache) ist als Satz 571 folgender Inhalt gespeichert:

```
/* 571 fusz31
010000 000000 000000 000000 000000 000000 000000 000000 000000 000000
010000 000000 000000 000000 000000 000000 000000 000000 000000 000000
(' Plattenhöhe ',HPL,' ist kleiner als die zul. Plattenhöhe ')
```

Die erste Zeile ist eine Kommentarzeile, eingeleitet mit der Zeichenfolge /*. Die zweite und dritte Zeile geben verschlüsselt die Art der im Text genutzten Variablen (z. B.: Zahl, Text) an, hier also der Variablen *HPL*, die den Wert 250 habe.

2.15 Informations- und Suchbefehle

FFEIN bietet die folgenden Befehle, um den Programmierer und den CAD-Konstrukteur über ihre Arbeit in dem Protokollfenster zu informieren.

- DRUCKE:a; druckt den Variablenwert 'a' im Ablaufsprotokoll-Fenster aus.
- QUITTUNG: listet alle Variablen der verschiedenen Ebenen auf.
- PUNKTE:
  INFORMIERE: ALLE(Punktnummerliste);
  MELDE: LISTEX, 'xp';

LISTEY, 'yp';
LISTEZ, 'zp';
ENDE: PUNKTE;
Die Koordinaten der gewünschten Punkte werden mit MELDE zu Kontrollzwecken gemeldet

- MELDE: EXTREMA, 'ex'; Dieser Befehl gibt die Maximal- und Minimal- Koordinaten der Teilepunkte an.
- TRACE; NOTRACE; TRACE schreibt die Ergebnisse der ausgeführten Operationen mitlaufend ins Ablaufprotokoll. NOTRACE schaltet diese Leistung wieder ab.
- LISTNUM; NOLIST; listet den Quelltext, der zwischen diesen Befehlen liegt, einschließlich Zeilennummer des Statements auf.
- INFO3D:
  LAGE: REFERENZ;
  PUNKTLAGE: TRANSFORMIERT;
  PUNKTNUMMERN: 999;
  TEIL: idnr;
  SCHNITTFLÄCHE: Eb1-Eb2-Eb3;
  ENDE;
  Das Teil wird mit einer virtuellen Ebene abgeschnitten und die Daten der Schnittpunkte der Teil-Ebene werden auswertbar. Die Punktnummern werden ab 1000 gemeldet.
- Das folgende Beispiel zeigt, wie konstruktionstypische Aufgaben kompakt in FFEIN formuliert werden: Es wird geprüft, welche Lage Punkt 17 zum geschlossenen Polygon der Punkte 1 bis 8 hat. Ergebniswerte können sein: 0- außerhalb, 1-innerhalb, 2-auf, 3-auf der Ecke einer Linie.

  INFORMIERE: (<1,8>);
  DRUCKE: INNPKT(17);

3 Werkzeuge für Nutzoberflächen

Die Gestaltung der Nutzoberfläche darf bei der Programmierung nicht in den Hintergrund gedrängt werden, siehe auch nach [28]:

„Die Benutzerschnittstelle von Programmen stellt einen der wichtigsten Programmteile dar. Schlechte Benutzerschnittstellen sind zumeist für die mangelnde Akzeptanz an sich recht guter, effektiver Programme verantwortlich. Anderseits ist dies leider nur allzu oft ein stark vernachlässigter Bereich gerade auch der professionellen Programmierung."

Der Programmanwender kommuniziert mit dem angewendeten Programm über die Nutzoberfläche, die dazu folgende Elemente bietet:

- Mausaktivitäten: Wählen, Ziehen, Zoom, Pan
- Tastenkombinationen (Shortcuts)
- Dialogfelder: mit Schaltflächen, Symbolen usw.
- Protokollfenster
- Klickmöglichkeit an der Nutzoberfläche (Pick Requester)
- Leitbilder
- Aktive Oberfläche mit grafischen 2D-Methoden

Diese Elemente werden nachfolgend diskutiert.

3.1 Mausaktivitäten und Shortcuts

Die Arbeitsschritte Wählen und Ziehen werden in CAD-Programmen so verwendet, wie sie in Windows bereits bekannt sind. Diese Strategie, allgemein Bekanntes immer wieder zu verwenden und nicht davon abzuweichen, ist für die Akzeptanz von

CAD-Systemen entscheidend.

Zoom bedeutet die Vergrößerung eines Bildausschnittes auf dem Bildschirm. Dabei wird im verwendeten CAD-System das Zooming zu einer einzigen Handbewegung vereinfacht. Mit der mittleren Maustaste wird eine Diagonale gezogen, z. B. von der linken unteren Ecke zur rechten oberen Ecke, um den gewünschten Ausschnitt zu definieren. Durch einen Klick der mittleren Maustaste wird das Zoomen rückgängig gemacht. Sowohl das Zoom im Zoom als auch mehrfache Mittelklicke sind rekursiv möglich.

Die Bildvergrößerung und -verkleinerung werden in anderen Programmen auch durch Scrollen gelöst.

Ein weiteres wichtiges Arbeitsmittel heißt Pan. Pan wird für die waagerechte oder senkrechte Verschiebung in einem Bild eingesetzt. Pan arbeitet im verwendeten CAD-System wie Zoom, wobei die Ausschnittdiagonale linienartig schmal bleiben muss, das heißt, dass eine waagerechte oder senkrechte Linie gebildet wird.

3.2 Dialogfelder

Die Dialogfelder sind Fenster mit verschiedenen Bausteinen für Aktionen des Anwenders, siehe *Abbildung 3-1*. Davon seien die wichtigeren genannt:

- Schaltflächen
- Symbole
- Label
- Eingabefelder
- Optionsfelder
- Darstellungsbilder

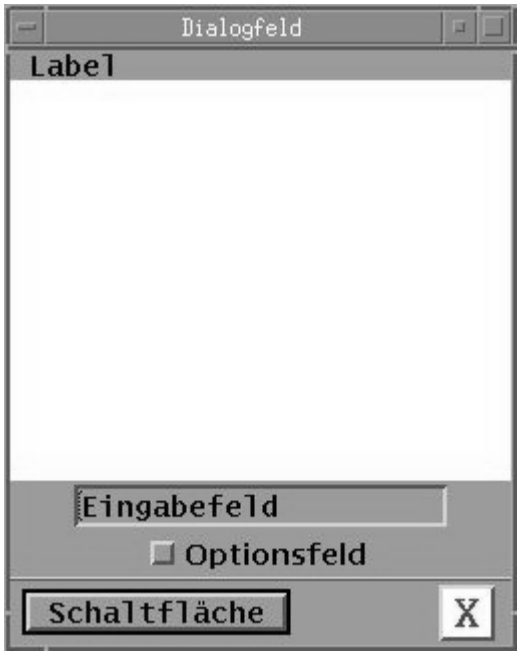


Abbildung 3-1: Dialogfeldbausteine

Der Quelltext des in der *Abbildung 3*-1 stehenden Dialogfeldes:

```
/* Dialogfeld
------5500
IBO (1)
BOX (1000,0), (3,Dialogfeld,-), (0,0), (+700+500), (1,BoComboClbCanc)
TBL (1001,1000), (1,options,-), (1,0,0), (6)
TBL (2000,1001,1,1,8), (1,block1,-), (1,0,0), (6)
LAB (2020,2000,1,1,8), (1, Label ,-), (2,a,-)
DRA (2021,2000,2,1,8), (5,Area1,-), (250,200)
TBL (1010,1001,2,1,1), (1,block2,-), (1,0,0), (6)
EDI (1012,1010,1,1), (1, ,grf_text_s), (1,12351,text,2edi,default ,16,80),
    (1,12351,text,2edi,default ,16,80), (0,20,20)
TOG (1014,1010,2,1), (2, Optionsfeld ,-), (18,21162,pbBL,BLfa,default ,16, 4),
    (18,21162,pbBL,BLfa,default ,16, 4)
SEP (1102,1001,5,1,20), (1,sep1,-), (0,5)
TBL (6600,1001,6,1,1), (1,block3,-), (1,0,0), (6)
PUB (6601,6600), (1, Schaltfläche ,-), (2,a,-), (Aktion starten / Box verschwindet), (2,0,0),
    (1,BoComboClbOK)
LAB (6604,6600), (1,      ,-)
PUB (6605,6600), (1,a_close24.xpm,-), (2,a,-), (Box verschwindet ohne Änderung),
    (0,1,0), (1,BoComboClbCanc)
------999999
/*
```

Die im Quelltext auftretenden Textobjekte können als Variablen definiert werden. So werden die CAD-Programme für die Fremdsprachlichkeit und den internationalen Einsatz bereitgestellt. Die zu definierenden Textelemente sind die Aufschriften im Dialogfeld und die Quickinfos. Die letztgenannten Textobjekte geben kurze Informationen (Messages) über das entsprechende Dialogfeldelement, über welchem der Cursor liegt.

Definierung der Textvariablen an einem Beispiel:
#def Aufschrift Schaltfläche **#def** Quickinfo Aktion starten / Box verschwindet PUB (6601,6600), (1, **#Aufschrift**,-), (2,a,-), **(#Quickinfo**), (2,0,0), (1,BoComboClbOK)

Dialogfelder werden in Konfigurationsdateien (mit der Erweiterung *.cfg*) programmiert. Im verwendeten System wird dazu jede Dialogbox durch 6 Bindestriche und durch die Boxnummer (logische Nummer) eingeleitet, z. B. ------*5500*. In der nächsten Zeile wird der Typ des Dialogfensters durch den Befehl *IBO* und die Nummer des entsprechenden Dialogboxentyps definiert. Danach folgt die Fensterdefinition, welche mit dem Kommando *BOX* beginnt und die eigene Nummer, den eigenen Namen des Fensters und die Koordinaten der Fensterplatzierung am Bildschirm enthält. Die Definitionen der wichtigeren Bauelemente des Dialogfeldes, wie die Schaltflächen, die Symbole etc. folgen. Der Block ist durch die nächste Box oder durch die Bindestriche mit einer beliebigen logischen Nummer als letzte Box in der Datei abgeschlossen.

Die wichtigeren Bauelemente des Dialogfeldes, die alle eine eigene Nummer und einen eigenen Namen besitzen, werden in den folgenden Kapiteln vorgestellt.

3.2.1 Schaltflächen

Durch Schaltflächen, siehe *Abbildung 3-1*, kann der Anwender z. B. vom aktuellen Dialogfeld auf ein anderes weiternavigieren, Dialogfelder schließen und/oder Methoden ausführen.

Die üblichen Schaltflächen führen - nach [8] - die folgenden Aktionen aus:

- *OK*: Das Dialogfeld wird geschlossen und die mit dem Dialogfeld spezifizierte Aktion wird ausgeführt.
- *Schließen*: wirkt wie *OK*, jedoch ohne die Dialogfeld-Aktion auszuführen.
- *Ausführen*: wirkt wie *OK*, das Dialogfeld bleibt jedoch für Folgeaktionen offen.
- *Sichern*: sichert den kompletten Inhalt eines Dialogfelds unter einem Namen.
- *Laden*: lädt den unter einem Namen gespeicherten Inhalt eines Dialogfelds.
- *Weiter*: Die Einstellungen des Dialogfeldes werden gespeichert, das aktuelle Dialogfeld wird geschlossen, und das Dialogfeld des nachfolgenden Arbeitsschritts wird aufgerufen.
- *Zurück*: Die Einstellungen des Dialogfeldes werden gespeichert, das aktuelle Dialogfeld wird geschlossen, und das Dialogfeld des vorliegenden Arbeitsschritts wird aufgerufen.

Die Definition einer Schaltfläche wird im verwendeten System mit dem Befehl *PUB* (pushbutton) eingeleitet, wonach die Platzierung im Dialogfenster, die Aufschrift und die Schriftart bestimmt werden. Die Quickinfos und Aktionen werden letztendlich im laufenden Programm mitgeteilt.

```
PUB (6601,6600), (1, Schaltfläche ,-), (2,a,-), (Aktion starten / Box verschwindet), (2,0,0),
    (1,BoComboClbOK)
```

Einige formale Hürden der Programmierung sind auffällig: Wenn die Aktion (Callback) eine Sequenz von mehreren Befehlen aufruft, die die maximale Zeilenlänge des Interpreter-Puffers (96 Charakter) überschreiten, wird die Aktion beispielsweise *(1, Bo Ho Clbp Do Clb Lis <6000: 12351: 123>*) abgekürzt, wobei *6000:12351* der Hinweis auf die Config-Datei und *123* der Hinweis auf die Boxnummer ist. Die Callbacks werden nach der Zeile ------*123* aufgelistet. Ein einzelner Befehl (hier Aktion) darf höchstens aus 256 Zeichen bestehen.

3.2.2 Symbole

Die Symbole, z. B. für „Dialogfenster schließen“  in der *Abbildung 3-1*, sind spezielle Schaltflächen, die nicht mit einer textförmigen Aufschrift, sondern mit einem erklärenden Bild ihre Funktion verdeutlichen. Sie werden genauso wie die Schaltflächen gebildet, nur statt der Aufschrift wird die Ikondatei *name.xpm* definiert. Die Symbole werden heutzutage immer öfter verwendet, weil sie die Frage der Mehrsprachigkeit mit ihren eindeutigen, selbsterklärenden Symbolbildern auf einfachstem Wege lösen können. Sie können mit einer Auflösung von 24*24 Bildpunkten (Pixels) bereits recht gut erklärend gestaltet werden.

PUB (6605,6600), (1,a_close24**.xpm**,-), (2,a,-), (Box verschwindet ohne Änderung), (0,1,0), (1,BoComboClbCanc)

3.2.3 Label

Labels, siehe *Abbildung 3-1*, sind textförmige Hinweise im Dialogfeld. Sie dienen meistens als Erläuterungen für die Eingabefelder. Ein Label hat demnach die Fremdsprachlichkeit zu berücksichtigen.

Beim Programmieren eines Labels wird die Zeile mit der Zeichenfolge *LAB* eingeleitet, dann werden die Platzierung im Dialogfenster, die Aufschrift und die Schriftart festgelegt.

LAB (2020,2000,1,1,8), (1, Label ,-), (2,a,-)

3.2.4 Eingabefelder

Eingabefelder, siehe *Abbildung 3-1*, dienen zum Eingeben von Texten oder Werten.

Zur Definition eines Eingabefeldes werden das Schlüsselwort *EDI* und dann wieder

die Koordinaten der Platzierung im Dialogfeld und die Schriftart angegeben. Es folgen zwei Speicher, der Datentyp des Eingabefeldes (z. B. text, integer, double), die Länge des Eingabefeldes, die Maximallänge des einzugebenden Textes. Der erste Speicher enthält die Voreinstellung, die beim Öffnen des Dialogfeldes eingelesen wird, und der zweite speichert die Eingabe des Anwenders. (Über den Speicher und über die Mehrsprachigkeit siehe auch *Kapitel 2.13*.)

EDI (1012,1010,1,1), (1, ,grf_text_s), (1,12351,text,2edi,default ,16,80), (1,12351,text,2edi,default ,16,80), (0,20,20)

3.2.5 Optionsfelder (Schalter)

Ein Optionsfeld (Toggle), siehe *Abbildung 3-1*, sorgt dafür, dass die zugehörige Funktion ein- oder ausgeschaltet wird. Er gibt 0 (falsch) oder 1 (wahr) als Ergebnis wieder.

TOG ist das Schlüsselwort in diesem Fall, dann folgen wieder die o. g. Eigenschaften wie die Platzierung im Dialogfenster, die Aufschrift, die Schriftart und die zwei Speicher.

TOG (1014,1010,2,1), (2, Optionsfeld ,-), (18,21162,pbBL,BLfa,default ,16, 4), (18,21162,pbBL,BLfa,default ,16, 4)

3.2.6 Darstellungsbilder

Die Darstellungsbild-Dateien, die mit dem eben angewendeten Programm angefertigt werden, und die in dieser Dissertation in den Beispielen die Erweiterung *.bmf_* (**b**o-cad **m**eta**f**ile) erhalten, können in das Dialogfeld eingelesen werden. Für solche Grafikdateien wurde ein Zeichnungsbereich in der *Abbildung 3-1* vorbereitet.

Im Quelltext wird der Bilddarstellungsbereich am Zeilenanfang durch den Befehl *DRA*, gefolgt von der Platzierung im Dialogfeld, der Bereichshöhe und der Bereichsbreite definiert.

DRA (2021,2000,2,1,8), (5,Area1,-), (250,200)

3.3 Protokollfenster

Das Protokollfenster ist ein Fenster oder ein Fensterteil, in dem die aktuellen Konstruktionsschritte festgehalten werden, ebenso die entsprechenden Meldungen, Warnungen und Fehler.

Abbildung 3-2: Protokollfenster

Im in *Abbildung 3-2* dargestellten Beispiel wurde ein Auftrag geöffnet, und zwischen zwei Trägern wurde ein Anschluss durch eine systemspezifische Konstruktionsmethode ausgeführt. Die fehlerhaften Einstellungen der Konstruktionsmethode haben die Kollision verursacht, worüber der Anwender im Protokollfenster in der gewählten Sprache gewarnt wurde. Der programmiertechnische Hintergrund der Mehrsprachigkeit wurde in *Kapitel 2.14* erklärt.

3.4 Wahlmöglichkeiten an der Nutzoberfläche

Die Eigenschaften und Beschränkungen eines Klicks sind in den Dateien *name.rqi*

definiert. Die Klickmöglichkeiten (Pick Requester) werden in *Kapitel 3.6.3* durch ein Beispiel detailliert erklärt.

3.5 Leitbilder

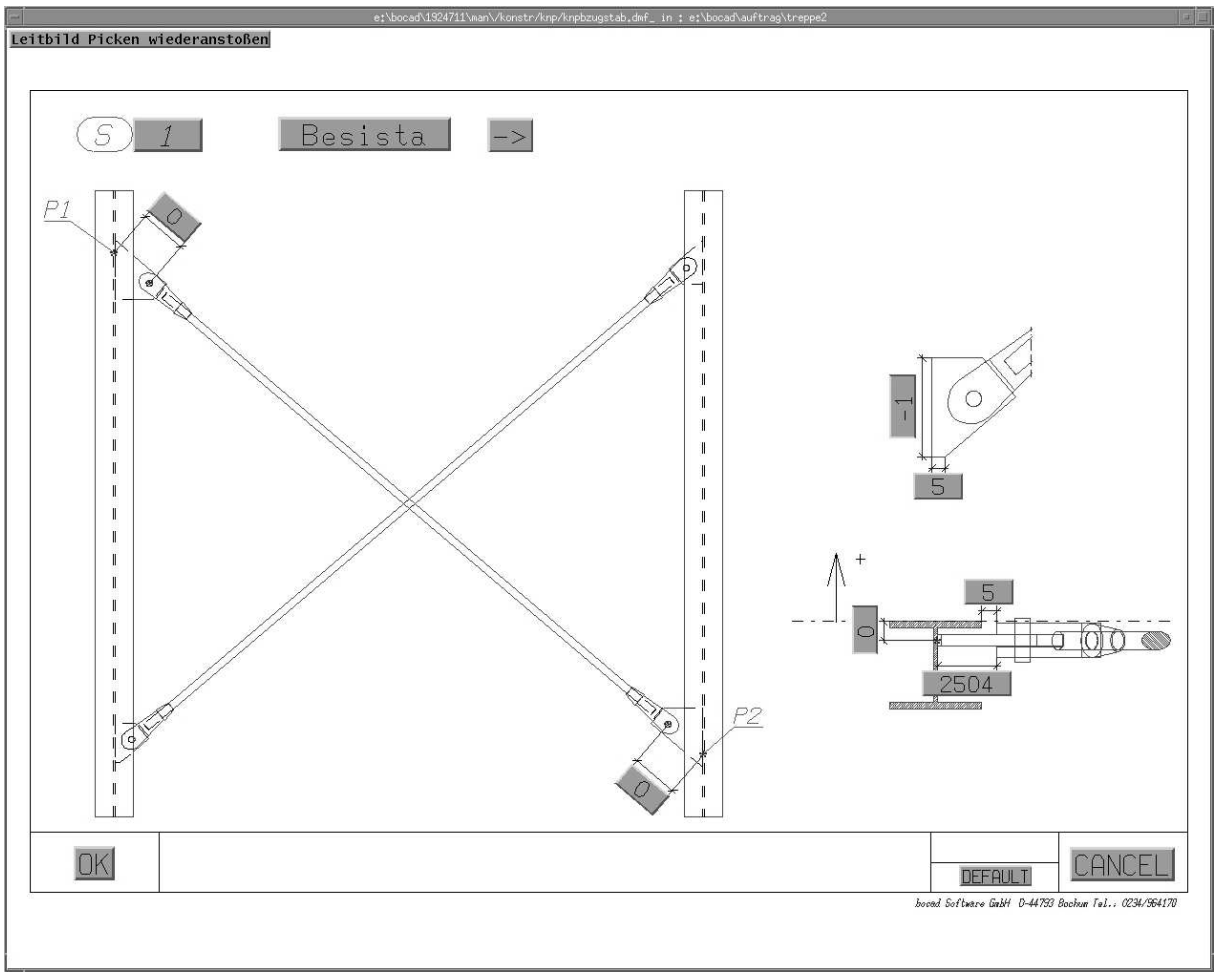


Abbildung 3-3: Leitbild des Zugstabs

3.5.1 Ziele von Leitbildern

Leitbilder dienen zur Eingabe und Online-Dokumentation von Methodenspezifikationen. Mit Hilfe der Leitbilder kann der Benutzer die Variablen der Konstruktion eindeutig und anschaulich bestimmen.

In [8] wird der Zweck von Leitbildern vorgestellt:

„Leitbilder dokumentieren bildhaft und selbsterklärend die Leistungen einer Konstruktionsmethode. Gleichzeitig gestatten sie das Auswählen von Lösungen und das Eingeben von Werten in Eingabefelder innerhalb von Maßketten. Leitbilder sind also gleichzeitig Dokumentation und anschauliches Eingabeformular. Umständliches und zeitraubendes Nachschlagen in Handbüchern sowie umfangreichere textförmige Erläuterungen entfallen."

Ein wichtiger, wesentlicher Vorteil von Leitbildern anstelle der Direkteingabe von Werten in der maßstäblichen Realkonstruktion ist, dass die Konstruktionselemente unmaßstäblich vergrößert und somit gut erkennbar dargestellt werden können, siehe *Abbildung 4-4.*

3.5.2 Konzeption und Entwicklung eines konkreten Beispiels „Zugstabkreuz"

Die CAD-Methode Zugstabkreuz stellt eine Verbindung aus 4 Stäben und 1 Kreisscheibe im Kreuzungspunkt her. Diese Bauelemente sind von den drei Firmen, *Besista*, *Detan* und *Rodan,* lieferbar. Hierzu wird als Anwendungsbeispiel ein Leitbild entwickelt, das alle in *Kapitel 3.5.1* aufgeführten wesentlichen Ziele erfüllt. Der Entwicklung des Leitbilds werden die Konstruktionsmethoden für anspruchsvoll gestaltete Verbände der o. g. Firmen zugrunde gelegt.

Der erste Entwicklungsschritt für ein neues Leitbild ist die Überlegung, welche Ansichten und Schnitte des Leitbild-Gegenstands erforderlich sind, um das Objekt und seine Spezifikationen textfrei und selbsterklärend darzustellen.

Das Erscheinungsbild sollte bezogen auf die Einrichtung und den Aufbau möglichst ähnlich zum bestehenden Leitbild ZUGSTAB *(*siehe *Abbildung 3-3)* gestaltet werden.

Hier sind als Übersicht die Träger mit dem Zugstabkreuz und als Detail auch die Schnitte der Konstruktion erforderlich. Sie werden jeweils als *name.bmf_* Datei in das Leitbild-Formular eingefügt, siehe *Abbildung 3-4.*

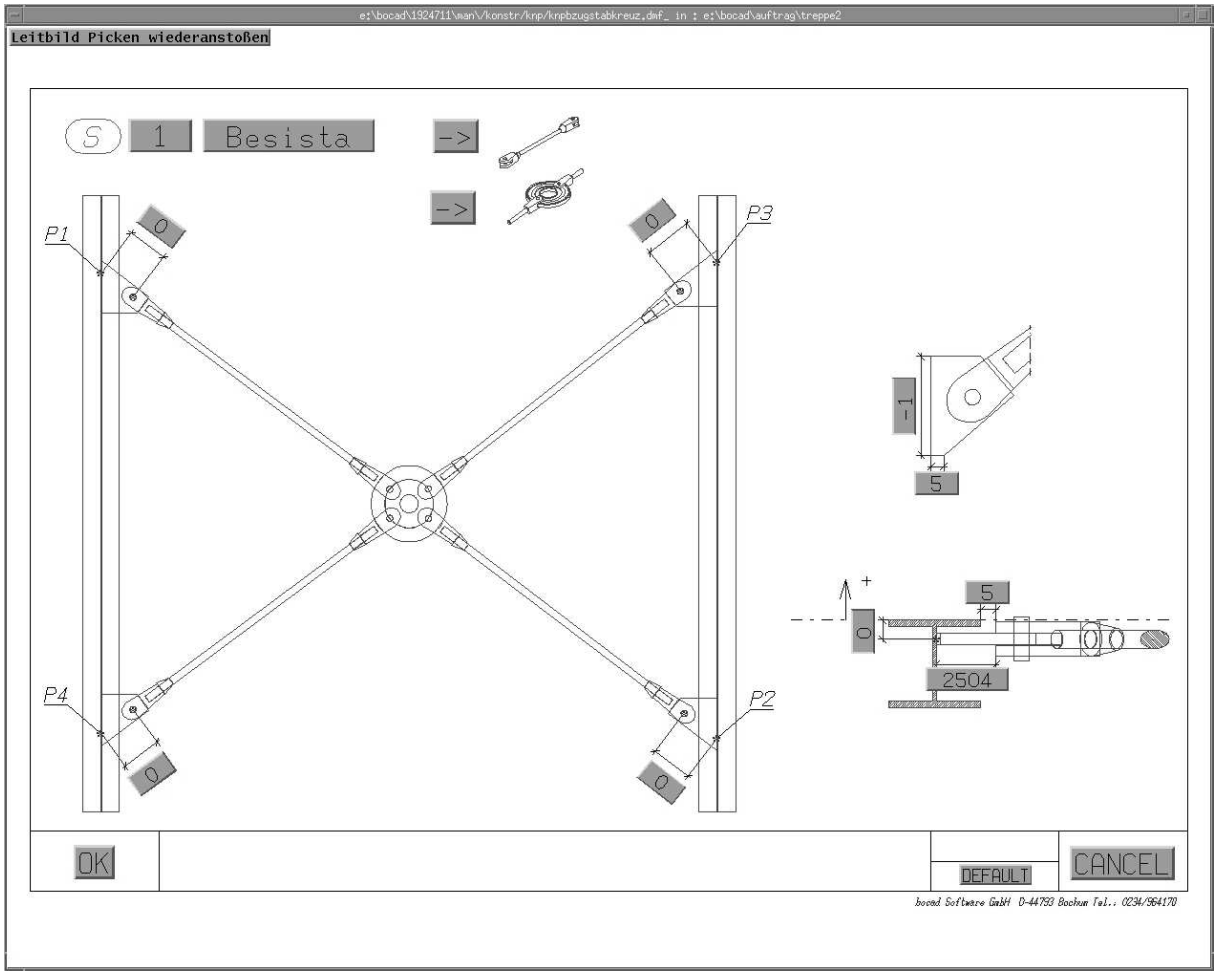


Abbildung 3-4: Konzeption für das Leitbild eines Zugstabkreuzes

Die Schaltfläche [Besista] in *Abbildung 3-4* als Voreinstellung zeigt das Hersteller-System, das beim Konstruieren verwendet wird. Die weiteren zwei Schaltflächen [→] lassen die Möglichkeit zu, die Eigenschaften des Zugstabs und der Kreisscheibe nach der Systemauswahl bei Bedarf stärker detailliert anzugeben. Diese beiden Auswahlkästchen sind mit einem Bild des Zugstabs und einem Bild der Kreisscheibe grafisch erläutert, also textfrei. Die Darstellung ist geeignet für globalen Einsatz weltweit, weil die Übersetzung in Fremdsprachen entfällt.

