Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document
D-91-06

Entwurf, vergleichende Bewertung und Integration eines Arbeitsplanerstellungssystems für Drehteile

Gerd Kamp

März 1991

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
D-6750 Kaiserslautern, FRG
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11, FRG
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern und Saarbrücken is a non-profit organization which was founded in 1988 by the shareholder companies ADV/Orga, AEG, IBM, Insiders, Fraunhofer Gesellschaft, GMD, Krupp-Atlas, Mannesmann-Kienzle, Philips, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct *systems with technical knowledge and common sense* which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- Intelligent Engineering Systems
- Intelligent User Interfaces
- Intelligent Communication Networks
- Intelligent Cooperative Systems.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Prof. Dr. Gerhard Barth
Director

Entwurf, vergleichende Bewertung und Integration eines Arbeitsplanerstellungssystems für Drehteile

Gerd Kamp

DFKI-D-91-06

© Deutsches Forschungszentrum für Künstliche Intelligenz 1991

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Deutsches Forschungszentrum für Künstliche Intelligenz, Kaiserslautern, Federal Republic of Germany; an acknowledgement of the authors and individual contributors to the work; all applicable portions of this copyright notice. Copying, reproducing, or republishing for any other purpose shall require a licence with payment of fee to Deutsches Forschungszentrum für Künstliche Intelligenz.

Entwurf, vergleichende Bewertung und Integration eines Arbeitsplanerstellungssystems für Drehteile