Die verschiedenen Hersteller-Systeme werden dem CAD-Ingenieur am Bildschirm in einer Form angeboten, die ihm die Auswahl anschaulich vereinfacht.

Den genannten Schaltflächen [→] wurden bei der Konzeption des Leitbilds programmtechnisch folgende Eigenschaften gegeben:

- Als Schaltflächentyp wurde callback (Aufruf) eingestellt, weil sie eine Dialogbox für Einstellung aufruft.
- Ein zugeordneter Methodenaufruf wurde definiert, der eine Methode (hier: ZGSTPA) ablaufen lässt, die das entsprechende Dialogfenster des ausgewählten Hersteller-Systems bestimmt.
- Der folgende Boxaufruf gibt das durch die Methode bestimmte Fenster an die Nutzoberfläche.
- Ein Funktionsaufruf kopiert den Inhalt eines Knopfes auf den angegebenen Speicher.

Das für diese Arbeit verwendete CAD-System stellt dem Entwickler von Leitbildern ein spezielles Dialogfenster entsprechend *Abbildung 3-5* zur Verfügung. In diesem Dialogfenster werden die o. g. Methodenaufrufe und der Schaltflächentyp festgelegt. Außerdem bietet dieses Dialogfenster die Möglichkeit, modellbehaftete Zeichnungen des CAD-Systems (bmf_ -Format) in modellfreie 2D Darstellungen für Leitbilder zu wandeln (dmf_ -Format).

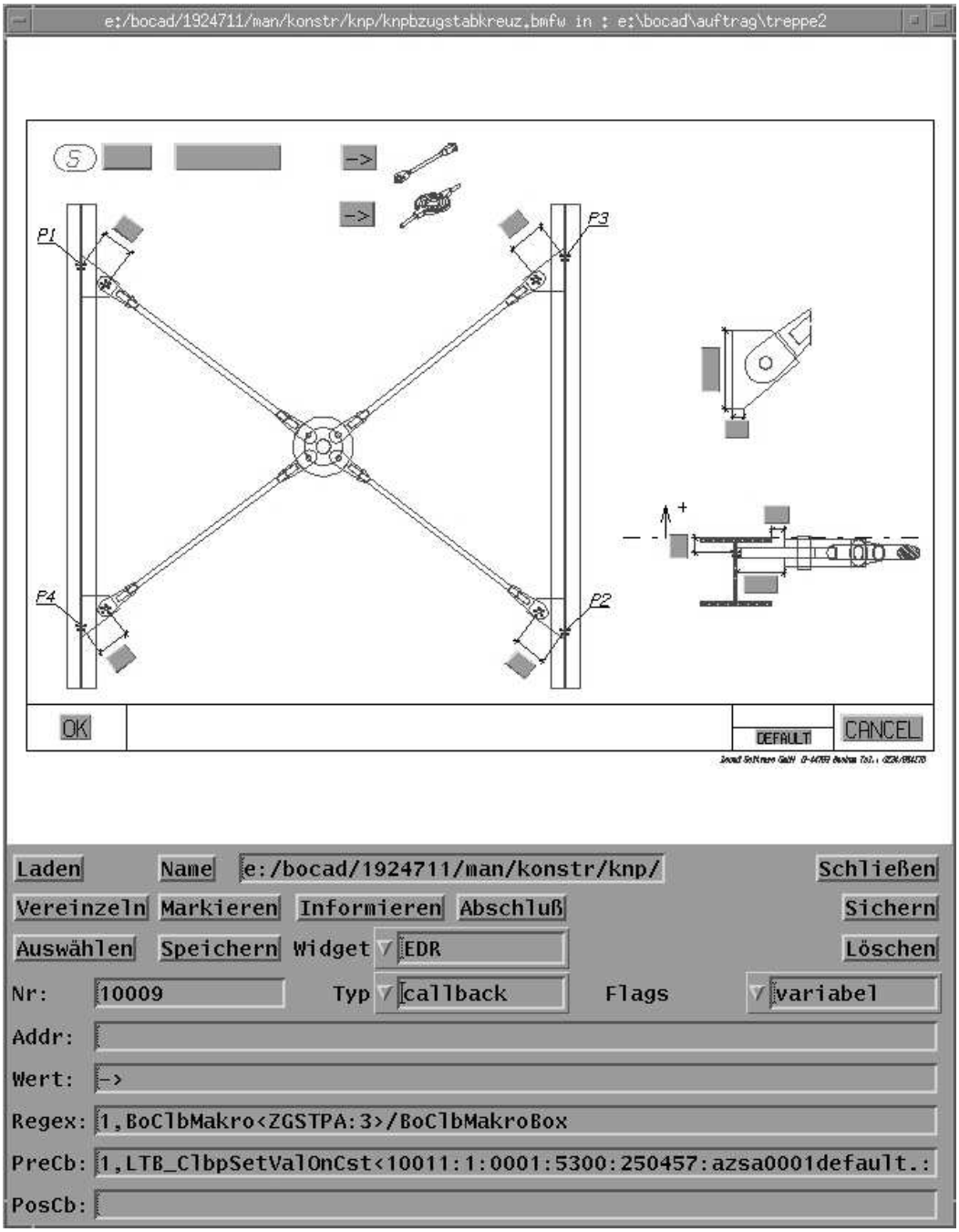


Abbildung 3-5: Dialogfenster für die Entwicklung von Leitbildern

3.6 Aktive Oberfläche mit grafischen Methoden

3.6.1 Zielsetzung

Unter aktiver Oberfläche mit grafischen Methoden wird hier verstanden, dass die grafischen Darstellungen in einem Leitbild nicht statisch unverändert bleiben, sondern sich in Abhängigkeit der Nutzeraktionen dynamisch verändern. Die passive Oberfläche wird so zu einer aktiven, „mitdenkenden“ Oberfläche erweitert.

Mit dem folgenden Beispiel wird dazu im ersten Schritt die Werteeingabe mit einem Eingabefeld in eine 2D-Darstellung vorgestellt, siehe *Abbildung 3-6*. So können Maßkettenwerte wie z. B. Breite oder Länge eines Bauelements mit einem Doppelklick an der Maßkette als änderbare Werte markiert werden. Nach diesem Doppelklick wird ein Dialogfeld für die Eingabe des entsprechenden Wertes geöffnet. Danach wird statt der bisherigen Darstellung des Objekts eine neue Darstellung gezeichnet, die den eingegebenen Wert quantitativ und sinnentsprechend berücksichtigt.

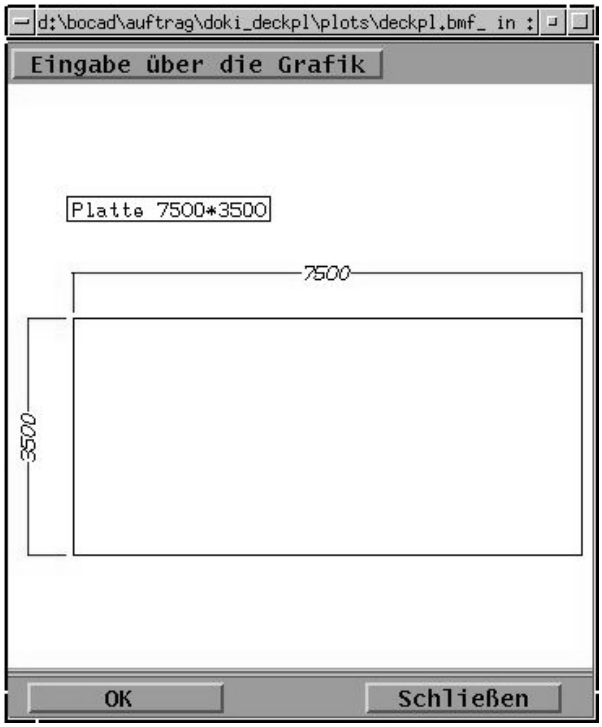


Abbildung 3-6: Dialogfenster für dynamische Eingabe

Dieser Vorgang hat den Vorteil, dass der Anwender die Wirkung seiner Änderungen sofort sieht und durch die Grafik eine hilfreiche, reale Visualisierung bekommt.

3.6.2 Zeichnungserstellung in aktiven Oberflächen

Die in dieser Arbeit entwickelte Lösung für aktive Oberflächen sei an einem sehr einfachen Beispiel erläutert, das in *Abbildung 3-6* dargestellt ist.

Das Dialogfenster enthält ein Feld für Grafik, durch welches auf jede Eingabe des Anwenders eine sofortige Vorschau hervorgerufen wird.

Die Vorschau-Zeichnung wird durch die folgenden Befehle aufgebaut:

```
Z3:
  OPN_BMF( );
    BMF_TEXT( );
    BMF_LINE( );
    BMF_BEMA( ); usw.
  CLS_BMF( );
ENDE:Z3;
```

Der erste Befehl *OPN_BMF* öffnet im Block *Z3* (Zeichnungsbereich) eine Grafikdatei mit der Extension .bmf_. Dann werden parametergesteuert Texte, Linien und Bemaßungen etc. in diese Datei ausgegeben. Mit CLS_BMF wird die Grafikdatei geschlossen.

3.6.3 Aufrufablauf

Der Aufruf von Dialogfenstern hat im verwendeten CAD-System eine komplexe Form, die exemplarisch erläutert wird. Zum Dialogfeld des *Bildes 3-6* lautet der Aufruf:

(1,BoHoClbpCrShowBox<**6000:12351:6374**:72:6502:12351:dp00:plan:default:16:80:0>)

Die im Aufruf genannten Zahlen bedeuten hierbei:

- *6000:12351* ruft die entsprechende Konfigurationsdatei auf,
- *6374* ist der Zeiger auf die in Zeile ------6374 stehende Befehlsfolge der Konfigurationsdatei.

Die wichtigsten Elemente der Befehlsfolge lauten:

```
------63740
#def Klick Eingabe über die Grafik
#def Klickmes Die Werte aendern
#def OK     OK
#def OKmes Methode ausführen
#def Schließen Schließen
#def Schließenmes Abbruch
------6374
[...]
PUB (1057,1050), (1,#Klick ,-), (2,a,-), (#Klickmes), (2,0,0),\
      (1,BoInstallReqGrp<60006374:23510000>)
[...]
PUB (1076,2100),(1, #OK,-),(2,a,-),(#OKmes),(1,0,1),\
      (1,BoHoClbClsShowBox<72>/BoClbMakro<DRAW_DECKPL:2>)
[...]
PUB (1079,2100), (1, #Schließen  ,-), (2,a,-), (#Schließenmes), (1,0,1),\
      (1,BoHoClbClsShowBox<72>)
/*
```

- *Eingabe über die Grafik*: bietet an der Nutzoberfläche die Klickmöglichkeit, die unter Nummer *60006374:23510000* in einem *name.rqi* File definiert wird.
- OK: Schließt das Dialogfenster und führt die Methode DRAW_DECKPL aus.
- *Schließen*: schließt das Fenster.

Damit das Vorschau-Bild selbst anklickbar wird und dort die gewünschten Maßangaben für Länge und Breite als Eingaben erfolgen können, wird ein zugehöriger Request durch die Schaltfläche [Eingabe über die Grafik] aktiviert, siehe *Abbildung 3-6*. Diese Schaltfläche liefert eine Klickmöglichkeit durch den Standardcallback BoInstallReqGrp<NrRequest:TypRequest> in das Grafikfeld. Die Requestnummer (*60006374)* und der Requesttyp *(23510000)* bestimmen die Requestgruppe, die nach

einem Doppelklick an der Grafik durchlaufen wird. Diese Requestgruppe liegt in diesem Fall in der Datei *grafik.rqi*.

```
Begin:60006374,23510000;
 RechteMausCallback:1,BoCreateCombiBox<6500:12351:65213>;
 BeginReq;
  RequesType:2;
  CursorType:101;
  MinElReq:1;
  MessFileNr:7000;
  MessNr:10;
  PickMode:6404;
  Securitylevel:1;
  ErgUsnr:4;
  ErgUstp:18069900;
  PrevUsnr:67020000;
  PrevUstp:28069900;
  CoorSource:30;
  SortListe:1;
  Callback:1,Ni3CallsFFein<DRAW_DECKPL:0>/BoCreateCombiBox<6500:12351:65211>;
 EndReq;
End;
```

Auch hier seien die wichtigsten Anweisungen kurz kommentiert:

- *RechteMausCallback* aktiviert durch das Drücken der rechten Maustaste den Callback. Hier wird ein Dialogfenster aufgerufen.
- *Requesttyp* legt fest, ob ein Primitivum, ein Segment oder eine Gruppe pickbar wird. Hier ist das Segment angegeben.
- *CursorTyp* gibt die Zeigerart an.
- *MessFileNr* legt die dem Namen der Meldungsdatei zugeordnete Nummer fest.
- *MessNr* wählt die Satznummer der Meldung im *MessFileNr* Bereich aus. Die ausgewählte Meldung gibt dem Programmbenutzer einen Hinweis über die zu befolgende Benutzeraktion an der Nutzoberfläche. Die Fremdsprachlichkeit soll auch in diesem Bereich beachtet werden.

- *PickMode* ist der individuelle Pickmode für diesen Request (Text/ Linie/ Teile/ Bemaßung). Hier ist die Bemaßung angegeben.
- *Callback*, ist das Ergebnis der Aktivierung, wenn der Request befriedigt ist. Hier wird der im Fenster des *Bildes 3-6* gewählte (gepickte) Bemaßungswert der Bemaßungsgrafik entnommen und in das Eingabefeld eines neu erscheinenden Dialogfensters (------*65211*) eingetragen, siehe *Abbildung 3-7*.

Der o. g. Callback ruft also das Dialogfenster des *Bildes 3-7* auf. In diesem kann der Anwender eine Änderung der Variablen für die Bemaßung vornehmen, wie durch die nachstehenden Schaltflächendefinitionen geregelt.

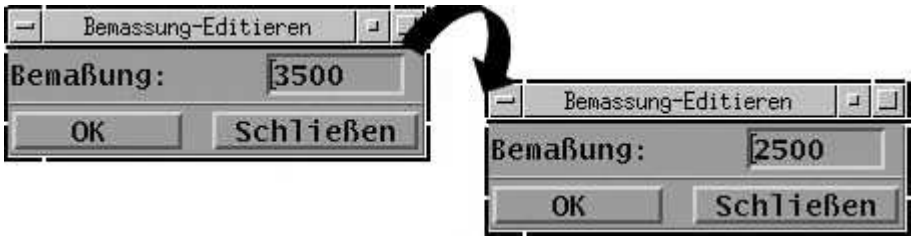


Abbildung 3-7: Änderung der Maßkettenwerte

```
------65211
[…]
PUB (6601,6600), (1, #OK,-), (2,a,-), (#OKmes), (2,0,0),\
    (1,BoHoClbpDoClbLis<6500:12351:65212>)
PUB (6605,6600), (1, #Schließen ,-), (2,a,-), (#Schließenmes),\ (0,0,0),
    (1,BoComboClbCanc/BoInstallReqGrpSec<->)
------65212
BoComboClbOK
BoClbMakro<DRAW_DECKPL:1>
BoHoClbpCrShowBox<6000:12351:6374:72:6502:12351:dp00:plan:default:16:80:0>
BoInstallReqGrpSec<->
------6522
```

Die Schaltfläche [OK] speichert den neu eingegebenen Wert auf "default" und

schließt die Dialogbox, dann lässt die Konstruktionsmethode *DRAW_DECKPL* durchlaufen. Abschließend öffnet das Fenster, das die Grafik enthält, mit der neuen Zeichnung. Die weitere Klickmöglichkeit an der Grafik ist ohne Aktivierung der Schaltfläche [Eingabe über die Grafik] möglich, siehe *Abbildung 3-8*.

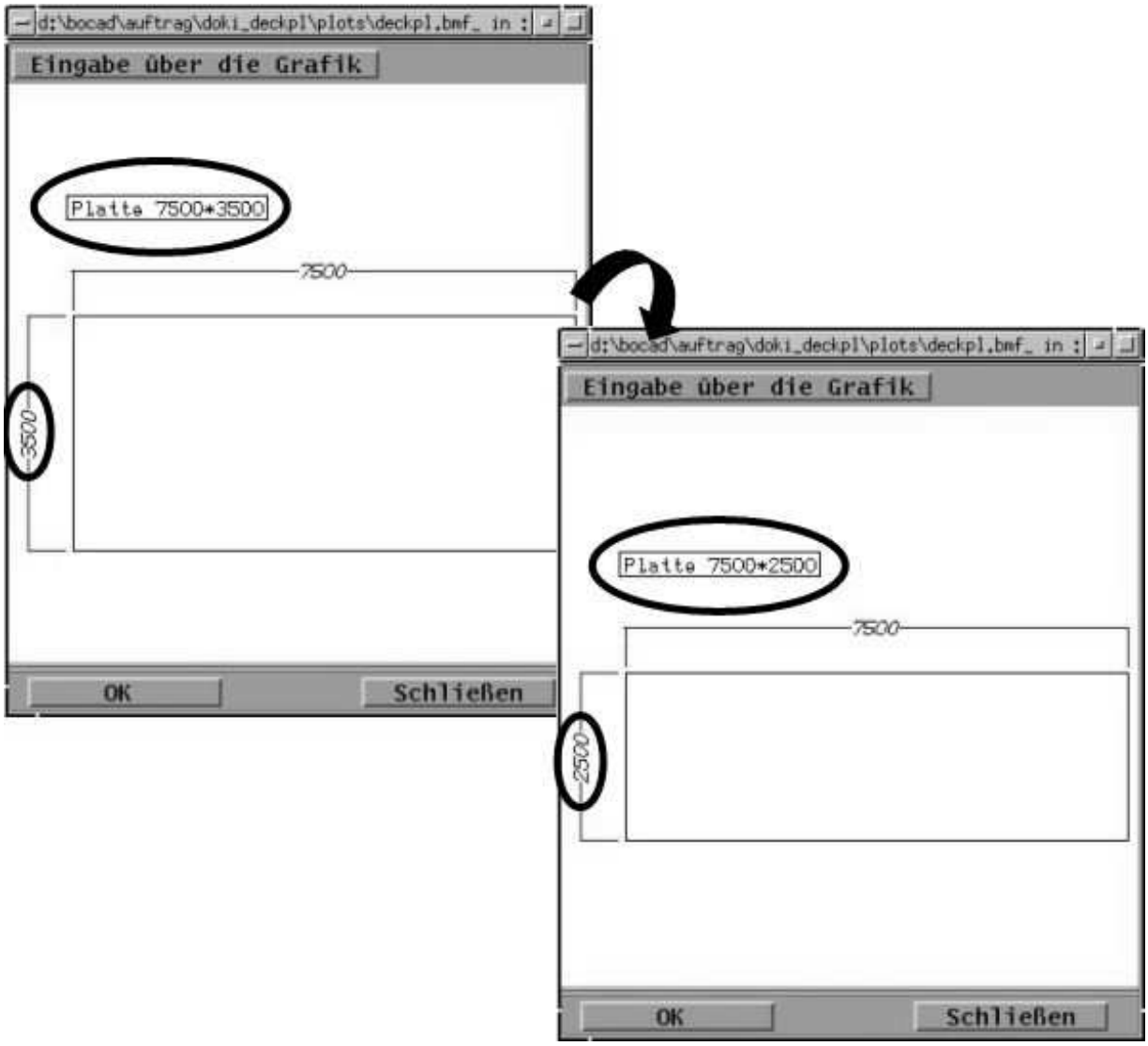


Abbildung 3-8: Vorschau-Aktualisierung durch aktive Nutzoberfläche

3.6.4 Speicher zwischen Nutzoberfläche und CAD-Methoden

Wenn ein Zahlenwert oder ein Text an der Oberfläche geändert wird, muss diese Änderung in CAD-Methoden weitergeleitet werden können. Im Laufe der Element-

auswahl (Klicken) entstehende Daten werden deshalb in einen Speicher (C-Store) geschrieben. Aus diesem Speicher können die Daten von CAD-Methoden aufgerufen werden.

Die Übergabe von Daten aus der Nutzoberfläche in CAD-Methoden wird ebenfalls am Beispiel der Vorschau-Grafik des *Bildes 3-8* erläutert.

Die in Methode *NI3_GETSGID* aufgerufene Methode *ni3_getlse* holt dazu im ersten Schritt die Identifikationsnummern („Identnummern“) der an der Nutzoberfläche gepickten Elemente.

```
NI3_GETSGID(-&mfnr=-1,-unr,-&idout=-1);
@ ------------------------------------------------
@ Segmentnummer auf cst melden.
@ ------------------------------------------------
@ mfnr     Fehlerflagge: -1 alles ok
@ unr      Usernummer
@ idout    CStore Typ 21
@ ------------------------------------------------
 %z mfnr=-1, idcst=-1, idout=-1;
 %m NI3_GETLSE(mfnr,unr,idcst); #WENN, mfnr, #RETURN,;
 %sCst: %a SetDesc:idcst, 3, 2,0,1,
                              2,4,1,
                              2,8,1;
      %a SeparateCol: idcst, 'spalten';
      %z idout=spalten1;
      %a FreeCst: 'spalten'<2,3>;
      %a FreeCst:idcst;
 %sENDE;
```

Der Befehl *SetDesc* setzt drei Deskriptoren für einen Speicher *idcst*, der mit *SeparateCol* auf drei verschiedene Speicher (eine Spalte pro Deskriptor) verteilt ist. Die erste Spalte enthält die Identnummern der Segmente, die zweite die Identnummern der Primitiva und die dritte die Identnummern der Gruppen. Diese Identnummern sind somit von der Nutzoberfläche in die Speicher importiert. In diesem Fall ist nur die sich auf die Segmente (hier: Bemaßung) beziehende Spalte relevant. Die Werte der an-

deren Spalten werden mit *FreeCst* freigegeben, d.h. gelöscht, siehe *Abbildung 3-9*.

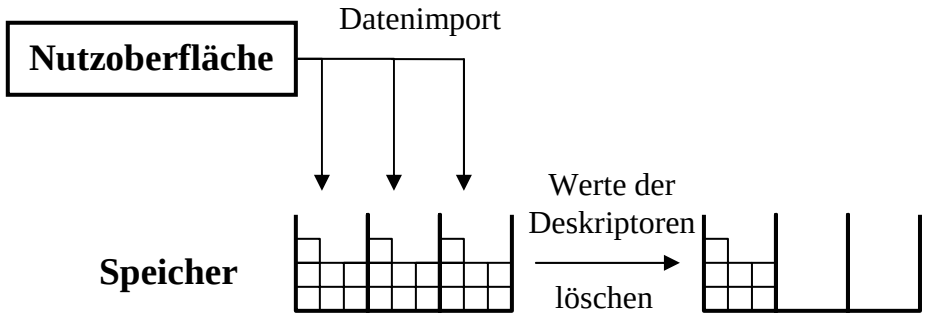


Abbildung 3-9: Datenimport in den Speicher

```
NI3_GETSGNR(-&mfnr=-1,-unr,'name='name');
@ ------------------------------------------------
@ Segmentnummer aus dem OLSE-Cst mit unr abholen
@ der utp ist durch den rqi fest definiert
@ ------------------------------------------------
@ mfnr     Fehlerflagge: -1 alles ok
@ unr      Usernummer
@ name     Feldname
@ ------------------------------------------------
 %z mfnr=-1, id=-1, sind0=0;
 %m NI3_GETSGID(mfnr,unr,id); #WENN, mfnr ' -id, #SPRINGE, RET;
 %sCst: %aExportInt:id,'sind';
       %aFreeCst: id;
 %sENDE;
#RET;
 %sZIELBLOCK:BLOCK(0)-1; %sKOPIERE:'sind'<>,'\name'<>; %sZIELBLOCK;
```

Die Ausgabe einer Integer-Liste des Speichers in die Methode wird mit dem Befehl *ExportInt* gelöst. Dann wird der Speicher freigegeben *(%aFreeCst:id)*, und das erzeugte Wertefeld in der Methode wird mit *ZIELBLOCK* in der Makroverschachtelung eine Ebene höher kopiert, siehe *Abbildung 3-10*.

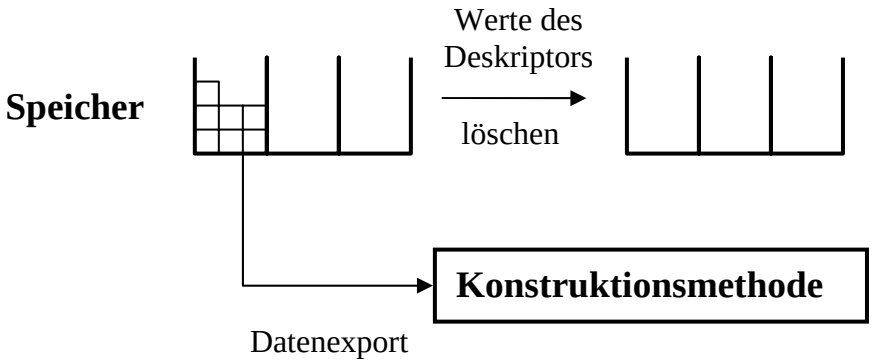


Abbildung 3-10: Datenexport von dem Speicher

3.6.5 Methoden

Das Prinzip von CAD-Methoden sei ebenfalls am Beispiel der Vorschau (*Abbildung 3-6)* kurz dargestellt. Die hierin verwendete CAD-Methode DRAW_DECKPL konstruiert und zeichnet eine rechteckige Deckplatte.

Die o. g. Methode hat nur einen Parameter *param* mit einem Wertebereich von *0* bis *2*, der wie folgt wirkt:

- param=0

  Die Methode liest den angeklickten Bemaßungswert aus der Grafik in das Änderungsfenster ein.

- param=1

  Jeder ausgewählte Wert wird nach der Änderung in der Grafik überschrieben, und die Grafik wird entsprechend dem neuen Wert neu gezeichnet.

- param=2

  Mit den im Speicher übernommenen Werten wird die Konstruktionsmethode ausgeführt.

Die Methode *draw_deckpl* ist also durch die *param* Parameterwerte auf 3 Arten nutzbar. Im ersten Schritt werden die gewünschten Eigenschaften der angeklickten Elemente eingelesen (Untermethode NI3_GETSGNR) und gegebenenfalls die gegenüber der Voreinstellung geänderten Werte übernommen. Anschließend wird in der Methode eine Zeichnung erzeugt, die sich aus Bemaßungen, Texten und Linien zusammensetzt.

Der Zeichnungsprozess beginnt mit der Interpretation der Datei *bodraw_start*, in der der Maßstab und der Name der zu erzeugenden Grafikdatei definiert werden. Die einzelnen in Methode draw_deckpl aufgerufenen Zeichnungsmethoden (*bmf_line, bmf_bema, bmf_text*) bilden die einzelnen Segmente der 2D-Grafik. Die Segmente erhalten im Programmablauf fortlaufende Segmentindizes zur Identifizierung. Das Schließen des oben geöffneten grafischen Metafiles wird durch die Methode *bodraw_end* durchgeführt.

(Siehe Quelltextausschnitte der Methode DRAW_DECKPL() in *Anhang 3.6*.)

Quellcode BODRAW_START:

```
%sCONTINUE;
 %aZ3:
  %aSTEUERUNG: %aINITIALISIERE;
     %aMASZSTAB: 1 : 10, 1 : 10;
  %sENDE;
  %z m$_dir='plots/';
  %mOPN_BMF('deckpl',0,-1,+1);
```

Quellcode BODRAW_END:

```
%sCONTINUE;
 %mCLS_BMF;
%sENDE:z3;
```

Das folgende Flussdiagramm stellt eine Übersicht über den Änderungsablauf an der aktiven Oberfläche dar, siehe *Abbildung 3-11*.

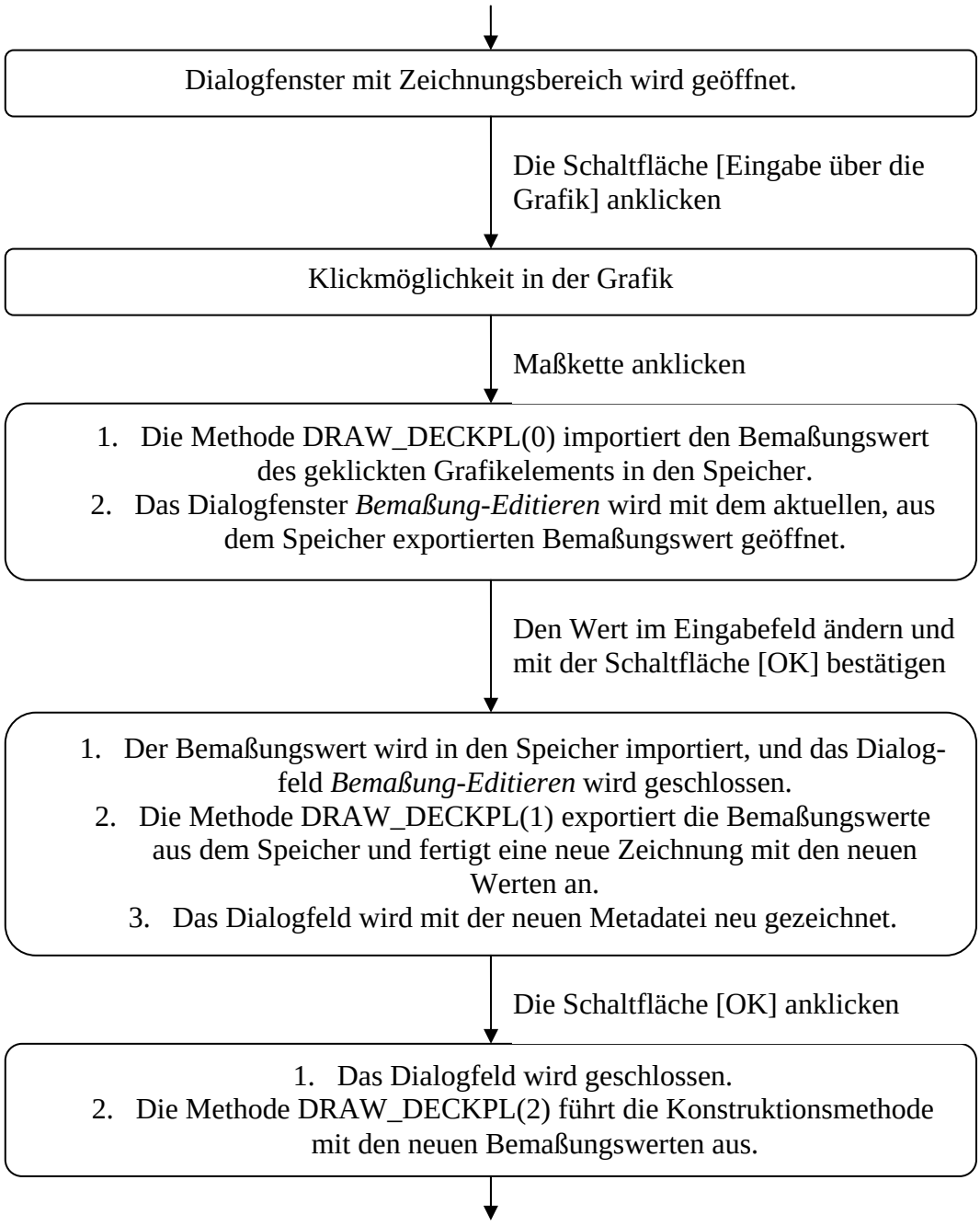


Abbildung 3-11: Änderungsablauf an der aktiven Oberfläche

4 Theoretische Methoden der Konstruktion für neue Märkte und Redesign von Konstruktionsmethoden

Die Denkschrift Bauinformatik [5] beschreibt den umfassenden Einfluss von computerorientierten Methoden auf die verschiedenen Unternehmen des Bauwesens.

„Einzelne Unternehmen der Bauwirtschaft nehmen direkten Einfluss auf die Entwicklung geeigneter Software. Der Eigenbedarf an Informations- und Kommunikationstechnik wird systematisch untersucht. Für das Management und die Bewirtschaftung der Betriebe, Projekte und Baustellen, für die administrativen und technischen Büros sowie für die Infrastruktur und die Unternehmensführung wird die erforderliche Software bei Bedarf entwickelt. Dabei werden gleichzeitig die Arbeitsprozesse, die Entscheidungsprozesse, der Informationsfluss und der Kommunikationsbedarf analysiert und verbessert. Die entwickelte Software ist auf die spezifischen Belange der Unternehmen ausgerichtet.“

Um das Konstruieren mit Theoretischen Methoden zu unterstützen, die hier CAD-Methoden genannt werden, bieten die CAD-Hersteller zumindest einen Grundstock an branschenspezifischen CAD-Methoden an. Im Laufe von Jahrzehnten wurden so bedarfsgesteuert Bibliotheken von CAD-Methoden für Stahlbau, Holzbau, Metallbau und Glasbau entwickelt, die die Leistungsfähigkeit des hier verwendeten CAD-Systems prägen.

Wie nach [14] erklärt wurde, die „Analyse allgemein üblicher Anschlussformen im Stahlbau ergibt folgende Unterteilung in Konstruktionsfamilien mit gemeinsamen Familienmerkmalen:

- Anbringen eines Anbauteils oder Verbindung an einen Träger ohne geometrischen Einfluss eines Nachbarträgers (Stützenfüße, Steifen, Kopfbolzendübel),
- Detaillierung eines Knotens, an dem nur zwei Träger beteiligt sind (häufigster auftretender Fall),

- Detaillierung eines Knotens, an dem mehr als zwei Träger beteiligt sind,
- Sonderformen.“

Jede Konstruktionsfamilie ist im verwendeten CAD-System als Menü angeboten. Die ersten zwei Konstruktionsfamilien werden in dieser Arbeit mit neuen Konstruktionsmethoden ergänzt, um neue Märkte zu erschließen. Dazu muss die Konstruktionslogik entsprechender Produkte erkannt, ingenieurwissenschaftlich analysiert und automatisierbar formuliert werden. „Das Konzept zur Erstellung von Konstruktionslogik basierte auf Annahme, dass es Gesetzmäßigkeiten im Konstruktionsvorgang gibt, die allgemeingültig sind. Die Vermutung begründete sich aus bereits erstellten Konstruktionsmethoden, die gleiche bzw. ähnliche Logik enthielten.“

Die Entwicklung einer praxistauglichen Konstruktionsmethode setzt eine lückenlos durchdachte Methodenstruktur voraus. Spezifikationen und Voreinstellungen sowie Gestaltungsregeln der beteiligten Objekte und Ihrer Verbindungen müssen dazu erkannt und analysiert werden.

Bei ingenieurwissenschaftlichen Aufgabenstellungen der Bauinformatik wird Software in der Regel nicht völlig neu entwickelt, sondern auf bestehenden Bausteinen aufgebaut, die in der Baupraxis im Alltagseinsatz stehen. Stabilität und Zuverlässigkeit dieser im Einsatz stehenden Bausteine darf selbstverständlich nicht gefährdet werden, so dass ein Software-Entwickler nicht die Originale, sondern nur die Kopien der Produktionsversion ändern darf. Erst nach Abschluss der Entwicklung und der sorgfältigen Qualitätsprüfung werden die Kopien als neue Version in die Produktionsversion integriert. Diese wird dann zunächst nur von ausgewählten Pilotanwendern eingesetzt, die experimentell die Richtigkeit nachweisen bzw. Fehler aufdecken. Erst nach erfolgreicher Pilotanwendung wird die neue Version zur allgemeinen Produktionsversion.

4.1 Neue Konstruktionsmethoden für neue Märkte

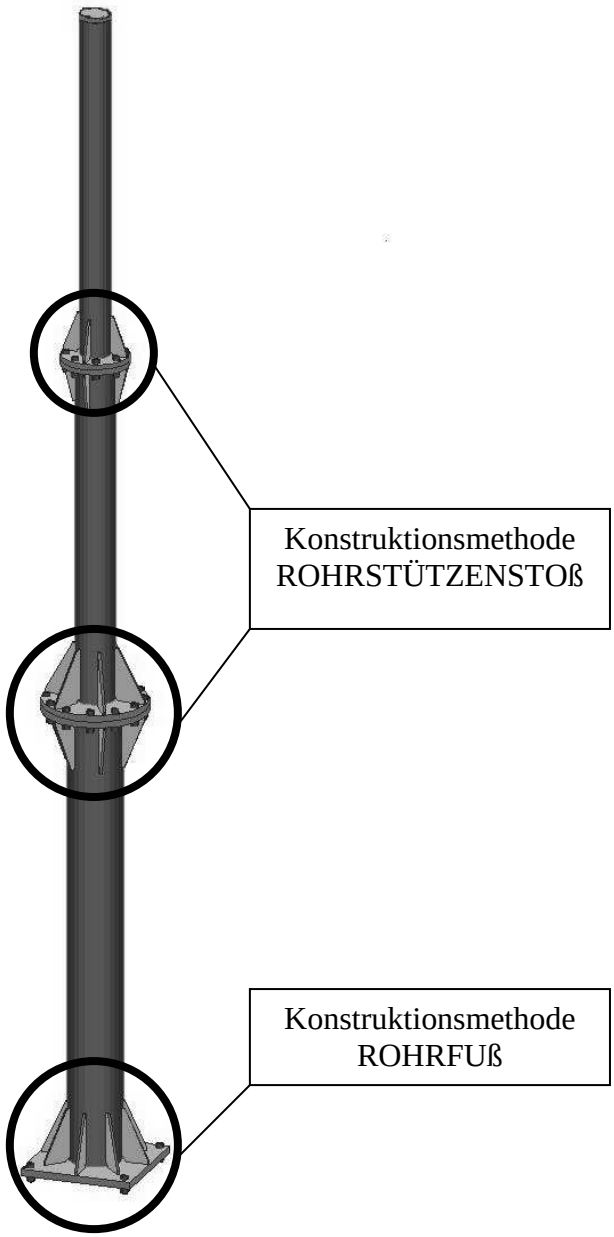


Abbildung 4-1: Typische, individuell detaillierte Mobilfunkantenne - 3D CAD-Zeichnung

Durch die starke, flächendeckende Verbreitung des Mobilfunks, aktuell verstärkt durch UMTS, ist für das rechnergestützte Konstruieren im Stahlbau ein interessanter neuer Markt entstanden: der Bau von Tausenden von Kompaktantennen mit System. Da die Aufstellbedingungen für derartige Antennen sehr stark variieren, ist jede Antenne individuell zu entwerfen und zu konstruieren, jedoch wiederholen sich konstruktive Details und das Konstruktionsprinzip. Dieser Fall ist zur Darstellung der theoretischen Methoden ideal zu präsentieren. Die Konzeption, das Design und die Codierung von neuen Methoden werden daher im Folgenden an diesem Anwendungsfall erläutert.

Das Bausystem derartiger Antennen ist in Musterentwürfen definiert. Aus diesen Unterlagen und weiteren Klärungsgesprächen musste die Konstruktionslogik des Bausystems hergeleitet werden. Die von der CAD-Methode automatisch zu lösende Konstruktionsaufgabe gliedert sich in Teilaufgaben mit definierten Schnittstellen, z. B. die Konstruktion des Antennenfußes und Stöße in Antennenschaft, siehe *Abbildung 4-1*.

Die Flussdiagramme, *Abbildung 4-3* und *Abbildung 4-6,* geben einen Überblick über die Hauptkomponenten der hier entwickelten CAD-Methoden für das automatische Konstruieren von Antennen. Es folgt exemplarisch die detaillierte Diskussion der CAD-Methoden für die Teilaufgaben ROHRFUß und ROHRSTÜTZENSTOß.

4.1.1 CAD-Methode ROHRFUß

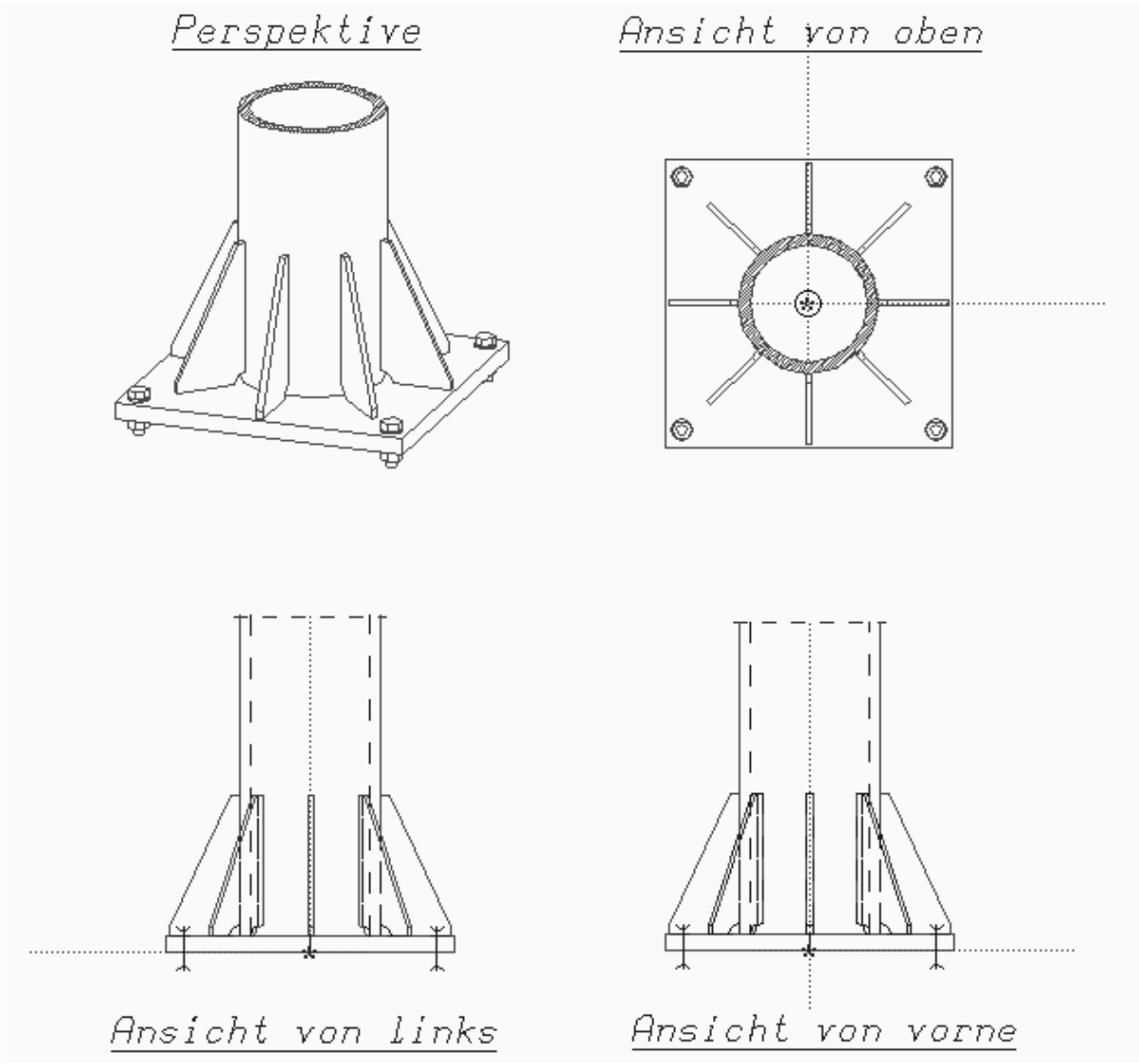


Abbildung 4-2: CAD-Zeichnung des Rohrfußanschlusses

4.1.1.1 Grundlagen

Mit der hier entwickelten Bauart ROHRFUSZ1 der Methodenfamilie ANBAU können ausgesteifte Fußplattenkonstruktionen für Rohre - insbesondere für den Antennenbau - konstruiert werden. Die Anzahl und die Gestaltung der Steifen lassen sich individuell steuern. Ebenso stehen für die Anordnung der Schrauben- bzw. Bohrungen-

Parameter zur Verfügung.

4.1.1.2 Methodenstruktur

CAD-Methoden mit gemeinsamen Grundprinzipien werden sinnvoll zu Methodenfamilien zusammengefasst, um für gemeinsame Konstruktionslogik stets dieselben Untermethoden zu verwenden. So bilden alle CAD-Methoden, die Anbauteile an ein Leitteil anfügen, die Familie ANBAU. Die spezielle Bauart von Stützefüßen für Antennen war die 31. Methode des Prinzips ANBAU. Diese Methode erhielt daher den internen Namen *ANBAU31*.

Die Spezifikationen der Bauart werden in der Methode *voran31* behandelt. Die vom Programmbenutzer wählbaren Spezifikationen stehen dabei in der Datei *v_anb_31*.

Die Fußblechgestaltung beginnt mit der Bestimmung der notwendigen Minimalabstände der Bohrungspunkte. Die Generierung der Platte und der Verbindungen, das Schweißen sowie die Loch- und die Schraubengenerierung folgen. Mit dem voreingestellten Schraubentyp HVMS wird in *(Methode fusz31)* die Schraubenstruktur um eine Sicherungsmutter nach DIN 7967 erweitert. Dazu ist die systemeigene Schraubendatenbank unter dem Namen *HVMS* um einen neuen Schraubentyp zu ergänzen.

Das Erzeugen der einzelnen Steifen über die Steifenkonturpunkte und das Erzeugen der Schweißnähte zwischen dem Fußblech und den Steifen in Methode *fsteife1* sowie die Naht zwischen Fußplatte und Rohrstütze in Methode *anbau31* vollendet die Konstruktion, s*iehe Abbildung 4-3.*

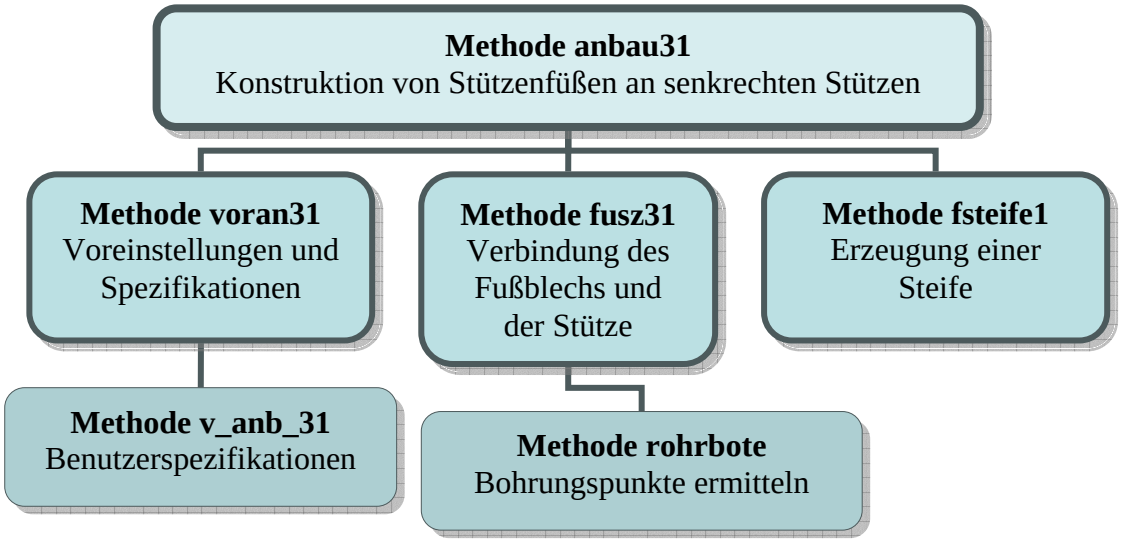


Abbildung 4-3: Konstruktionsmethodenstruktur, Beispiel Rohrfuß

4.1.1.3 Leitbild

Nach den in *Kapitel 3.5* erläuterten Prinzipien wurde für ein geeignetes Leitbild Antennenfüße entwickelt. Die inneren Eckabschnitte der Steifen werden mit dickeren Linien dargestellt. Die Steifen am Leitbild haben von der Realität abweichende Eigenschaften - vergrößerte Dicke und Eckabschnitte -, um die einzugebenden Konstruktionseigenschaften einfacher verständlich zu machen.

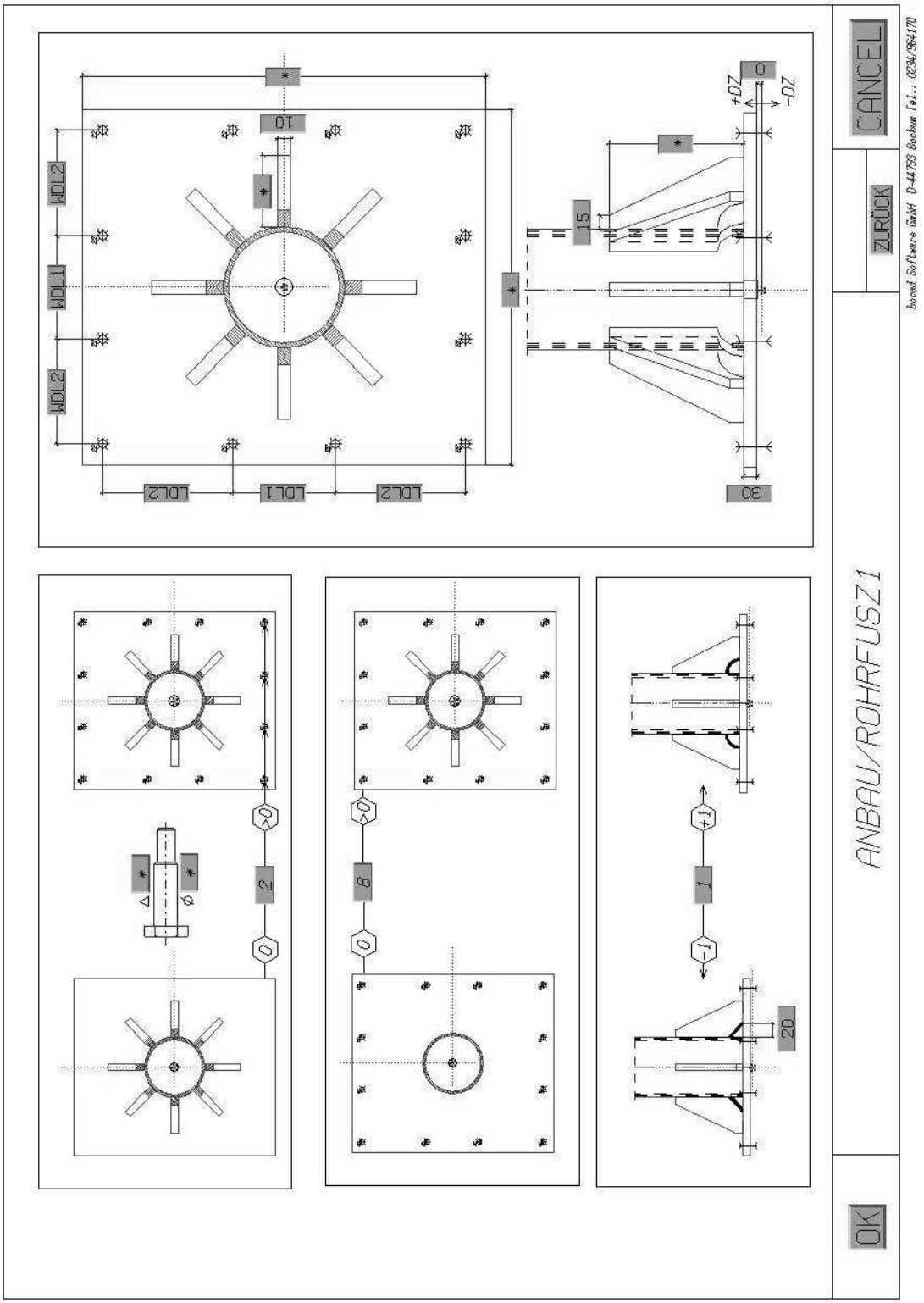


Abbildung 4-4: Rohrfußleitbild

4.1.2 CAD-Methode ROHRSTÜTZENSTOß

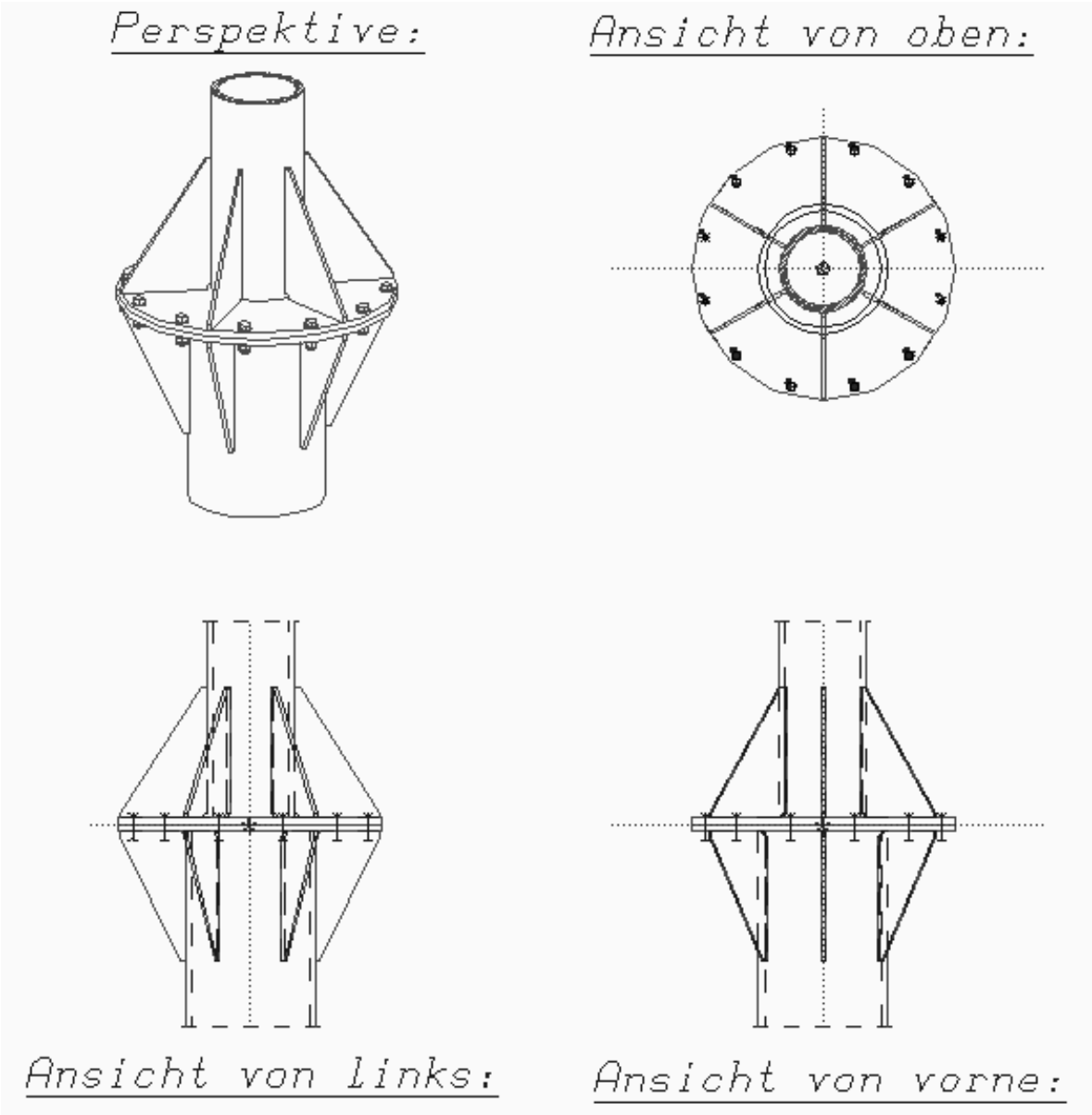


Abbildung 4-5: CAD-Darstellung des Stützenstoßanschlusses

4.1.2.1 Grundlagen

Mit der Methode ROHRSTSTOSZ1 innerhalb der Methodenfamilie KNOTEN werden ausgesteifte Stöße zwischen den Schüssen eines Antennenmasts für Rohrträger erzeugt. Für die Rohrflansche können entweder Blindflansche nach DIN 2527 ausge-

wählt werden, oder es sind frei wählbare Abmessungen anzugeben. Die Anzahl und die Geometrie der Steifen lassen sich wie beim Fußanschluss individuell steuern. Ebenso stehen für die Anordnung der Schrauben- bzw. Bohrungen- Parameter zur Verfügung.

4.1.2.2 Methodenstruktur

Die Spezifikationen und Voreinstellungen wurden in der Methode *vorkt69* wegen der Einheitlichkeit und der einfachen Nutzung so eingestellt, dass die hier angewandten Namen der Variablen mit den Variablennamen des Rohrfußkonstruktionstyps übereinstimmen.

Wenn die Kopfplatten platziert werden, tauchen Probleme auf, die aus der Darstellung eines abgerundeten Teils stammen. Der Grund hierfür ist, dass der Kreis im rechnerinternen Modell als Vieleck abgebildet wird. So definiert das Programm den Durchmesser eines Kreises nicht direkt, siehe *Abbildung 4-7*. Daher ist es sinnvoll, Punktobjekten die Eigenschaft „Kennung“ und entsprechenden Punkten die Kennung für Kreissegment- Anfang und -Ende zu geben. Dieses Problem wird mit den Befehlen ZWANGSKONTUR und KENNUNG gelöst.

```
@ Kopfplatten verlegen
@ ---------------------
 %aEBENE: 7-99-9;
 %aBESCHREIBUNG:ZWANGSKONTUR;
   $PNR,BENPL,'BL'TPL,QU,[<1500,1499+AUSR>],MITTIG,OBER,VORN;
   %zIDPL1=%fNTLI(0);
   $PNR,BENPL,'BL'TPL,QU,[<1500,1499+AUSR>],MITTIG,OBER,HINTEN;
   %zIDPL2=%fNTLI(0);
@
 %aEDITIERE: %aSETZETEIL:IDPL1,IDPL2;
          %aKENNUNG:,210891;
          %sENDE;
```

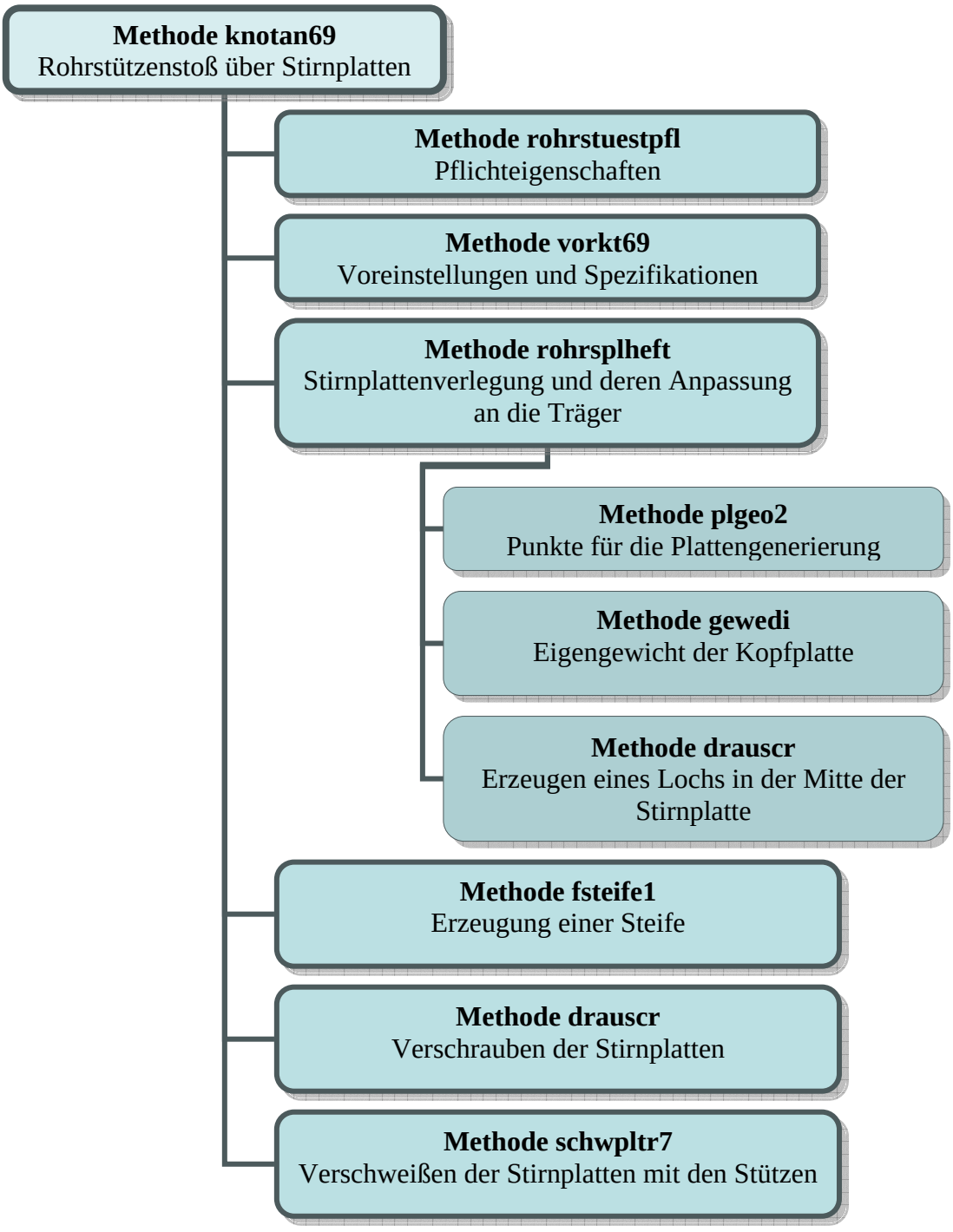


Abbildung 4-6: Struktur der Konstruktionsmethode Rohrstützenstoß

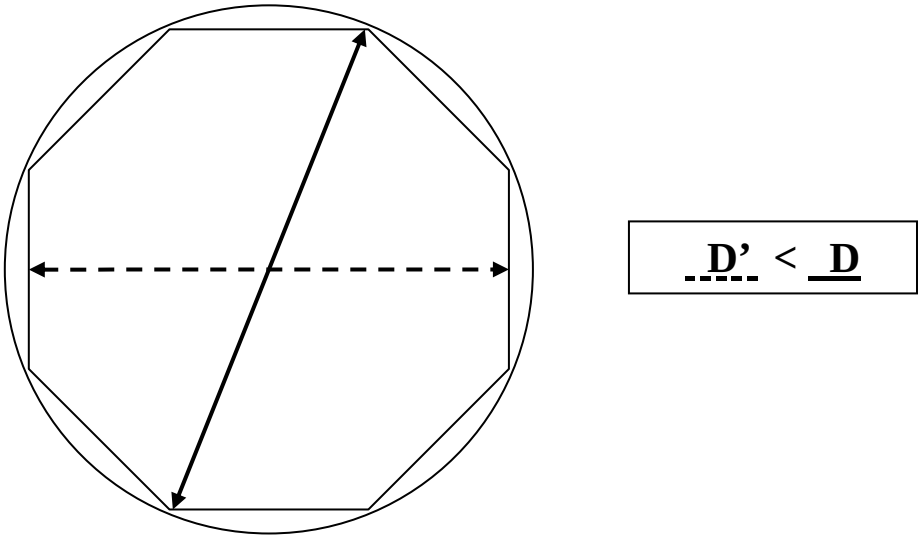


Abbildung 4-7: Problematik der Durchmesserbestimmung bei der rechnerinternen Kreiserzeugung

Für das Erzeugen des Lochs in der Flanschmitte, das zur Kabeldurchführung dient, wurde nicht mit Hilfe eines Verschneidungskörpers erzeugt, sondern als Objekt *Schraubenverbindung* ohne Schraube gewählt. Die Objekteigenschaften und Methoden von Schrauben sind bei später erforderlichen Ergänzungen oder Änderungen für derartige Bohrungen geeignet, z. B. auch beim Verschieben oder Löschen. Siehe *Abbildung 4-6*.

4.1.2.3 Leitbild für den Stoß eines Antennenmastes

Das konzipierte Dialogfenster wird über die Schaltfläche [Tabelle] im Leitbild (siehe *Abbildung 4-9*) geöffnet. Durch die integrierte Tabellenanwendung unterscheidet sich dieses Dialogfenster von dem zuvor diskutierten Einführungsbeispiel für Leitbilder.

Abbildung 4-8: Dialogfenster *Tabelle*

Wenn das Optionsfeld *Tabellenanwendung* in *Abbildung 4-8* eingeschaltet ist, werden die in dem Fenster „Tabelle“ vorgenommenen Einstellungen – wie Nennweite und Nenndruck – berücksichtigt.

Beim Aufbau des Fensters *Tabelle* wurde die Frage der Mehrsprachigkeit beachtet. Die Textelemente, die durch Variablen bestimmt wurden, wurden in den Konfigurationsdateien der entsprechenden Sprache definiert.

Quelltext des Fensters *Tabelle*:

```
/*
------57000
#def Titel Tabelle
#def NW Nennweite
#def ND Nenndruck
#def Opt Tabellenanwendung
------5700
#in 6000 12351 57000
IBO (1)
BOX (1,0),(1, #Titel,-),(0,0),(+230+5),(1,BoComboClbCanc)
RCL (2,1),(1,für icons,-),(1,1,1),(7)
TBL (1000,  2),          (1,für icons,-),(1,1,1),(7)
LAB (100,1000,1,2),(1,  ,-)
CBC (101,1000,1,3,8), (1,#NW ,-), (5700, 250457, nenn, 0101, default , 16, 8), (5700, 250457, nenn, 0101, default , 16, 8), (5700,250457,13), (5,8), (0,0)
LAB (102,1000,2,2),(1,  ,-)
CBC (103,1000,2,3,8), (1,#ND ,-), (5700, 250457, nenn, 0101, default , 24, 8), (5700, 250457, nenn, 0101, default , 24, 8), (5700,250457,12),(5,8),(0,0)
```

```
LAB (104,1000,3,2),(1,  ,-)
TOG (105,1000,3,3,8), (1,#Opt ,-), (5700, 250457, nenn, 0102, default , 16, 4), (5700, 250457, nenn, 0102, default , 16, 4)
/*
#in 6000 12351 5310
------1234567890
```

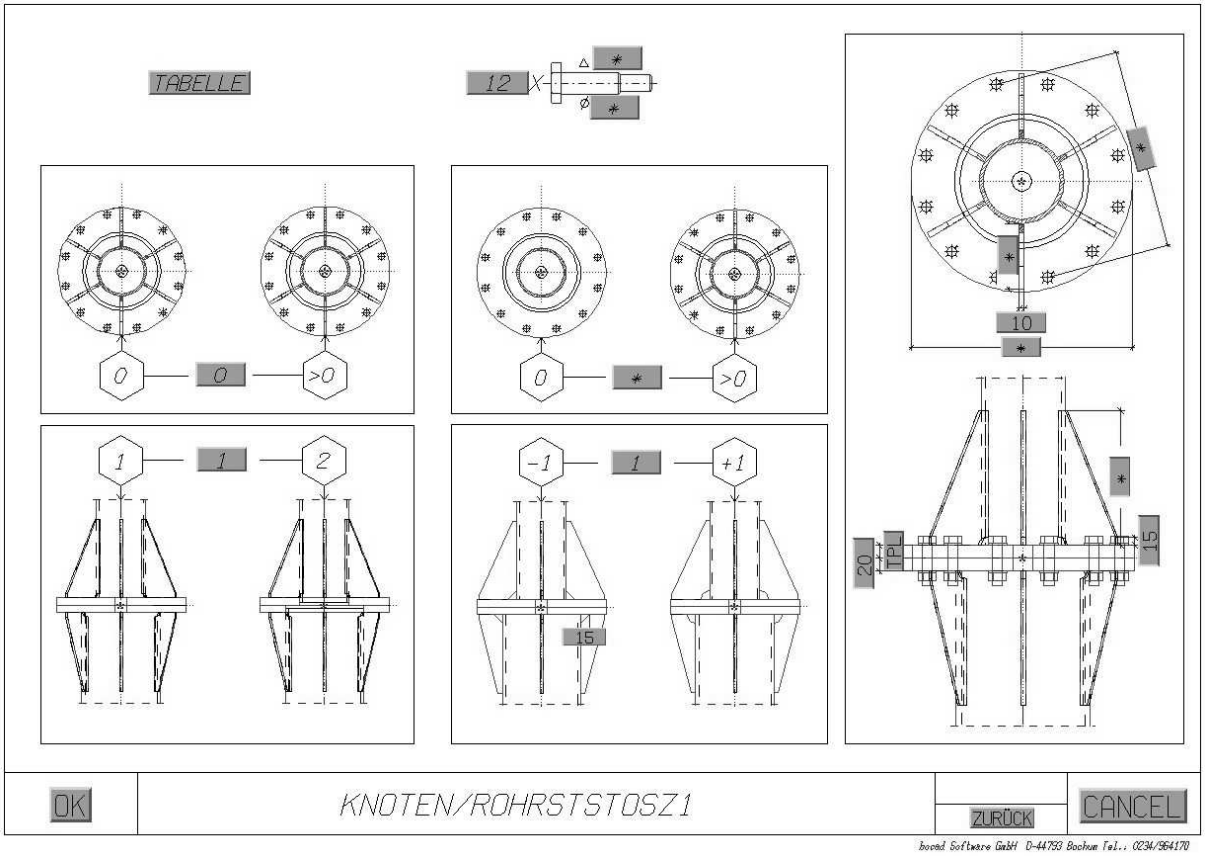


Abbildung 4-9: Zwischenstand des Leitbildes *Rohrstützenstoß* mit Tabellendaten

Der in der *Abbildung 4-9* dargestellte Zwischenstand des Leitbildes wurde mit realen, maßstäblichen Größen der Konstruktionselemente angefertigt. In diesem Zwischenstand besitzt es noch nicht die beim Leitbild des Rohrfußes, siehe *Abbildung 4-4*, angewandten Vorteile eines Leitbildes, deshalb sind die Steifenparameter schwer erkennbar.

4.2 Redesign von Methoden

Die Forschung und die Entwicklung im Bauwesen verlagern sich mit steigendem Baubestand mehr und mehr vom Spektrum der Neubauten zum Spektrum des Bauens im Bestand. Die gleiche Tendenz gilt analog bei steigendem Bestand an Softwareprogrammen und Bausteinen für die Forschung in der Bauinformatik. Der Bestand an Software ist laufend auf den neuesten Stand von Forschung und Entwicklung zu bringen. Dieser Prozess wird Redesign genannt. An einem typischen Fall wird nun Redesign von Konstruktionsmethoden erläutert.

Die Anwendungserfahrungen der Baupraxis, die steigenden Anforderungen und die Technologiesprünge machen das Redesign von Methoden erforderlich. So wurde ein Problem in der Methode zur Stirnplattenerzeugung zwischen zwei Trägern erkannt, nachdem statt der idealisierten die reale Profilausrundung zwischen Flansch und Steg von Trägerprofilen bei der zeichnerischen Darstellung von Profilen gefordert worden war.

Nahezu zwei Jahrzehnte hatte es gedauert, die Ausrundung von Profilen in den CAD-Methoden konstruktiv zu berücksichtigen, obwohl sie in den 3D-Modellen zur Reduktion des Speicheraufwands nicht dargestellt werden. Seit einigen Jahren fordern jedoch die Architekturbüros von den Konstruktionsbüros des Bauwesens die Detailzeichnungen mit Darstellung der Ausrundungen.

Nach der Einführung der realen Ausrundung tritt folgendes Problem auf, *Abbildung 4-10*: Nach dem Klinken sieht die Schnittkontur, bezogen auf die Mittelachse des Nebenträgers, nicht symmetrisch aus. Deshalb sind die Oberkanten der Nebenträger, die an den Hauptträger anschließen, nicht auf einer Ebene.

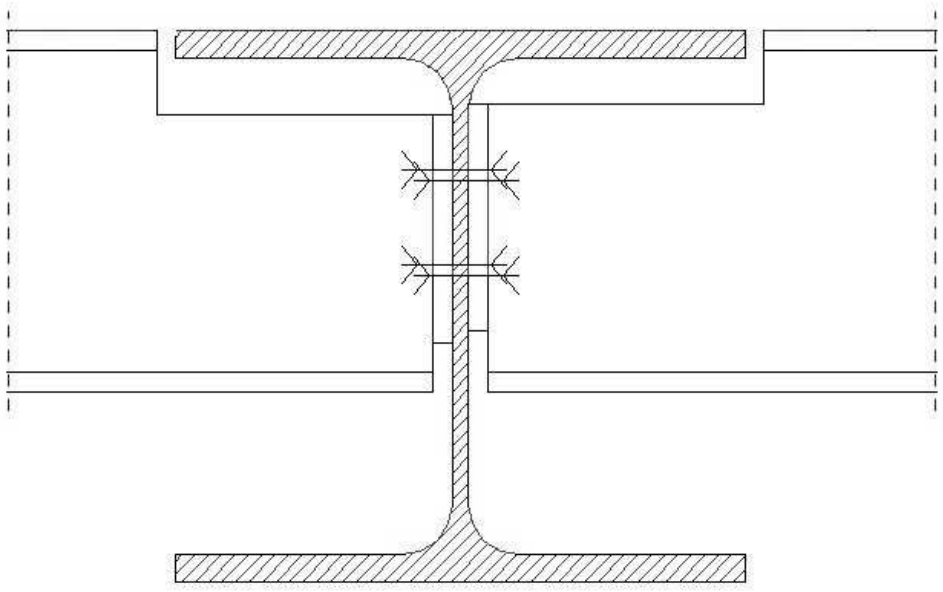


Abbildung 4-10: Fehlerhaft unsymmetrische Ausklinkungen

Die Suche der Fehlerursache ist der erste Schritt beim Ablauf des Redesigns. Dabei stehen Test- und Ortungshilfen dem Entwickler für das Redesign zur Verfügung, wie in *Kapitel 2.15* erläutert.

Die Konstruktionsmethode erzeugt eine virtuelle Schnittfläche. (Der Hintergrund der virtuellen Ebenenerzeugung ist in *Kapitel 2.15* erläutert.) Diese virtuelle Schnittfläche, die die Schnittpunkte für die weiteren Konstruktionsschritte bestimmt, wurde bislang am Steg des Hauptträgers erzeugt. Beim I-Träger mit Ausrundung ist es für das Redesign nicht empfehlenswert, die Schnittfläche weiterhin am Steg zu erzeugen. Die Flanschpunkte in der Ecke wären wegen des Ausrundungs-Polygonzugs nicht eindeutig ermittelbar. Die Verschiebung der Schnittfläche um einen geringen Abstand bietet die einfachste Lösung für ein in allen Situationen korrektes Redesign.

5 CAD-Methodenrecorder und dessen Weiterentwicklung, der Logikrekorder

5.1 CAD-Methodenrecorder als Entwicklungswerkzeug

5.1.1 Zielsetzung

Der hier konzipierte und entwickelte Methodenrecorder hat die Zielsetzung, CAD-Methoden über interaktiv vorgenommene Konstruktionsschritte und Ressource-Dateien - ähnlich zu dem von der Sprache Visual Basic bekannten *Makrorekorder* - zu erstellen. Das Prinzip des Recorders ist das Speichern von Methoden und interaktiven Aktionen, die dann an anderer Stelle - auch mit veränderten Parametern - wiederholt genutzt werden können.

Grundsätzlich werden alle interaktiven Generierungs- und Detaillierungsschritte über den Recorder als Ablaufprotokoll der einzelnen Aktionen des Entwicklungsingenieurs erzeugt. Die prinzipiell sinnvollen Konstruktionsebenen der Profiltypen sind dazu vorab definiert. Sie werden über den Recorder von Fall zu Fall ausgewählt.

CAD-Methoden sind erweiterungsfähiger als Ablage-Kopien von selbstkonstruierten Anschlüssen. Bei Ablage-Kopien werden identisch Anschlüsse an weitere Bedarfsstellen kopiert. Sie sind wegen der systembedingten Identität unflexibel. Bei CAD-Methoden des Recorders ist hingegen die nachträgliche Änderungsmöglichkeit das Hauptziel. Gestaltvariablen sind definierbar, und die Konstruktionsschritte der Konstruktionslogik können beliebig editiert sowie hinzugefügt oder auch gelöscht werden.

5.1.2 Nutzoberfläche des vorhandenen systemeigenen Recorders

Die Softwarefirmen des Bauwesens, die zunehmend neue Märkte erobern müssen, um zu überleben, brauchen immer wieder neue Arbeitskräfte, die die Anforderungen der Märkte möglichst ohne Programmiersprachkenntnisse und sogar ohne die softwarespezifischen Kenntnisse CAD-orientierter Makrosprachen wirksam erfüllen können. Für dieses Ziel wurde eine Nutzoberfläche des Recorders entwickelt, siehe *Abbildung 5-1*.

Das Dialogfenster wird mit einem Klick auf der Symbolleiste *(*siehe *Abbildung 5-3)* mittels linker Taste aktiviert, siehe *Abbildung 5-1.*

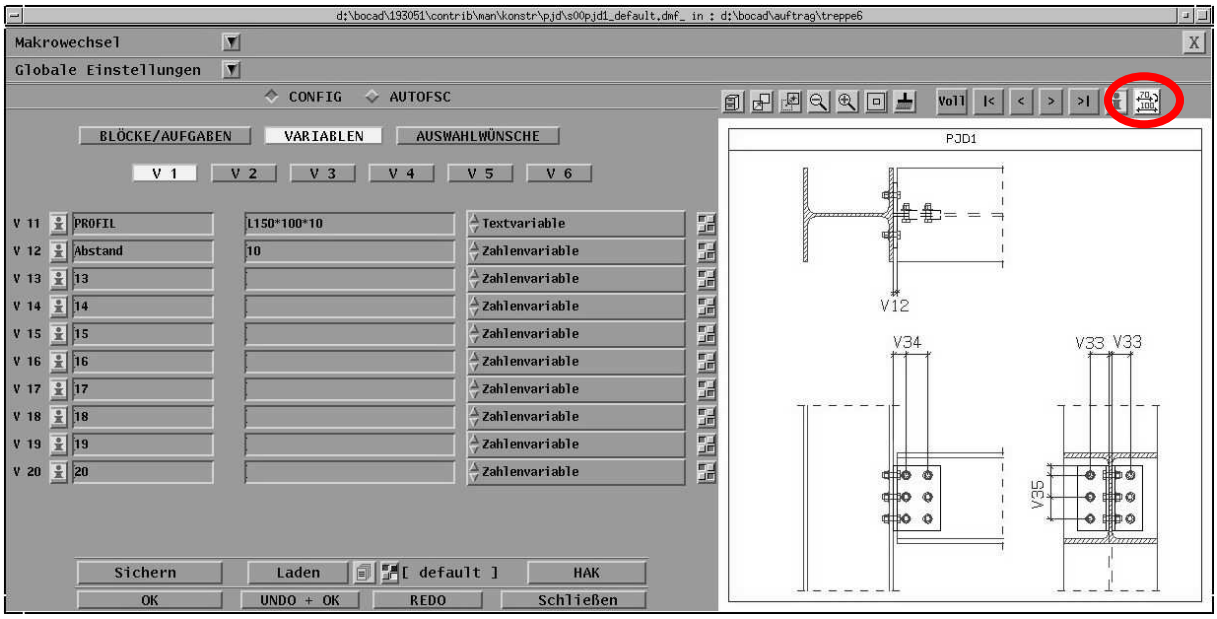


Abbildung 5-1: Dialogfenster eines CAD-Methodenrecorders

Rechte Seite des Fensters entspricht einem Leitbild zu der entwickelten Methode. Die zu den dargestellten Maßketten gehörigen Variablen werden automatisch mit V (Variable) und einer Zahl zwischen 11-60 benannt. Die Variableneigenschaften sind im linken Fensterteil wählbar. Die Änderung einer Variablen wird gleichzeitig im linken Fensterteil und im rechten Fensterteil, dem Leitbild dargestellt.

Mit dem Symbol *Bemaßung editieren* können Abmessungen im Leitbild des Dialogfensters geändert werden. Der Klick auf die Maßkette öffnet das Bemaßungseingabefenster, in dem die Änderung, wie im *Kapitel Aktive Oberfläche 3.6.3* erläutert, durchgeführt wird. Nach dem Schließen des Bemaßungseingabefensters erscheint der neue Variablenwert auch in der linken Fensterseite.

Der Hintergrund ist der Folgende:

Alle Abbildungselemente (Segmente) des Bildes haben eine eigene Identnummer, der ein eigener Speicherplatz zugeordnet ist. Nach dem Klick der Maßkette werden die entsprechenden Werte aus dem dazugehörigen Speicherplatz eingelesen. Der Wert aus diesem Speicher wird in der linken Dialogfensterteil kopiert. Nach der Wertänderung der Variablen ist der neue Wert im Maßkettenänderungsfenser mit *OK* zu bestätigen. Dieser Bestätigung folgen zwei Aktionen, nämlich Speicherung des neuen Werts und entsprechendes Neuzeichnen des Leitbilds.

5.1.3 Anwendungserfahrungen mit dem systemeigenen Recorder

In *Kapitel 4.1.2* war mit der Programmiersprache des CAD-Systems die Methode *Rohrstützenstoß* programmiert worden. Die gleiche CAD-Methode wurde hier alternativ nochmals quasi ohne Programmierkenntnisse mit dem Recorder entwickelt. Dieses Makro ist nun im CAD-System über eine Symbolleiste aufrufbar. Beim Konstruieren ist die Möglichkeit einer nachträglichen Änderung gegeben. Die Änderung beinhaltet sowohl die Modifizierung der Variablenwerte durch die Schaltfläche [Variablen] als auch die Steuerung der entsprechenden Konstruktionselemente über die Schaltfläche [Blöcke/Aufgaben], siehe *Abbildungen 5-5 und 5-7.*

5.1.3.1 Auswahlbilder

Dem Methodenentwickler stehen leere Rahmen für Auswahlbilder zu der zu entwickelnden Methode zur Verfügung. In diese leeren Rahmen werden Bilder der entwickelten, wählbaren Konstruktionen eingefügt.

Abbildung 5-2 zeigt, dass die Wahl unter verschiedenen Konstruktionsvarianten durch Auswahlbilder für die späteren Anwender sehr erleichtert wird. Die Auswahlbilder sind ohne Anpassung global einsetzbar, d. h. fremdsprachliche Übersetzungen entfallen.

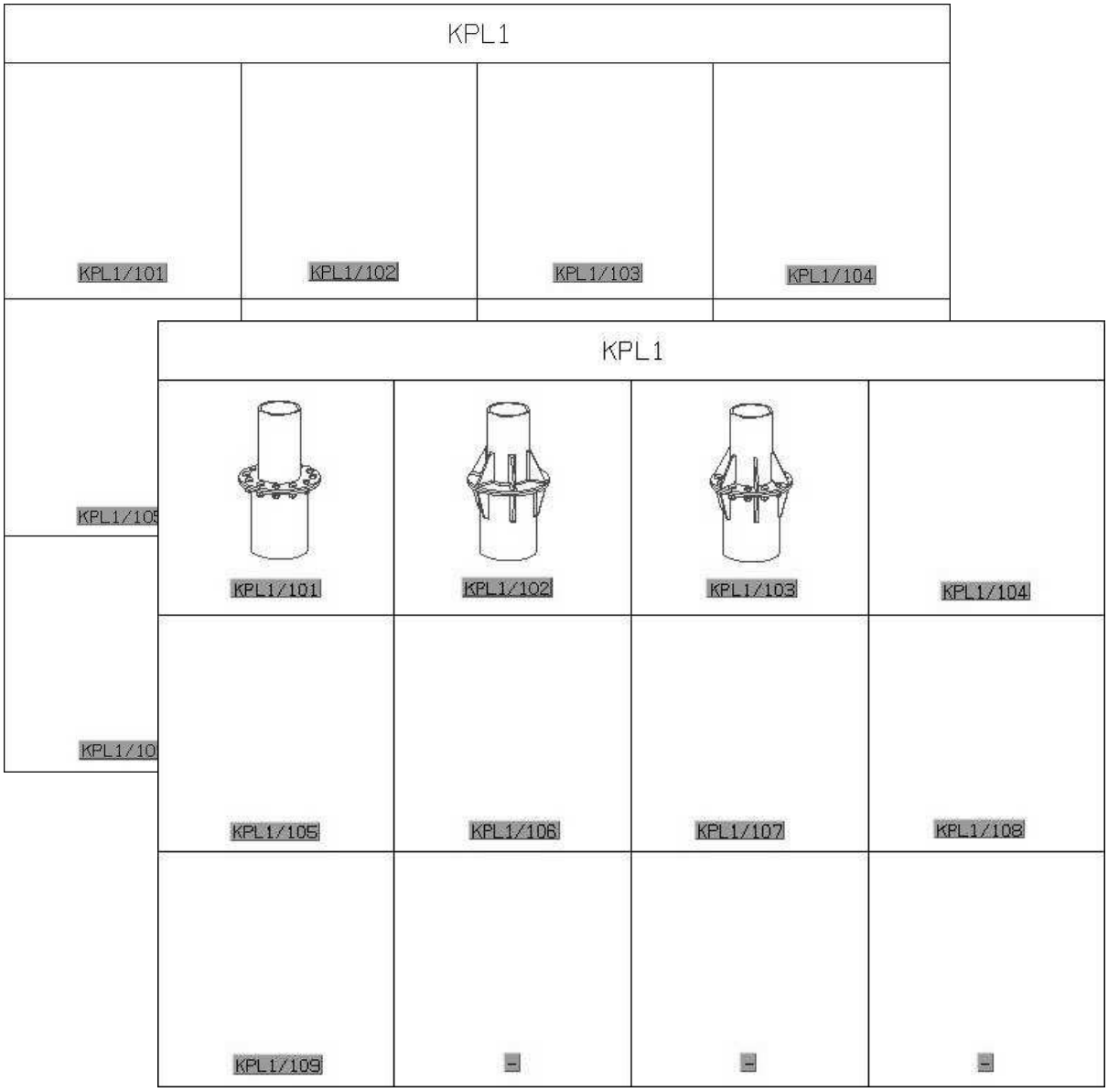


Abbildung 5-2: Auswahlbilder in vorgefertigtem Rahmen

5.1.3.2 Symbolleisten für entwickelte CAD-Methoden

Mit dem in *Abbildung 5-3* dargestellten Dialogfeld kann der Methodenentwickler ein Symbol gestalten, über das der spätere Anwender die entwickelte Methode direkt wählen kann. Die Methode wird über das Dialogfeld in *Abbildung 5-3* dem entsprechenden Symbol zugeordnet.

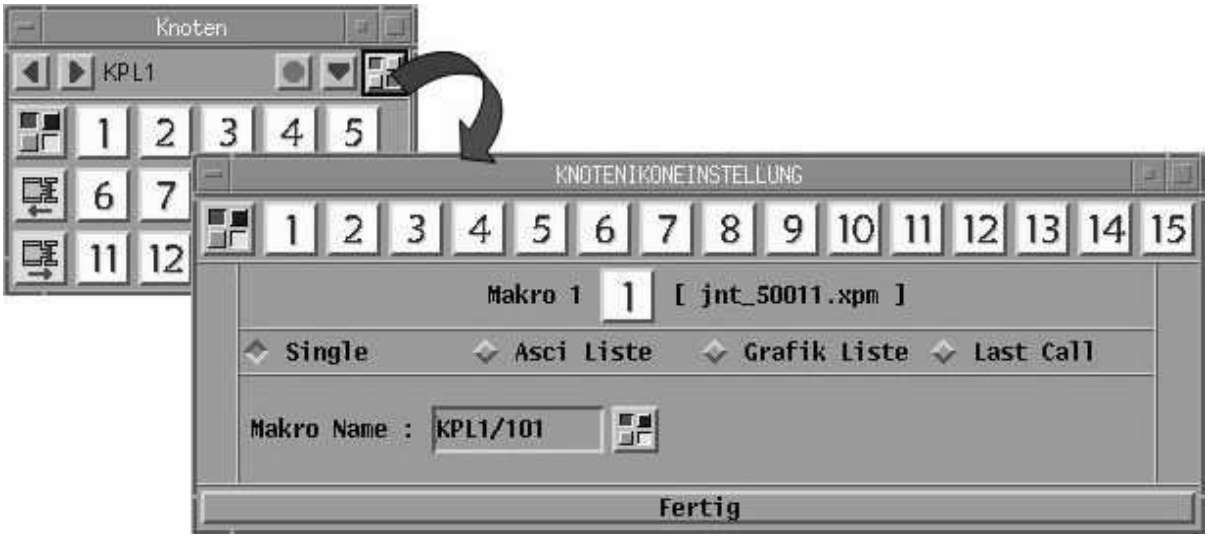


Abbildung 5-3: Erstellung von Symbolen zur direkten Wahl von Methoden

5.1.3.3 Blockdiagramm des systemeigenen Recorders

Abbildung 5-4 zeigt die Struktur des systemeigenen Recorders in Form eines Blockdiagramms, um den generellen Ablauf des Recorders und seine Leistungen zu analysieren.

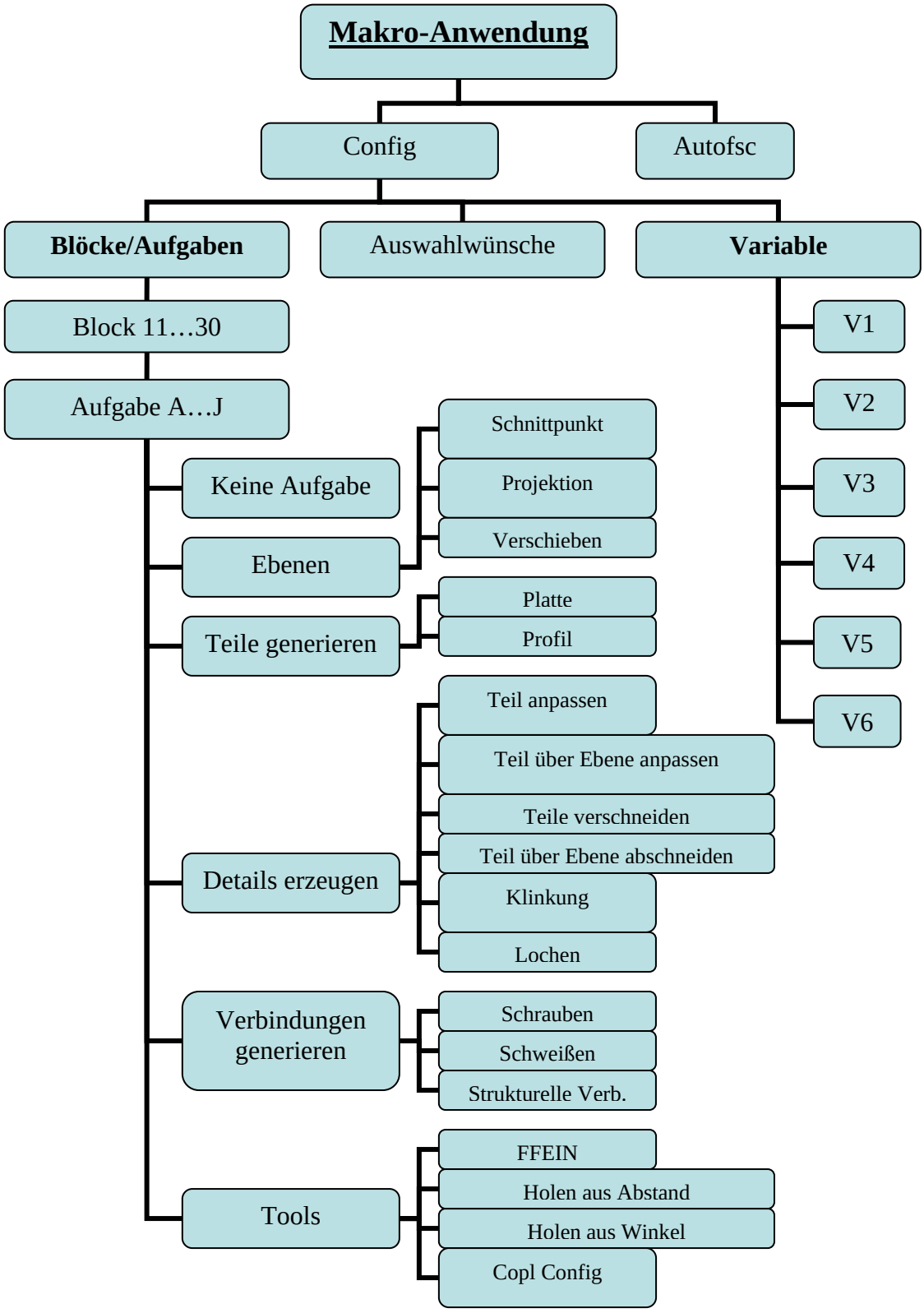


Abbildung 5-4: Blockdiagramm Methodenrecorder

5.1.3.4 Variable

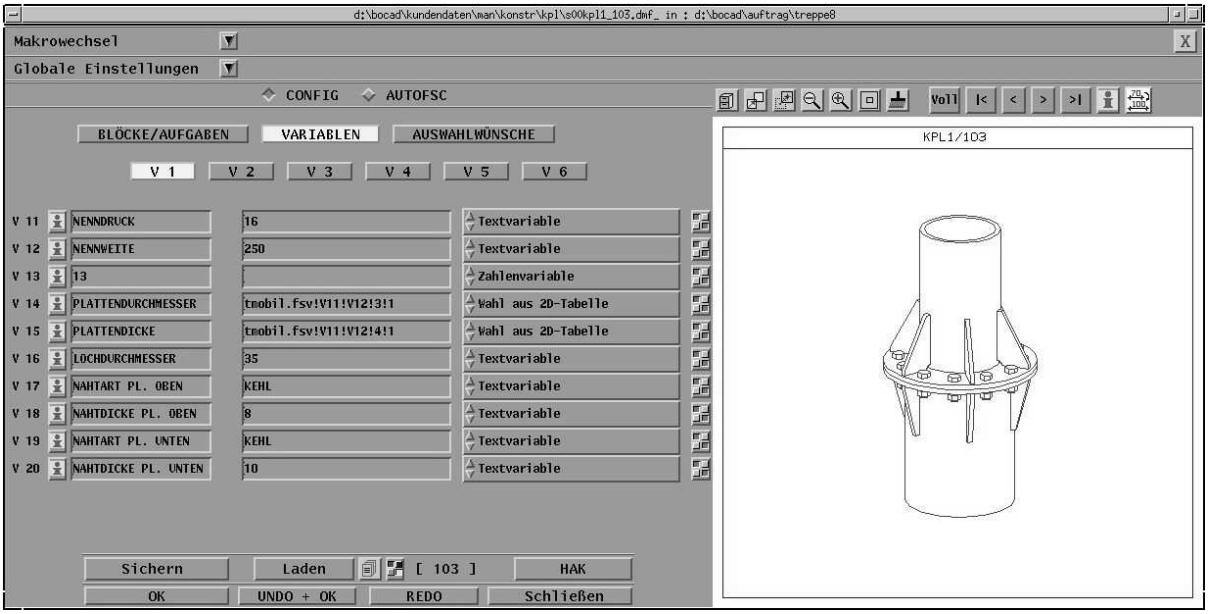


Abbildung 5-5: Eigenschaften von Konstruktionsvariablen

Der Sinn einer CAD-Methode ist es, die Konstruktionslogik einer Lösungsfamilie formal so zu beschreiben, dass der Konstruktionsprozess automatisiert wird. Um innerhalb einer Konstruktionsfamilie eine bestimmte Lösung automatisch konstruieren zu lassen, sind vom Anwender lediglich die Werte der bestimmenden Konstruktionsvariablen anzugeben. Der Methodenentwickler legt die Eigenschaften der Konstruktionsvariablen dazu über die Schaltfläche [Variablen] in *Abbildung 5-5* fest.

Als Konstruktionsvariable stehen verschiedene Typen zur Verfügung:

- Zahlenvariable
- Textvariable
- Wertefeld mit Zahlenvariablen
- Textfeld mit Textvariablen
- Freie Eingabe: z. B. Gleichung
- Wahl aus Tabellen: nächstgrößere oder nächstkleinere Zahl
- Wahl aus linearer Tabelle: Aus einem Eingabewert ergibt sich ein Tabellenwert

- Wahl aus 2D-Tabelle: Aus zwei Eingabewerten ergibt sich ein Tabellenwert
- Abfragebedingung

5.1.3.5 Auswahlwünsche

Mit Schaltfläche [Auswahlwünsche], siehe *Abbildung 5-6*, werden die beim Konstruieren vom Anwender nacheinander anzuklickenden Elemente wie Träger, Ebenen oder Punkte festgelegt.

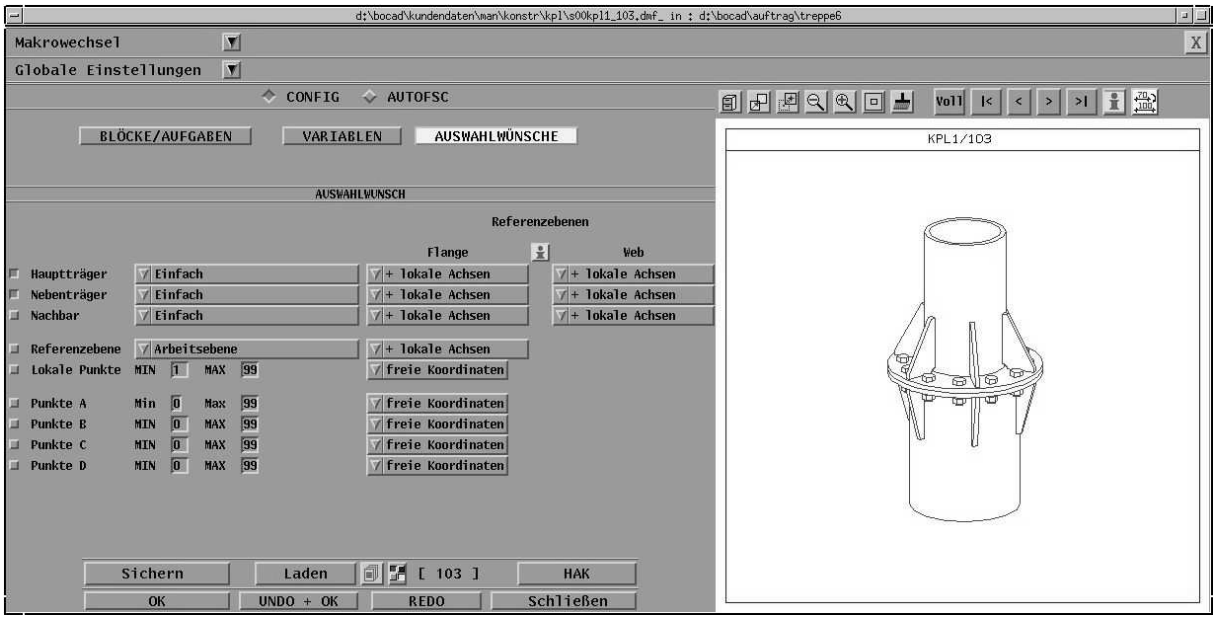


Abbildung 5-6: Dialogfenster mit Schaltfläche [Auswahlwünsche] (aktiviert)

5.1.3.6 Blöcke und Aufgaben

Mit Schaltfläche [Blöcke/Aufgaben], siehe *Abbildung 5-7*, werden die einzelnen Konstruktionsschritte festgelegt, die insgesamt die Konstruktionslogik einer Lösungsfamilie bilden. Der Kern einer CAD-Methode (Blöcke/Aufgaben) besteht also aus der Gesamtheit ihrer Konstruktionsschritte.

Zusammengehörige Konstruktionsschritte werden geblockt. Die Blöcke befinden sich auf der höchsten Ebene der Hierarchie. Die Aufgaben, die die Blöcke bilden, folgen eine Ebene tiefer. Bei diesen Aufgaben handelt es sich um die Folge der einzelnen

Konstruktionsschritte, siehe *Abbildung 5-4.*

Konstruktionsschritte sind in folgende Kategorien eingeordnet:

- Definition der Arbeitsebenen,
- Generierung von Teilen,
- Bearbeitung der Konstruktionselemente (detaillieren),
- Erzeugung von Verbindungen,
- Aufruf von Untermethoden.

Methoden sind in Untermethoden aufteilbar. Der Aufbau wird dadurch übersichtlicher und die Untermethoden können auch in anderen Anwendungen wieder verwendet werden.

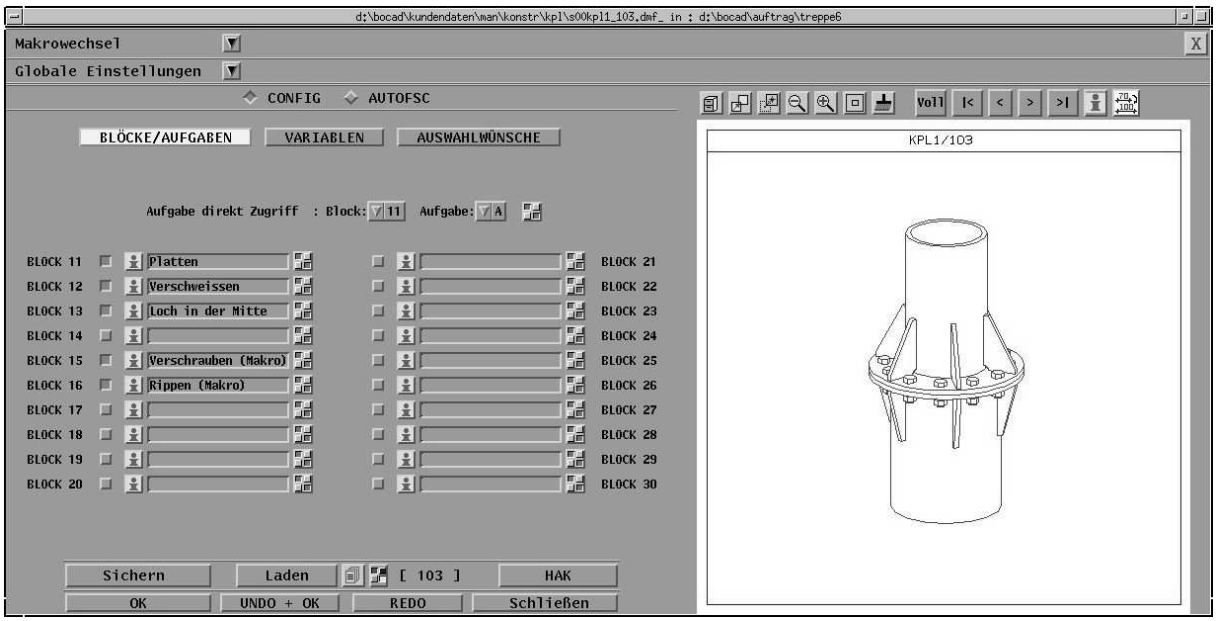


Abbildung 5-7: Dialogfenster mit Schaltfläche [Blöcke/Aufgaben] (aktiviert)

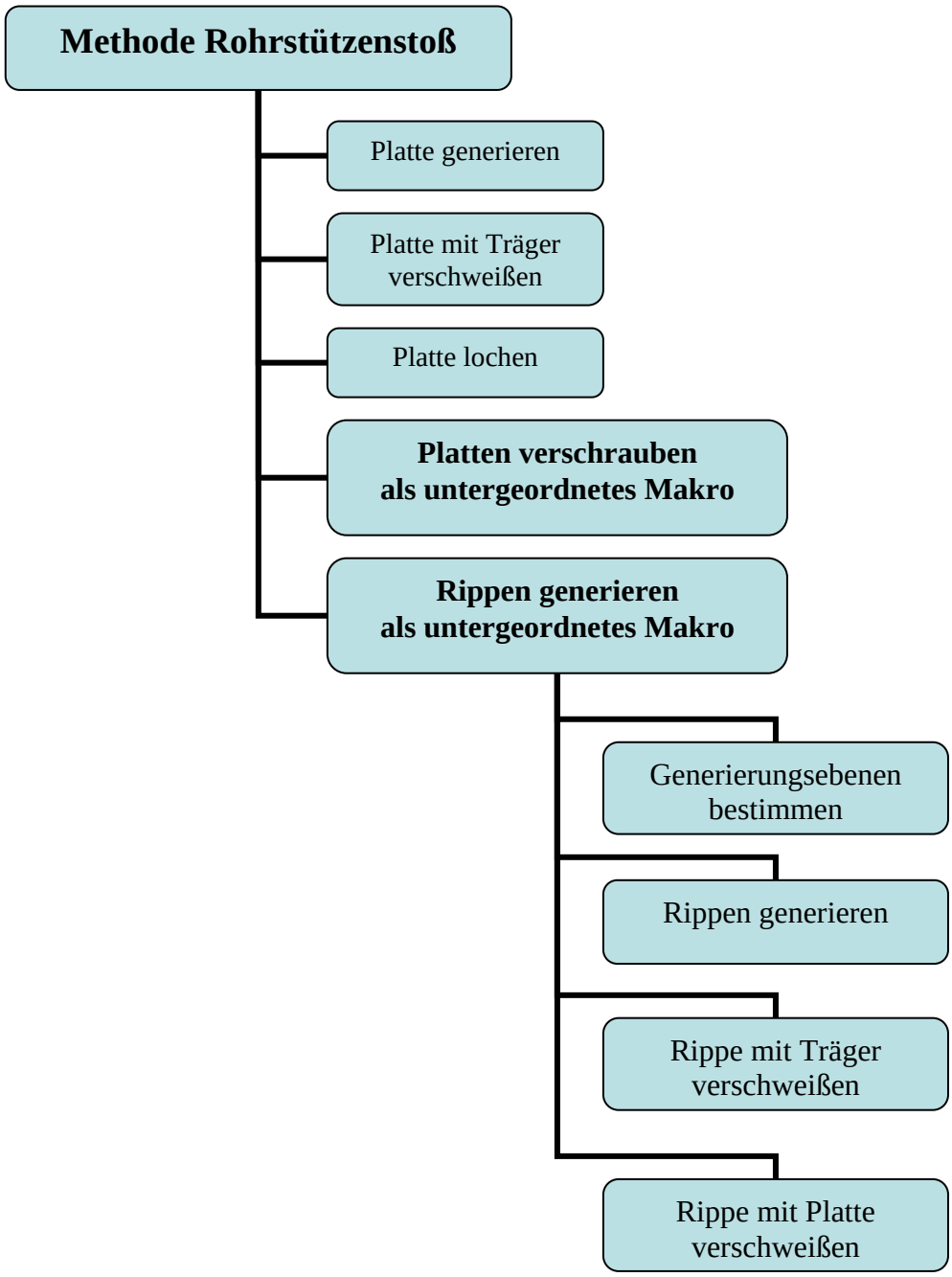


Abbildung 5-8: Recorder-Befehlsfolge für Methode *Rohrstützenstoß*

5.2 Logikrekorder als Weiterentwicklung des Methodenrecorders

Der bisher beschriebene Recorder wurde von Anwendern für so kompliziert befunden, dass er in der Baupraxis vor Ort nicht eingesetzt wurde. Die Folge der notwendigen Arbeitsschritte ist wegen der unübersichtlichen, ineinander geschachtelten Dialogfenster entsprechend *Abbildung 5-9* nicht erkennbar. Die Variablen, die Blöcke, die Aufgaben usw. sind unanschaulich kodiert, (durch diese Kodierung wird die Referenz in der Methode gelöst, z. B. V14, Block 22, Aufgabe J), der systematische Ablauf des Konstruierens geht aus den Augen verloren, weil die Arbeitsvorgänge in beliebiger Reihenfolge möglich sind.

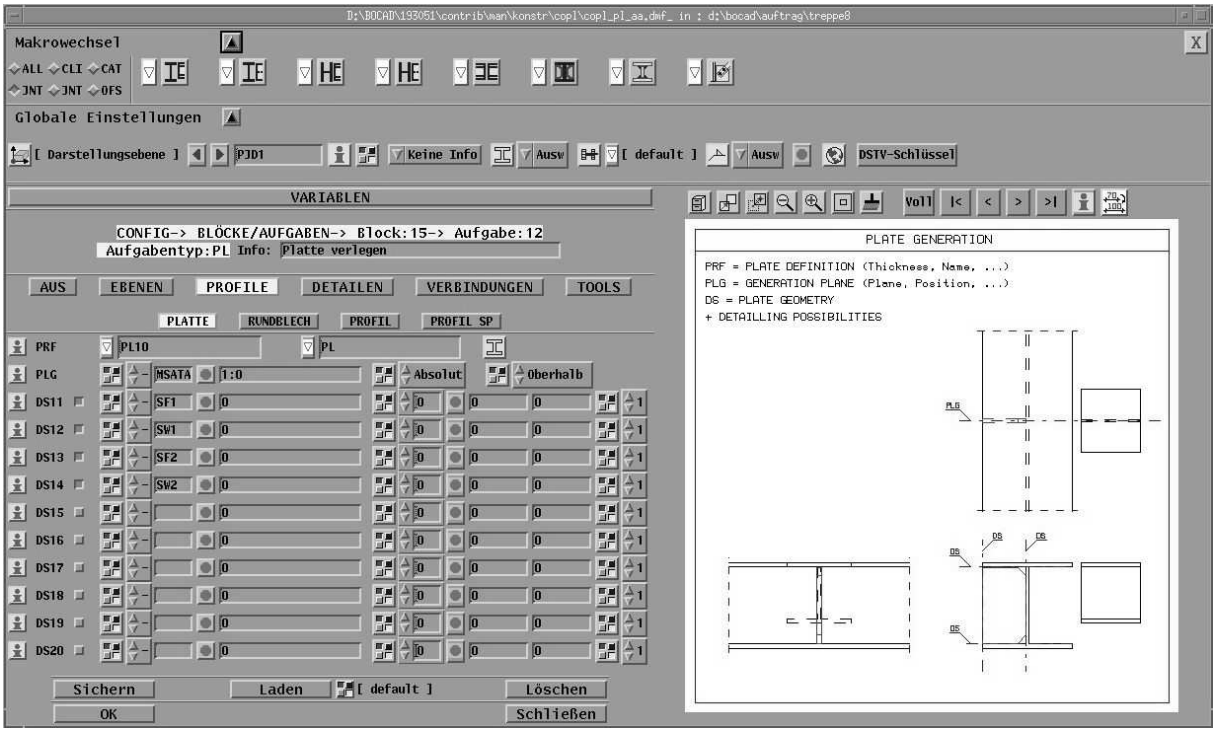


Abbildung 5-9: Kompliziertes Dialogfenster der Plattenerzeugung

Eine Lösung, die in der Baupraxis keine Akzeptanz findet, ist ingenieurwissenschaftlich nicht ausreichend und deshalb zu verwerfen.

In dieser Arbeit wird daher ein neuer Ansatz konzipiert und ingenieurwissenschaftlich im Experiment nachgewiesen. Dieser neue Ansatz wird Logikrekorder genannt, da er

folgerichtig die Konstruktionslogik aufzeichnet.

Die gewünschten Konstruktionen werden mit einem Assistenten für Konstruktionslogik, der in den Logikrekorder eingebettet ist, in systematisch aufgebauten Schrittfolgen aufgenommen.

Der Logikrekorder wird mit Hilfe von einigen ausgewählten Beispielen aus dem Treppenbereich vorgestellt.

5.2.1 Nutzoberflächengestaltung des Logikrekorders und Ablauf der Aktionen

Der Logikrekorder wird im CAD-System wie ein eigenständiger Konstruktionsauftrag unter einem Namen, z. B. *Tre3_112* (Konstruktionsfamilie TRE3 für Kundentreppen und Methodennummer 112 in der Lösungsfamilie) in *Abbildung 5-10*, gestartet.

Die Nutzoberflächenelemente des Logikrekorders sind das Bild, das eine durch Methoden erzeugte grafische Meta-Datei darstellt, und die Symbole für die Benutzeraktionen, siehe *Abbildung 5-10.*

Abbildung 5-10: Nutzoberfläche des Logikrekorders


Nach dem Klicken des praxisüblichen Symbols *Aufzeichnen* beginnt analog zu den vom Betriebsystem Windows allgemein bekannten Assistenten schrittweise die Aufzeichnung des Konstruktionsablaufs. In den ersten drei Schritten wird der Arbeitsvorgangstyp bestimmt, siehe am Beispiel *Abbildung 5-11*, und dann werden die Eigenschaften des Arbeitsvorgangs, wie in *Abbildung 5-12*, festgelegt. Die so aufgenommenen Arbeitsvorgänge sind nachträglich mit dem Klicken auf das Symbol *Aufzeichnen* ergänzbar.

Abbildung 5-11: Festlegen des Arbeitsvorgangstyps *Geländer an die Platte anpassen*

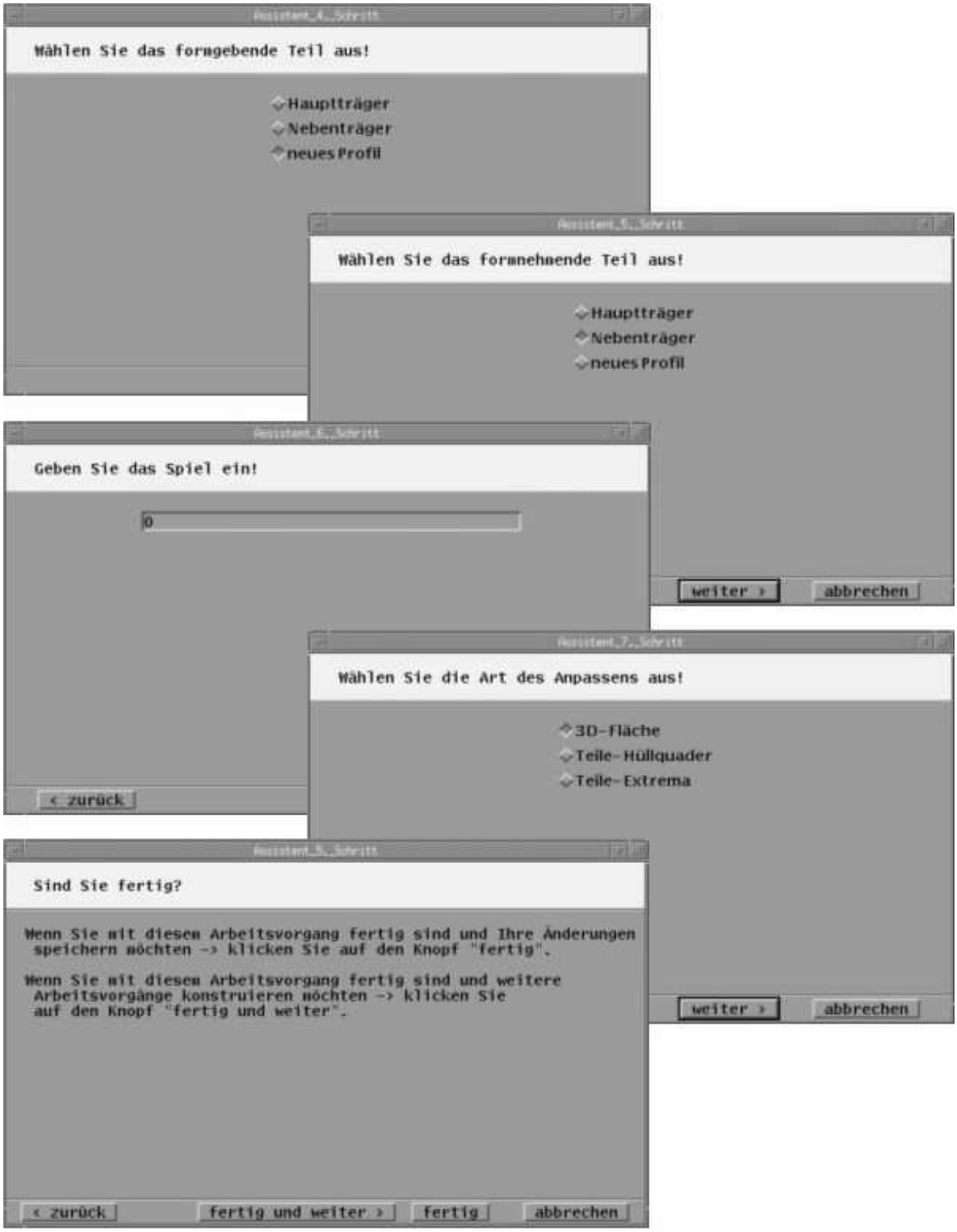


Abbildung 5-12: Aufzeichnen des Arbeitsvorgangs *Geländer an die Platte anpassen*

Die in *Abbildung 5-4* dargestellten Aufgaben (hier Arbeitsvorgänge) werden in den Logikrekorder übernommen. Die wichtigsten, wie z. B.:

- Teile generieren → Platte,
- Details erzeugen → Teil anpassen, siehe *Abbildungen 5-11 und 5-12*,
- Verbindung generieren → Verschweißen,
- Verbindung generieren → Verschrauben,

sind im Logikrekorder verwirklicht. Weitere Arbeitsvorgänge sind in den Logikrekorder noch zu übertragen. Der Logikrekorder ist wegen der logischen Struktur der Assistentenfenster mit weiteren Arbeitsvorgängen einfach erweiterbar, siehe *Kapitel 5.2.2*.

Die folgenden Aufzeichnungsablaufdiagramme (*Bild 5-13, 5-14, 5-15* und *5-16*) stellen die miteinander durch Assistenten verketteten Schritte der Arbeitsvorgänge dar.

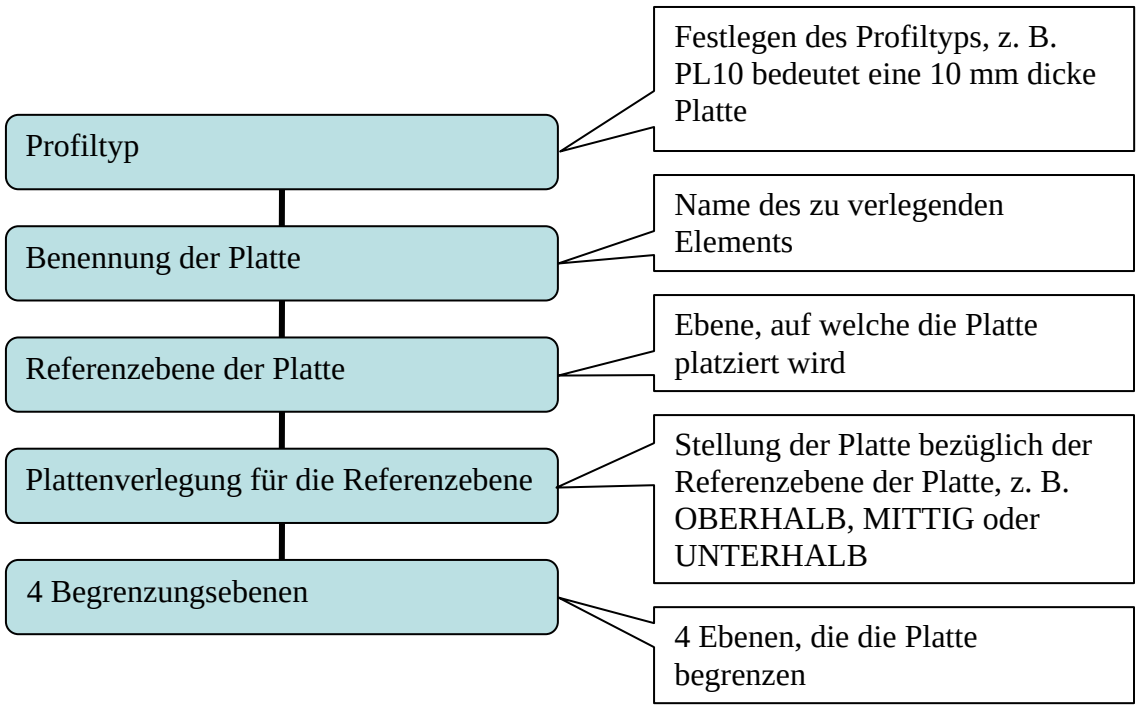


Abbildung 5-13: Arbeitsvorgang *Platte verlegen*

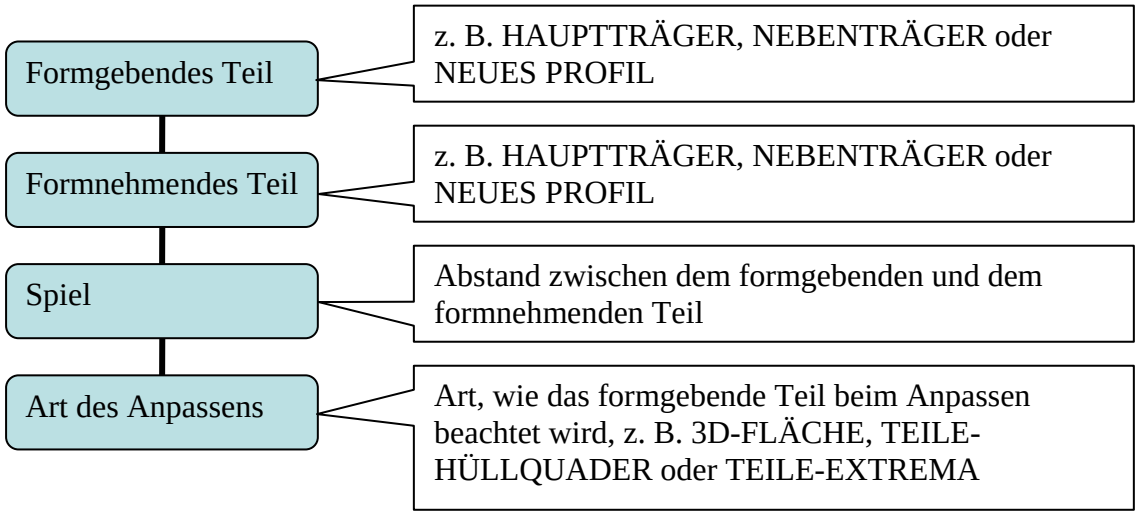


Abbildung 5-14: Arbeitsvorgang *Teil an ein Teil anpassen*

Das Aufzeichnen des Arbeitsvorgangs *Teil an ein Teil anpassen* wurde konkret in *Abbildung 5-12* dargestellt.

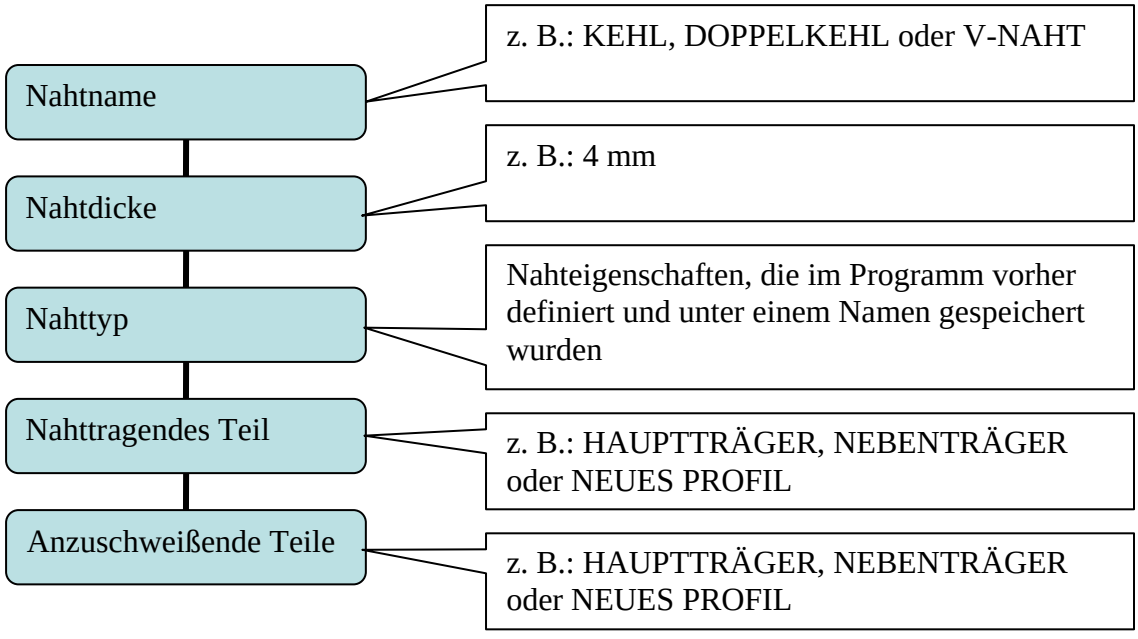


Abbildung 5-15: Arbeitsvorgang *Verschweißen von Teilen*

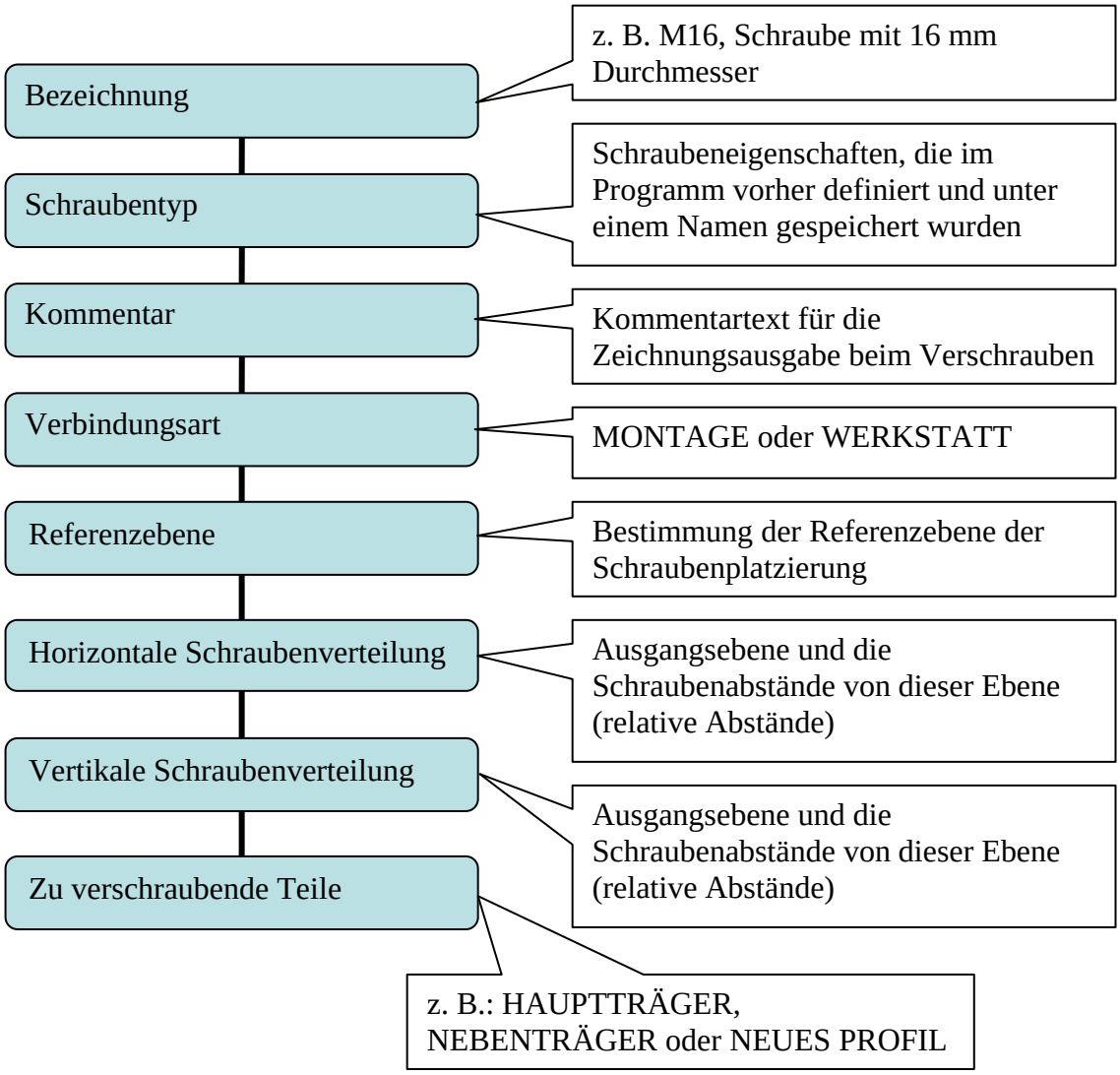


Abbildung 5-16: Arbeitsvorgang *Verschrauben von Teilen*

Die Ebenen haben zentrale Funktion in der Methodenerstellung. Ein Profil oder eine Platte wird grundsätzlich als Stab zwischen 2 Punkten platziert, siehe *Kapitel 2.11*. Auch die Schraubenplatzierung wird über Punkte bestimmt. Diese Punkte werden hier nicht durch die konventionellen 3 Koordinaten (x, y, z), sondern durch einen Schnitt von drei zueinander nicht parallel liegenden Ebenen festgelegt. Die drei Ebenen sind im oben dargestellten Arbeitsvorgang *Verschrauben von Teilen* gut erkennbar. Die Referenzebene, die Ebenen zur horizontalen Schraubenverteilung und die

Ebenen zur vertikalen Schraubenverteilung überschneiden sich in den gewünschten Punkten.

Das folgende Ablaufdiagramm (*Abbildung 5-17*) zeigt die notwendigen Eigenschaften, mit denen eine Ebene festgelegt wird.

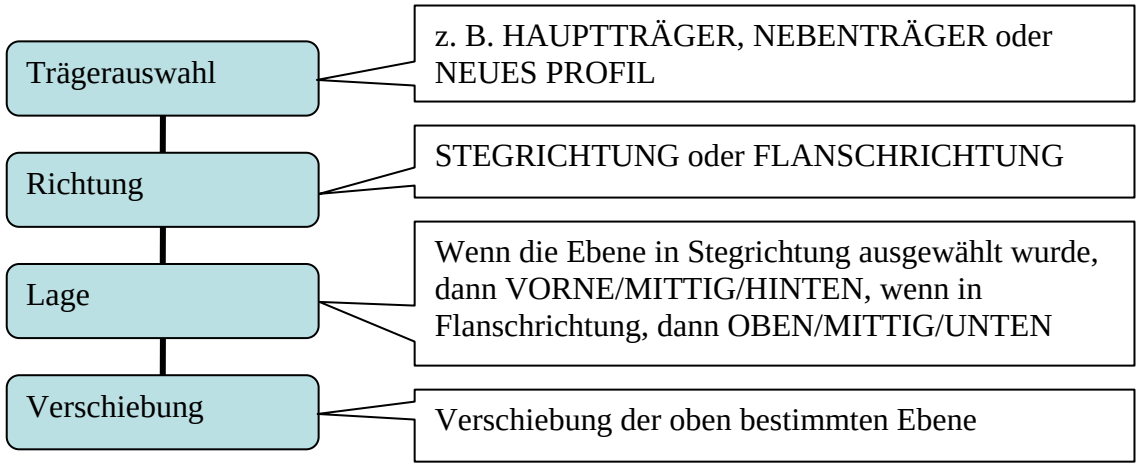


Abbildung 5-17: Festlegen einer Ebene

Die Ebenen haben Beziehungen zu den Elementen des Anschlusses, siehe Ablaufdiagramm *Festlegen einer Ebene*. Mit diesen Beziehungen werden die Punkte in den Anschlüssen koordinatenunabhängig erzeugt.


Nachdem die gewünschte CAD-Methode aufgezeichnet wird, kommt es oft vor, dass die Methode mit anderen Einstellungen gebraucht wird. Das Symbol *Bearbeiten* öffnet ein Fenster, in dem alle aufgenommenen Arbeitsvorgänge aufgelistet sind, siehe *Abbildung 5-18.*

Abbildung 5-18: Bearbeitung der Arbeitsvorgänge

Das Optionsfeld, das vor dem Namen des Arbeitsvorgangs zu finden ist, reguliert, ob der entsprechende Arbeitsvorgang beim Methodenablauf beachtet wird. Das hinter dem Arbeitsvorgangsnamen des zu bearbeitenden Arbeitsvorgangs stehende (in der *Abbildung 5-18* markierte) Bearbeitungssymbol ruft den Assistenten für die nachträgliche Bearbeitung auf, siehe *Abbildung 5-19*.

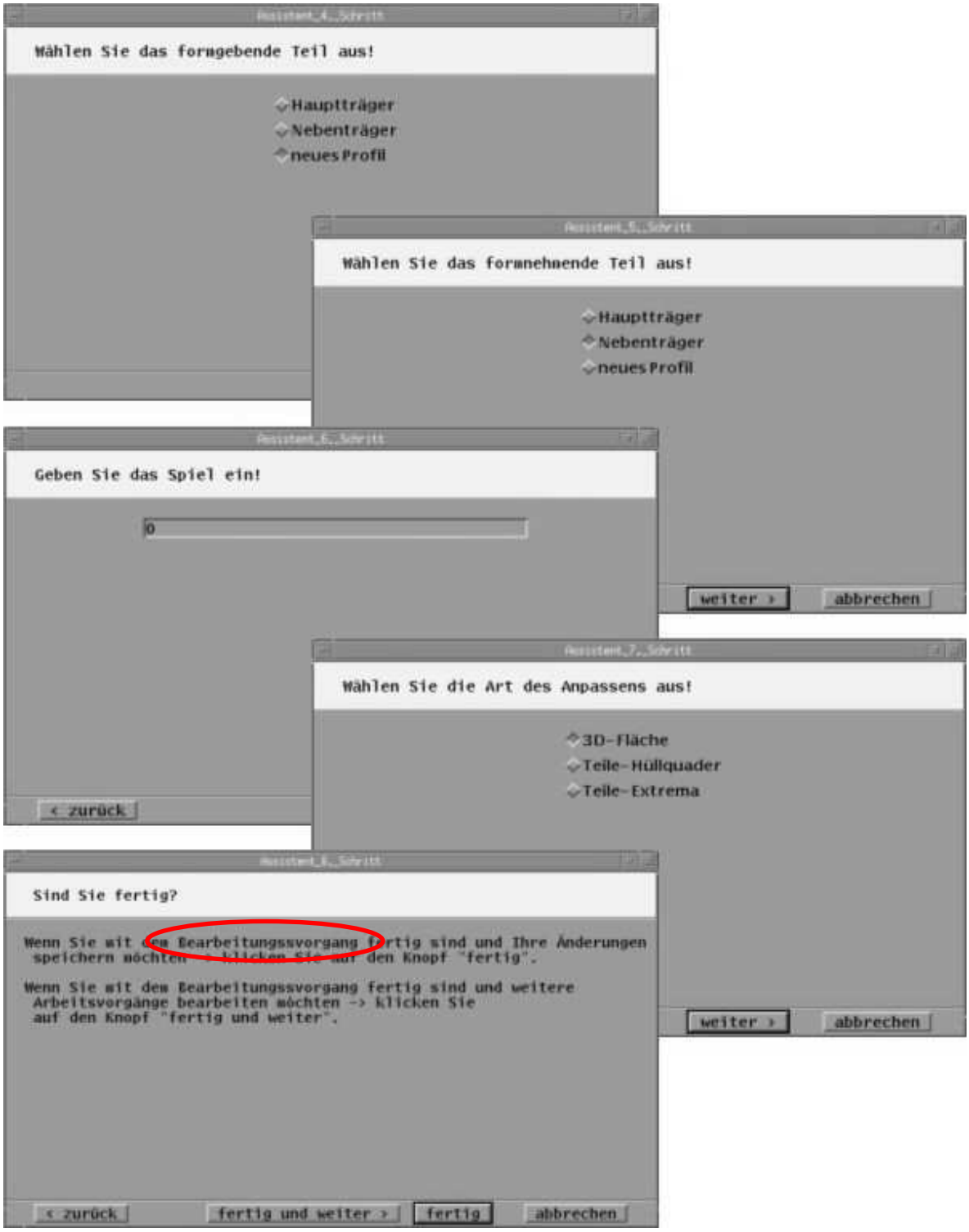


Abbildung 5-19: Arbeitsvorgang „*Geländer an die Platte anpassen*“

 Nach dem Aufzeichnen oder der Bearbeitung wird über das hierfür übliche Symbol die Ausführung der Methode ermöglicht. Nach der Auswahl des Ausführungssymbols im Logikrekorder sind die bereits erzeugten Einzelteile, z. B. Haupt- und Nebenträger pickbar. Der Anschluss zwischen Einzelteilen, dessen Methode aufgezeichnet und zur Lösungsfamilie erweitert wurde, wird dann automatisch ausgeführt.

Für die einfache Erkennung des angefertigten Makros bei der späteren Wiederverwendung sollte ein dreidimensionales Bild des Anschlusses im Logikrekorder-Fenster erzeugt werden.

Zunächst wird eine durch Methoden erzeugte grafische Meta-Datei eingelesen, die aus Texten und Linien besteht. Die Erstellung einer solchen Datei wurde bereits im *Kapitel 3.6* erläutert. Nach dem Aufzeichnen und Ausführen der Methode kann ein dreidimensionales Bild des Anschlusses erzeugt werden. Dann wird - nach der in eigener Landessprache ausgewählten Leitmeldung - diese Meta-Datei mit einem dreidimensionalen Bild des Anschlusses überschrieben und unter dem gleichen Namen im Auftrags-Verzeichnis gespeichert.  Das Klicken auf das Symbol *Bild auffrischen* ersetzt die neue Grafikdatei im Logikrekorder-Fenster, siehe *Abbildung 5-20*.

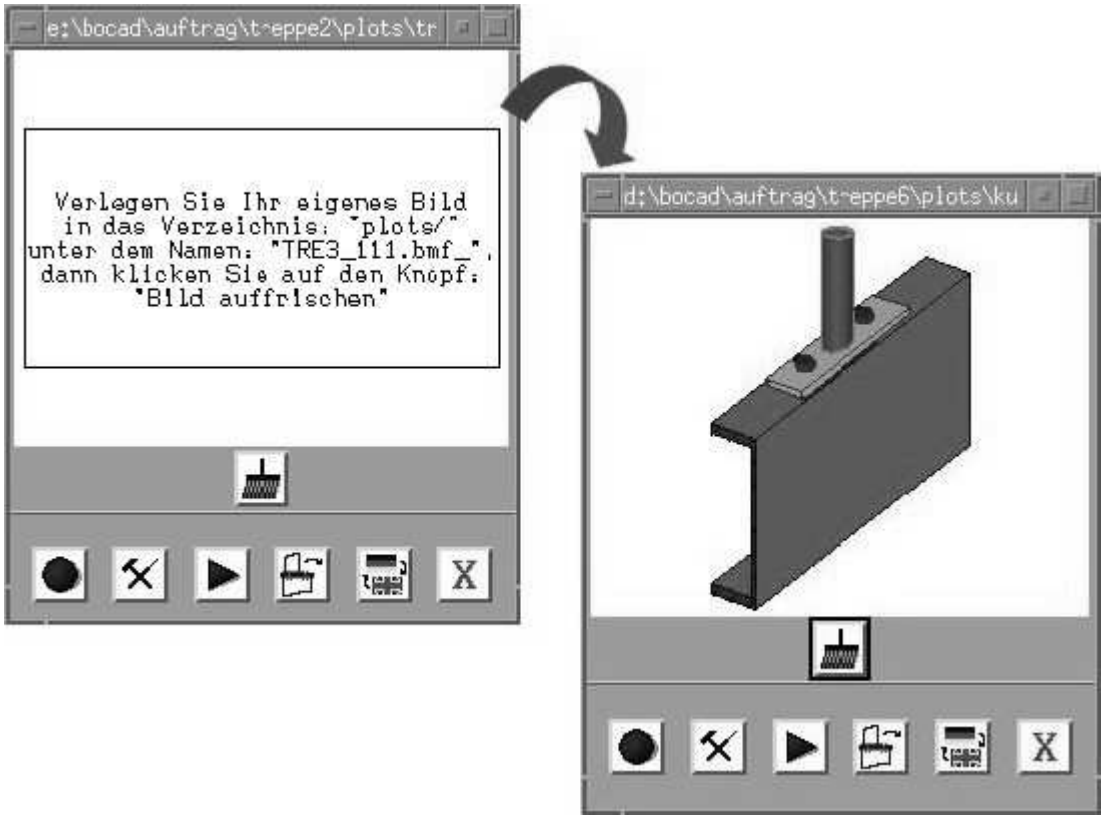


Abbildung 5-20: Ersatz der Leitmeldung durch Konstruktionsergebnis als Erkennungsbild eines Anschlusstyps

 Während des Konstruierens werden mehrere Konstruktionsmethoden aufgenommen und ausgeführt. Wenn eine bereits in der Konstruktion ausgeführte Methode wieder verwendet werden soll, muss der Name der Methode nicht mehr bekannt sein, um die entsprechende Methode aufzurufen, weil sie von der Konstruktion geklont werden kann.

Diese Funktion wird über das Symbol *Klonen* aktiviert, wonach der Logikrekorder geschlossen wird. Nach der Wahl des in der Konstruktion mit der zu klonenden Methode angefertigten Konstruktionsobjektes wird der Logikrekorder der gewünschten CAD-Methode geöffnet. *Abbildung 5-21* zeigt das Klonen einer Methode. Dabei wird die Konstruktionsmethode, die einen Anschluss zwischen Geländer und Wange erzeugt, geschlossen. Nach dem Klonen wird die gewünschte CAD-Methode eines

Wange-Stufe-Anschlusses im Logikrekorder geöffnet.

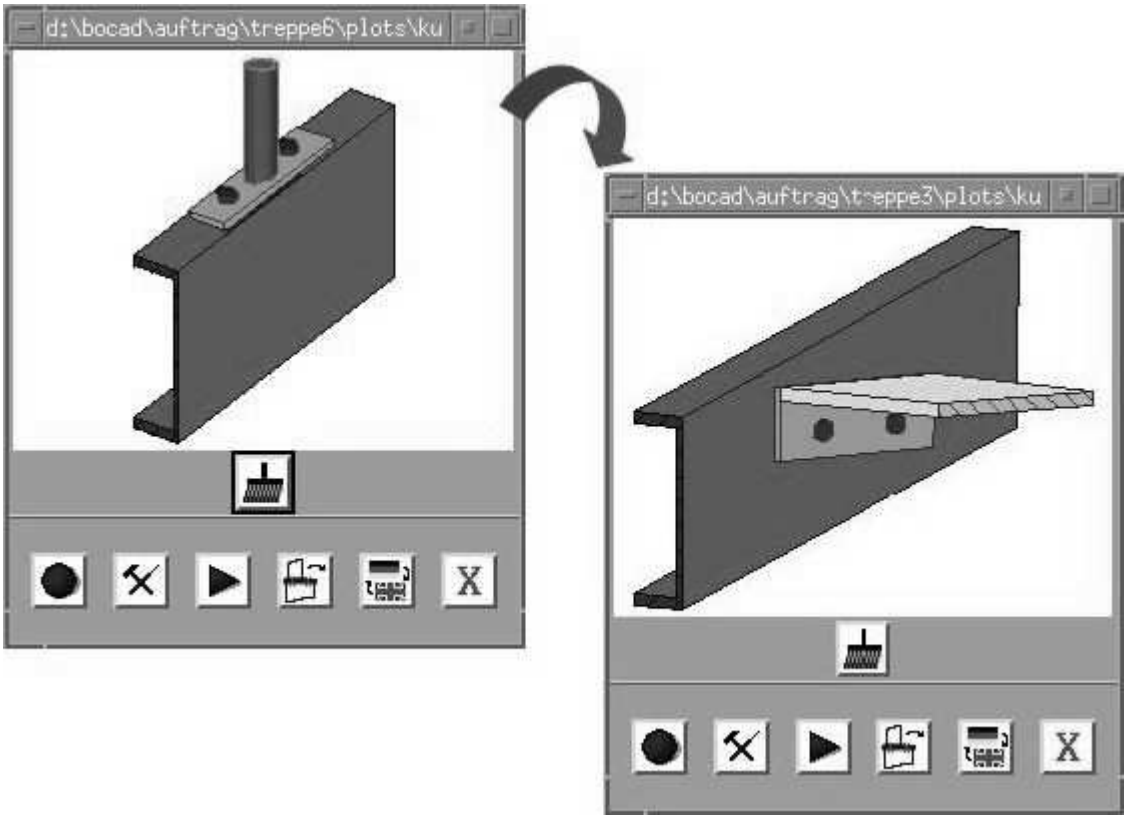


Abbildung 5-21: Klonen eines Bauteils nach Anschluss

 Heutzutage ist die Frage der Mehrsprachigkeit wegen der Globalisierung und der Ausbreitung der Märkte ein wichtiger Gesichtspunkt, an den bei der Entwicklung unbedingt gedacht werden muss. Der Logikrekorder ist prinzipiell länderunabhängig, denn Textförmige Leitmeldungen - sowohl in Dialogboxen als auch in Metadateien - können nach *Abbildung 5-22* entsprechend der gewünschten Landessprache gewählt werden.

Abbildung 5-22: Sprachenwechsel-Fenster

Die Textvariablen der Dialogboxen werden in verschiedenen Konfigurationsdateien nach Sprachen definiert. Somit hat jede Sprache (3 Sprachen exemplarisch entwickelt) eine zugehörige Konfigurationsdatei, die nach der Sprachauswahl aktiviert wird.

Quelltext-Ausschnitt der Konfigurationsdatei *labels_deu.cfg*:
------1000 #def var22 AUS #def var23 AUS #def var121 ERSTELLEN #def var122 FILTER ZEIGEN #def var123 FREIE EINGABE: #def var124 FREIER FFEIN-AUFRUF
Quelltext-Ausschnitt der Konfigurationsdatei *labels_eng.cfg*:
------1000 #def var22 OFF #def var23 OUT #def var121 CREATE #def var122 SHOW FILTER #def var123 FREE INPUT: #def var124 CALL FREE FFEIN

Eine andere Art der Übersetzung ist es, die Texte und ihre Übersetzungen in den Definitions-Dateien zu sammeln. In diesen Dateien werden die Textelemente jeweils nach ihrer Sprache aufgeteilt, und die Wörter werden mit dem Zeichnen *§* verbunden. Beim Programmstart werden die Textwerte aus der entsprechenden Spalte der gewählten Sprache ausgelesen.

Quellcode-Ausschnitt von einer Definitions-Datei (labels.dat):

```
AUS§OFF§
ERSTELLEN§CREATE§
FILTER ZEIGEN§SHOW FILTER§
FREIE EINGABE:§FREE INPUT:§
FREIER FFEIN-AUFRUF§CALL FREE FFEIN§
Fußzeile§Footer§
FUßZEILE§FOOTER§
```

Im ersten - schließlich beim Konstruktionsrekorder angewandten - Fall ist die Übersetzungsdatei zwar größer, was jedoch in mehrerer Hinsicht Vorteile hat:

- Aktive Sprachumschaltung im laufenden Programm,
- Vermeiden von Missverständnisse wegen wörtlicher statt sinngemäßer Übersetzung.

Der o. g. zweite Vorteil bedarf der Klärung. Die Verwendung der Definitions-Dateien hat die Übersetzung vereinheitlicht, damit dieselbe Übersetzung an allen Stellen des Programms Gültigkeit hat. Die Praxis zeigt aber, dass diese Vereinheitlichung bei komplexen Programmen nicht immer korrekt ist, weil die Übersetzung Wort für Wort in vielen Fällen Missverständnisse verursacht, z. B.: *AUS* kann sowohl *OFF* als auch *OUT* bedeuten. Die Bedeutung eines Wortes ist von der Anwendung des Wortes im Programm abhängig, deswegen können gleiche Wörter in verschiedenen Dialogfenstern unterschiedliche Übersetzungen bedingen.

5.2.2 Assistent für Konstruktionslogik

Wenn Programme zum ersten Mal auf dem Computer installiert werden, sind zahlreiche Assistenten des Setup-Programms dabei behilflich. Solche Assistenten werden in dieser Arbeit auch im CAD-Bereich eingesetzt.

Der Modul des Assistenten für Konstruktionslogik besteht aus vielen Fenstern, die - wegen der nachträglichen Bearbeitung, Ergänzung und Korrektur - systematisch aufgebaut sein sollen.

Die Komponenten eines Assistentenfensters sind in einer Konfigurationsdatei entsprechend *Abbildung 5-23* programmiertechnisch strukturiert.

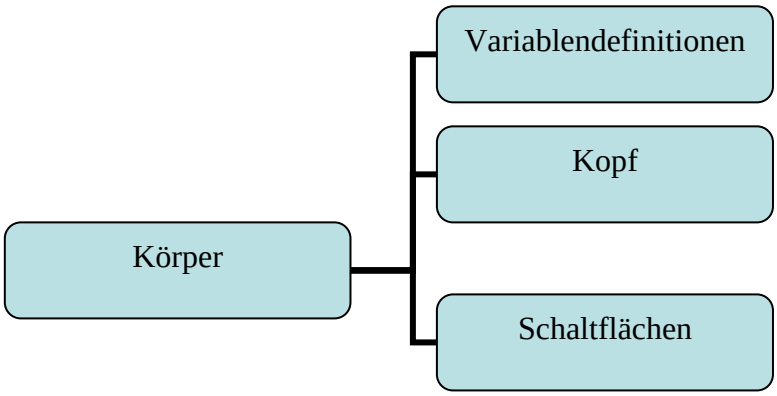


Abbildung 5- 23: Aufbau eines Assistenten-Fensters

Die Boxnummern der Arbeitsschritte werden nach *Abbildung 5-24* systematisch strukturiert.

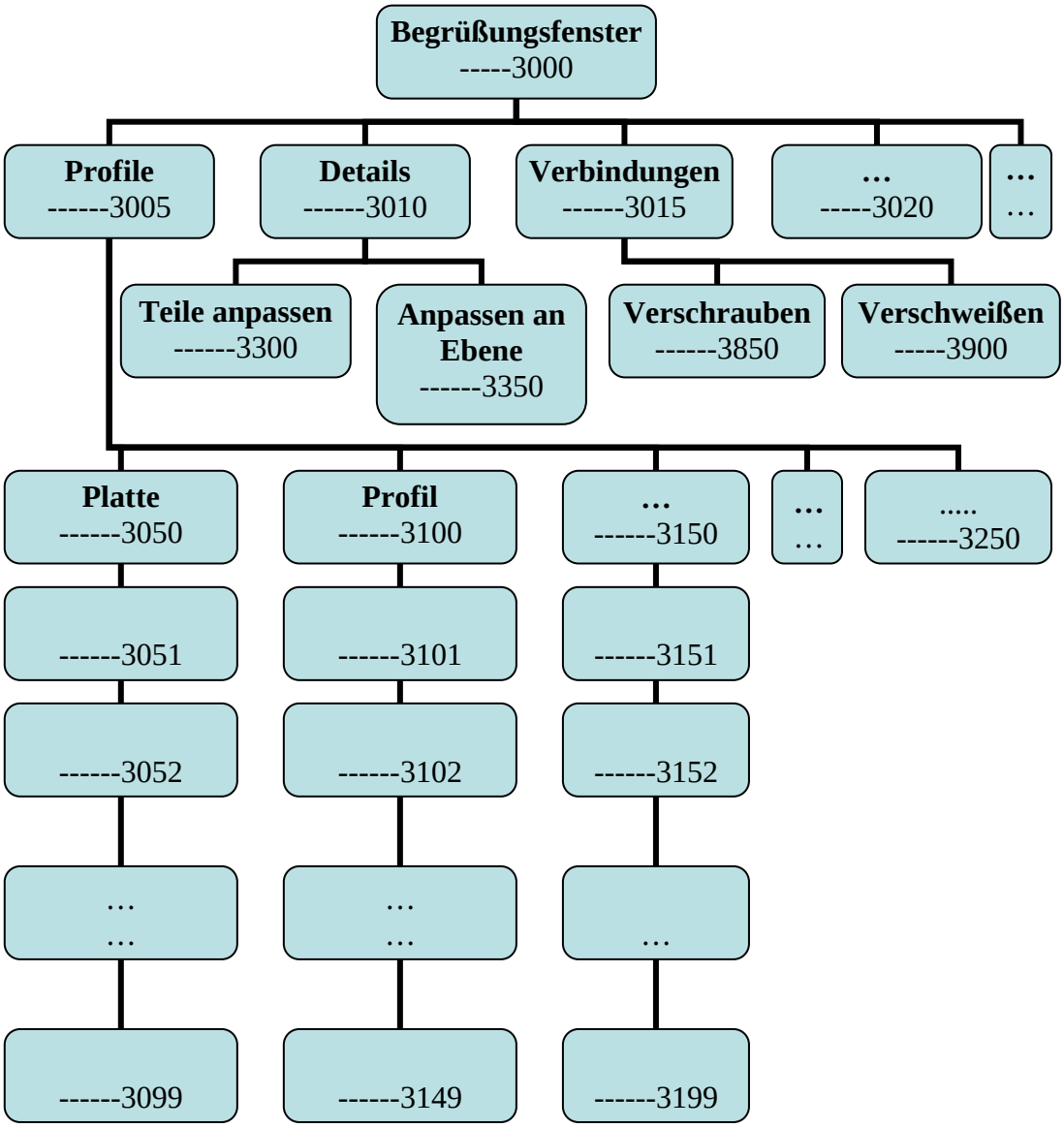


Abbildung 5-24: Strukturierung des Fensterkörpers

Der Konfigurationsdateibereich von 3000 bis 4000 bildet den Körper des Assistenten. In einer Box dieses Bereiches sind weitere Boxaufrufe mit dem Befehl *#in* eingebettet.

Im folgenden Beispiel wird ein Dialogfenster des Assistenten erklärt, in dem der Name des Arbeitsvorgangs *Platte verlegen* vom Konstrukteur bestimmt wird. Siehe auch *Abbildung 5-25*.

Abbildung 5-25: Assistent an der Nutzoberfläche

Im Quelltext:
------3050 #in 30 21162 **1050** #in 30 21162 **2000** /* TBL (1300,1001,10, 1, 1),(1,options,-),(1,0,0),(6) LAB (1301,1300,1,1,1),(1, ,-) LAB (1302,1300,2,1,1,99,1),(1, ,-),(0,0, 0,0,0,250) **EDI** (1323, 1300, 3, 3, 1), (9, TSK11, -), (#rscnr, 21162, assi, stPL, default , 20, 40), (#rscnr, 21162, assi, stPL, default , 20, 40), (0, 0, 40) /* #in 30 21162 **4001** /*

Zuerst werden in der Box *1050* die Variablen definiert, dann wird der Assistentenkopf unter der Box 2000 eingelesen. Mit dem Eingabefeld (EDI, siehe *Kapitel Eingabefelder 3.2.4*) ist der Körper des Dialogfensters an der Reihe, und letztendlich wird die Box 4001 vom Schaltflächenbereich aufgerufen, siehe *Abbildung 5-23,* und weitere Detaillierung im *Anhang 5.2.2*.

5.2.3 Experimenteller Nachweis an firmenspezifischen Bauarten von Treppen

Treppen sind Bauelemente, die sich grundsätzlich für automatisches Konstruieren eignen. Die Treppendetails sind in der Praxis jedoch so vielfältig, dass firmenübergreifend vorgefertigte Methoden die firmenspezifischen Anforderungen nicht erfüllen.

Der Markt drängt die Softwarefirmen zur Einführung von selbstgestaltbaren CAD-Methoden. Aus diesen Gründen sind am Beispiel von Treppen einerseits die Leistungsfähigkeit, andererseits die ingenieurwissenschaftlich relevante Praxisbrauchbarkeit des neu konzipierten Logikrekorders nachgewiesen.

Nach [14] wurde erläutert: „Im Stahl- und Anlagebau sind häufig geschweißte und geschraubte Anschlüsse zu konstruieren, um zwei Träger miteinander zu verbinden." Genauso entstehen die Treppenanschlüsse mit Geländerhandlauf und -pfosten, Geländerpfosten und Knieleisten, Geländerpfosten und Wange, Wange und Stufe immer zwischen zwei Stäben.

Diese individuell gestalteten, sich bei einer Treppe wiederholenden Knoten sind eine so überschaubare Konstruktionsaufgabe, dass die eingegrenzte Anzahl der Arbeitsvorgänge des Logikrekorders nicht überschritten wird.

Das folgende Beispiel (*Abbildung 5-26*) zeigt einen Ausschnitt im Treppenbereich einer komplexen Konstruktion, deren Anschlüsse mit dem Logikrekorder leistungsfähiger und effektiver als mit herkömmlichen Methoden angefertigt werden können.

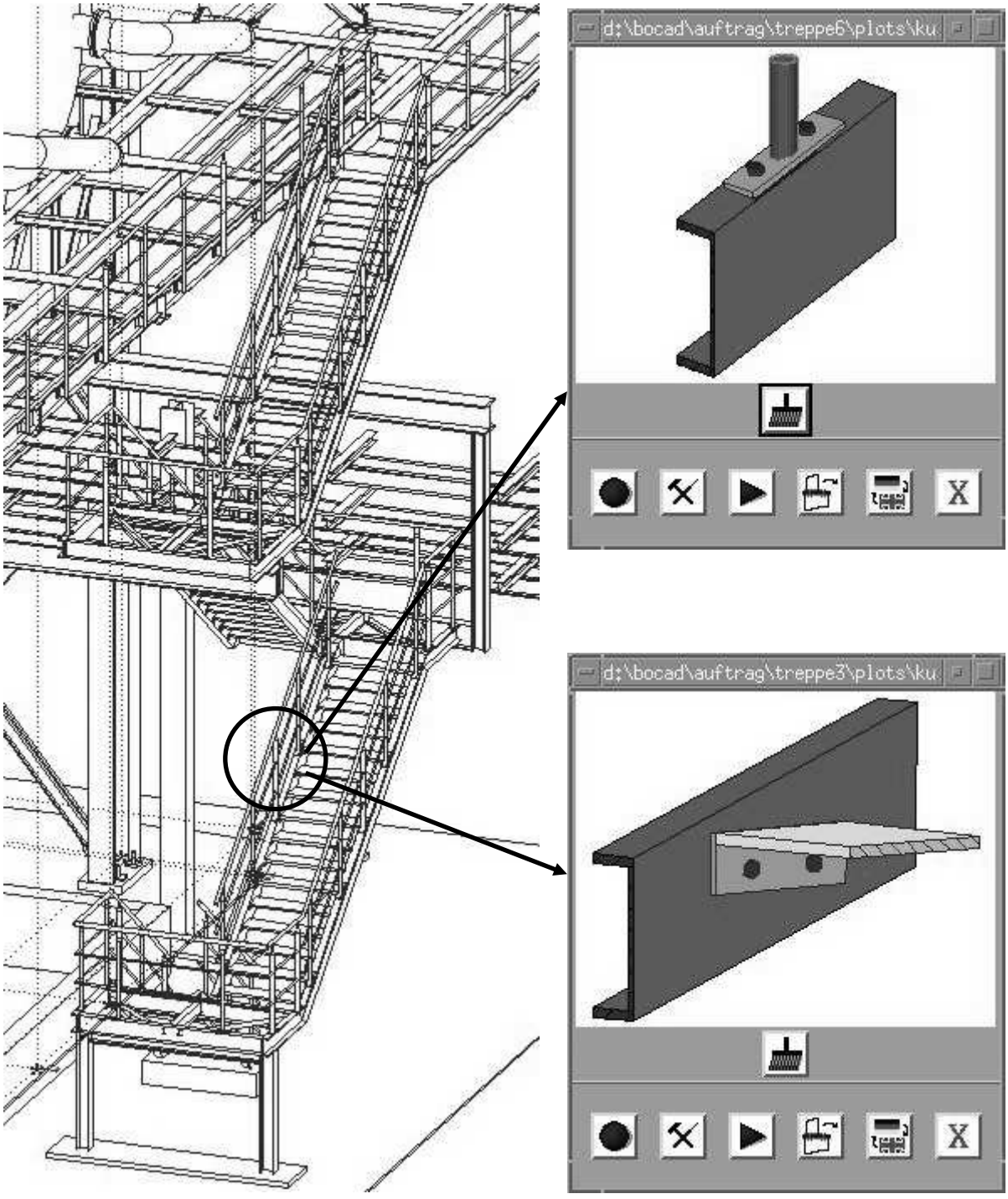


Abbildung 5-26: Kältespeicher über Tanklage (Ausschnitt im Treppenbereich) – Quelle: Ingenieurbüro Olker GmbH

6 Zusammenfassung und Ausblick

6.1 Zusammenfassung

Diese Arbeit hat zwei wesentliche, betriebsystemunabhängige Ergebnisse: die Bereitstellung der Grundlagen eines Rekorders, der Arbeitsschritte mit der Konstruktionslogik aufnimmt, und die Grundsteinlegung zur Berücksichtigung der Mehrsprachigkeit von Bausoftware.

Der Logikrekorder hat vordergründig die Aufgabe, den Baubedarf durch die Basiselemente der Informatik zu befriedigen und das Wissen des Bauwesens zu erweitern. Automatisierte, mehrfach verwendbare CAD-Methoden, die mit dem Logikrekorder vom Programmanwender selbst nach Bedarf erstellt werden können, dienen zur effizienten und gleichzeitig zur zeitökonomischen Projektdurchführung.

Die CAD-Methoden zur Erstellung gewünschter Anschlüsse, die mit dem Logikrekorder aufgenommen werden, werden mit Hilfe eines sehr gut verständlichen Konstruktionslogik-Assistenten in logisch entwickelten Schrittfolgen angefertigt. Somit sind die von der angewandten Software gebotenen Konstruktionsmethoden beliebig ergänzbar und wenn es notwendig ist, können sie durch den gleichen Assistenten in einem einfachen Prozess auch nachträglich geändert werden.

In laufenden Projekten kommen oft Änderungen vor, dass z. B. ein neues Profil in einem Anschluss gebraucht wird. In diesem Fall muss der alte Anschluss gelöscht werden, und die entsprechende, selbst aufgezeichnete CAD-Methode wird automatisch von Neuem ausgeführt statt des bisher gewohnten zeitaufwendigen, mehrschrittigen Anschlusswiederaufbaus.

Die Anforderungen der Mehrsprachigkeit wurden bei der Entwicklung des Logikrekorders im Verlauf der Arbeit beachtet. Die länderunabhängige Anwendbarkeit der Software ist angesichts der heutigen wirtschaftlichen Tendenz, wie die Globalisierung und die Ausbreitung der Märkte, unvermeidlich und deshalb von größter Wichtigkeit. Deshalb wurden alle Elemente der Nutzoberfläche genauestens untersucht.

Die Textelemente der Dialogboxen wurden als Variablen, die der aktuell ausgewählten Landessprache entsprechen, definiert. Die statischen Grafikdateien wurden zu dynamischen Grafikdateien weiterentwickelt, wobei für Textelemente - wie bei Dialogboxen - Variable verwendet werden. Im Ablaufprotokollfenster erscheint die automatische Ausgabe von Meldungen in der entsprechenden Landessprache.

Bei der Konzeption wurden – wo immer möglich - grafische Symbolen verwendet. Dabei wurden selbsterklärende, dreidimensionale Bilder an Stelle von textförmigen Aufschriften eingesetzt, genauso, wie bei der Anwendung von Leitbildern und von Auswahlbildern für die Wahl verschiedener Konstruktionsvarianten.

Der Logikrekorder wurde mehrsprachig und damit länderunabhängig angefertigt, um für dessen Einsetzung auf den ausländischen Märkten ohne zusätzlichen Mehraufwand auf dem schnellsten Weg größtmöglichen Erfolg zu garantieren.

6.2 Ausblick

Der Modul des Konstruktionslogik-Assistenten besteht aus einer Sammlung von zukunftsorientiert schlüssig strukturierten Fenstern, die eine schnelle, nachträgliche Bearbeitung problemlos ermöglichen. Die Ergänzung des Logikrekorders um weitere Funktionen und Arbeitsvorgänge wurde sorgfältig vorbereitet.

Der Logikrekorder wurde in seiner Anwendung in der Praxis zuerst im Treppenbau erfolgreich angewendet. Die hier erarbeiteten Erkenntnisse können wegen ihrer Übertragbarkeit auch in anderen Anwendungsbereichen, wie Beton-, Verbund-, Holz-, Glas- und Fassadenbau unterstützend und in jeder Hinsicht gewinnbringend verwendet werden.

Die Nutzoberflächenelemente des Konstruktionsprogramms wurden hinsichtlich der Mehrsprachigkeit gründlich untersucht und diskutiert. Die in der Arbeit behandelten Gedanken über die Fremdsprachlichkeit sind nicht nur auf neu entwickelte Systemteile oder Systeme übertragbar. Sie bilden gleichzeitig auch die wichtigsten Grundla-

gen des Reengineerings einsprachiger Systeme.

Weiterhin gilt nach Koch [14]: „Der Erfolg dieser Weiterentwicklungen hängt u. a. davon ab, dass sich die beispielhafte Zusammenarbeit mit der Praxis fortsetzen lässt.“

Literaturverzeichnis

[1] Chang, Y.: *Eine von Landessprachen unabhängige Nutzoberfläche mit intelligenten CAD-Objekten des Bauwesens.* Dissertation, Wuppertal 2002. Volltext-Link: http://elpub.bib.uni-wuppertal.de/edocs/dokumente/fb11/diss2002/chang/

[2] Streich, B., Weisgerber, W.: *Computergestützter Architekturmodellbau: CAAD-Grundlagen, Verfahren, Beispiele*. Birkhäuser, 1996. ISBN: 3-7643-5363-5.

[3] Rooney, J., Steadman, P.: *CAD: Grundlagen von Computer Aided Design.* Oldenbourg, 1990. ISBN: 3-486-20706-7.

[4] Pegels, G.: *Grundlagen vernetzt-kooperativer Planungsprozesse für Komplettbau mit Stahlbau, Metallbau, Holzbau und Glasbau*. Zwischenbericht, DFG-Schwerpunktprogramm 1103, WEB-Link: http://www.iib.bauing.tu-darmstadt.de/dfg-spp1103/de/index.html

[5] Pegels, G.: *Arbeitskreis Bauinformatik*. Denkschrift, Wuppertal 2001.

[6] Fähnrich, K.-P., Janßen, Ch., Groh, G.: *Werkzeuge zur Entwicklung graphischer Benutzungsschnittstellen: Grundlagen und Beispiele.* Oldenbourg, 1996. ISBN: 3-486-22889-7.

[7] Willms, A.: *C++ Programmierung: Programmiersprache, Programmiertechnik, Datenorganisation*. Addison-Wesley, 2001. ISBN: 3-8273-1627-8.

[8] Pegels, G.: *CAD – Konstruktion einer Halle*. Vorlesungsskript, Wuppertal 2001.

[9] Bergmann, U.: *C++ Eine Einführung.* Hanser, 1996. ISBN: 3-446-18498-8.

[10] Prinz, P., Kirch-Prinz, U.: *C++ Lernen und professionell anwenden.* mitp Verlag, 2002. ISBN:3-8266-0824-0.

[11] Balzert, H.: *Lehrbuch der Objektmodellierung: Analyse und Entwurf.* Spektrum Akademischer Verlag, 1999. ISBN: 3-8274-0285-9.

[12] *Der Brockhaus in Text und Bild 2002.* Brockhaus, 2001.
Der Brockhaus in Text und Bild 2003. Brockhaus, 2003. ISBN: 3-411-70677-5.

[13] Balzert, H.: *Objektorientierung in 7 Tagen.* Spektrum Akademischer Verlag, 2000. ISBN: 3-8274-0599-8.

[14] Koch, H.-D.: *Ansätze und grundlegende Elemente eines CAD/CAM-Systems für den Stahlbau*. Dissertation, Wuppertal 1991. ISBN: 3-925714-53-7.

[15] Westphal, Ch.: *Bauinformatik für virtuelle Unternehmen.* Shaker, 2002. ISBN: 3-8322-0798-8.

[16] Müller, B.: *Reengineering.* Teubner, 1997. ISBN: 3-519-02942-1.

[17] Paech, B.: *Aufgabenorientierte Softwareentwicklung.* Springer, 2000. ISBN: 3-540-65738-X.

[18] Balzert, H.: *Lehrbuch der Software-Technik.* Spektrum Akademischer Verlag, 2000. ISBN: 3-8274-0480-0.

[19] Ziegler, J.: *Eine Vorgehensweise zum objektorientierten Entwurf graphisch-interaktiver Informationssysteme.* Springer, 1997. ISBN: 3-540-62547-X.

[20] Steinrötter, M.: *CAD-Verfahren der Informationsgewinnung und –verarbeitung am dreidimensionalen Körpermodell von Stahlbauten.* Dissertation, Wuppertal 1995.

[21] Fabian, E.: *CAD-Konstruktionsmethoden komplexer Baugruppen – raumerschließende Stahlbaukonstruktionen*. Dissertation, Wuppertal 1996. ISBN: 3-89653-111-5.

[22] Janßen, Ch.: *Dialogentwicklung für objektorientierte, graphische Benutzungsschnittstellen.* Springer, 1996. ISBN: 3-540-60719-6.

[23] Koller, R.: *CAD: automatisiertes Zeichen, Darstellen und Konstruieren.* Springer, 1989. ISBN: 3-450-51062-1.

[24] Oberquelle, H.: *Sprachkonzepte für benutzergerechte Systeme.* Springer, 1987. ISBN: 3-540-18235-7.

[25] Fähnrich, K.-P.: *Methoden und Werkzeuge zur softwareergonomischen Entwicklung von Informationssystemen.* Jost-Jetter, 2000. ISBN:3-931388-31-X.

[26] Rauterberg, M., Spinas, P., Strohm, O., Ulich, E., Waeber, D.: *Benutzerorientierte Software-Entwicklung: Konzepte, Methoden und Vorgehen zur Benutzerbeteiligung.* vdf Hochschulverlag, 1994. ISBN: 3-7281-1959-8.

[27] Schindler, U., Klinner, K., Nestler, W.: *Microsoft Excel: Grundlagen der Makroprogrammierung.* Vieweg, 1990. ISBN: 3-528-14651-6.

[28] Hörmann, F.: *Excel-Makro-Programmierung.* IWT-Verlag, 1993. ISBN: 3-88322-395-6.

[29] Dumke, R.: *Makroprogrammierung: Einf. – Übungen – Praxis.* Fachbuchverlag, 1988. ISBN: 3-343-00228-3.

[30] Spona, H.: *Access Programmierung: professionelle VBA- und Makro-Programmierung.* Data Becker, 1999. ISBN: 3-8158-1331-X.

[31] Tanenbaum, A. S.: *Moderne Betriebssysteme.* Pearson Studium, 2002. ISBN: 3-8273-7019-1.

[32] Brittain, J. L., Head, G. O., Schaefer, A. T.: *AutoCAD 12 Makros und Menüs: individuelle Anpassungen.* te-wi-Verlag, 1993. ISBN: 3-89362-291-8.

[33] Greis, K. P.: *MS-WORD: Makro Programmierung.* Markt und Technik Verlag, 1990. ISBN: 3-89090-328-2.

[34] Maas, M.: Computergestützte Methode für das Entwerfen von Tragkonstruktionen. Dissertation, Wuppertal 2001. Volltext-Link: http://www.bib.uni-wuppertal.de/elpub/fb11/diss2001/maas/

[35] Schlensker, M.: *Verfahrensgrundlage des rechnerunterstützten Konstruierens mit Hilfe einer Konstruktions-Methodenbank*. Dissertation, Bochum 1982.

[36] Haake, H.-P.: *Grundlagen zum dreidimensionalen rechnergestützten Konstruieren im Stahlbau.* Dissertation, Bochum 1982.

[37] Stary, Ch.: *Interaktive Systeme: Software-Entwicklung und Software-Ergonomie.* Vieweg, 1996. ISBN: 3-528-15384-9.

[38] Klösch, R.: *Objektorientiertes reverse engineering: von klassischer zu objektorientierter Software.* Springer, 1995. ISBN: 3-540-58374-2.

Anhang

3.6 Quelltextausschnitte der Methode DRAW_DECKPL()

Gepickte Bemaßung bestimmen:

```
DRAW_DECKPL(param=1,testlv=0)
@
#SPRINGE,%l param;
#0; @ Eingabe über die Grafik/ Bemaßung/ vor der Wertenänderung ->
@     liest den Wert aus dem Grafik in der Box ein
#1; @ Eingabe oben/ nach der Wertenänderung
 %m NI3_GETSGNR(mfnr,4,'sind'); #WENN, mfnr, #RETURN,;
@
 #WENN, param=0, #SPRINGE, ende_1;
 %z txt1='   ';
 %mGETELM(mfnr,1,12351,'text','2edi','default ','txt',1:(16,80));
 %z text=%fWERT(1,txt1);
 #WENN, text<=0;
 #DANN, %sDRUCKE:%g1,1;
       %sDRUCKE:'Textangabe ist sinnlos';
       #RETURN,;
 ##;
#ende_1;
```

Zeichnung erzeugen:

```
#PUFFER, BODRAW_START;
@
@ Ueberschrift
@ ------------
 %mBMF_TEXT(mfnr,testlv,1:(-l2,b2+fak*3,0), 0,
         0, 1,schl,
         2, 0, 3,txthoe[txtind],0.9,
         1, 0.0, 1, 0, 0,
         1, 0.3, 15.0,
         1, 0, 0.18, 7,
         0, 0,2, , 0);
 #WENN, mfnr, #SPRINGE, ende_zchng;
@
@ Plattenabmessungen
@ ------------------
 %sWERTEFELD:koo_pl,-l2,-b2,0,  l2,-b2,0,
                  l2, b2,0, -l2, b2,0,
                 -l2,-b2,0;
```

```
@
 %mBMF_LINE(mfnr,testlv,1:('koo_pl'<>), 0,
          0, 1,3, 0, 0.18,
          1,0.3,15.0,1,0.3,15.0,7,
          0, 0, 2,0);
 #WENN, mfnr, #SPRINGE, ende_zchng;
@
@ Gesamtmass oben (in x)
@ ----------------------
 %sWERTEFELD:koo_mko, -l2, b2+fak*1 ,0,
                    l2, b2+fak*1 ,0,
                   -l2, b2+fak*1 ,0,
                   -l2, b2+fak*.2,0,
                    l2, b2+fak*.2,0;
@
 %mBMF_BEMA(mfnr,testlv,1:('koo_mko'<>), 0,
          0, 2, 1, '-',
          1, 1, 0, 0, 1,
         11, 0.1, 0.1,
          2, 1, 0, 1.0,
          2, 0, 0.18, 2,
          0, 2,txthoe[txtind], 0.9,
          1,10.0, 2);
 #WENN, mfnr, #SPRINGE, ende_zchng;
@
@ Gesamtmass seitlich (links)
@ --------------------------
 %sWERTEFELD:koo_mkl, -l2       , -b2,0,
                   -l2       ,  b2,0,
                   -l2       , -b2,0,
                   -l2-fak*.2, -b2,0,
                   -l2-fak*.2,  b2,0;
 %mBMF_BEMA(mfnr,testlv,1:('koo_mkl'<>), 0,
          0, 2, 1, '-',
          1, 1, 0, 0, 1,
         11, 0.1, 0.1,
          2, 1, 0, 1.0,
          2, 0, 0.18, 2,
          0, 2,txthoe[txtind], 0.9,
          1,10.0, 2);
 #WENN, mfnr, #SPRINGE, ende_zchng;
 #ende_zchng;
 #WENN, mfnr, %sDRUCKE:%g1,1;
@
#PUFFER, BODRAW_END;
```

```
@
 %z blf='                                ', erg=%fKETT(blf,blf),
 erg=%fLEGE(erg,'plots/deckpl.bmf_',1);
 %mSETELM(mfnr,6502,12351,'dp00','plan','default ',1:(16,80),1:(erg));
@
 #SPRINGE, ENDE;
#ENDE;
```

5.2.2 Assistent für Konstruktionslogik

Definitionsbereich:

```
------1050
#def ueberschrift 3. Schritt
#def frage Geben Sie Ihrem Arbeitsschritt einen Namen!
#def clbzurueck (1, Bo Combo Clb OK/ Bo Clb Makro <assist: 3050>/ Bo Clb Makro1
<copl_cons1: 35>/ Bo Create Combi Box <30: 21162: 3005>)
#def clbweiter (1,BoHoClbpDoClbLis<30:21162:10500>)
```

Dialogfensterkopf:

```
------2000
#in 30 21162 1
IBO (1)
BOX (1000   ,0),(1,Assistent #ueberschrift,-),  (3,0),(+350+350),(1,BoComboClbCanc)
TBL (1001,1000),(1,rclpick,-), (1,0,0),(6)
TBL (1100,1001,3,1,20),(1,options,-),(1,0,0),(6)
LAB (1101,1100,1,1,1),(1,   ,-)
LAB (1102,1100,2,2,1),(9,#frage,-)
LAB (1103,1100,3,3,1),(1,   ,-)
BGC (1100),(grey95)
BGC (1101),(grey95)
BGC (1102),(grey95)
BGC (1103),(grey95)
SEP (1200,1001,5,1,20),(1,sep1,-),(0,4)
/*
```

Allgemeine Variablendefinition:

```
------1
#def weiter  weiter >
#def mesweiter weiter
#def abbr  abbrechen
#def mesabbr Box verschwindet ohne Änderung
#def clbabbr (1,BoComboClbCanc)
```

```
#def zurueck  < zurück
#def meszurueck zurück
#def fertweit  fertig und weiter >
#def mesfertweit weiter
#def fertig  fertig
#def mesfertig Box verschwindet mit Änderung
#def rscnr 31
```

Schaltflächen:

```
------4001
SEP (1900,1001,20,1,20),(1,sep1,-),(0,4)
RCL (2000,1001,21,1,1),(1,rcl2,-),(0,1,1),(6)
LAB (2021,2000),(1,   ,-)
PUB (2022,2000),(1,#zurueck,-), (2,a,-),(#meszurueck),(0,0,1),#clbzurueck
LAB (2025,2000),(1,                    ,-)
PUB (2030,2000),(1,#weiter,-), (2,a,-),(#mesweiter),(0,0,1),#clbweiter
LAB (2031,2000),(1,   ,-)
PUB (2035,2000),(1,#abbr,-),(2,a,-),(#mesabbr),(0,0,1),#clbabbr
LAB (2036,2000),(1,   ,-)
```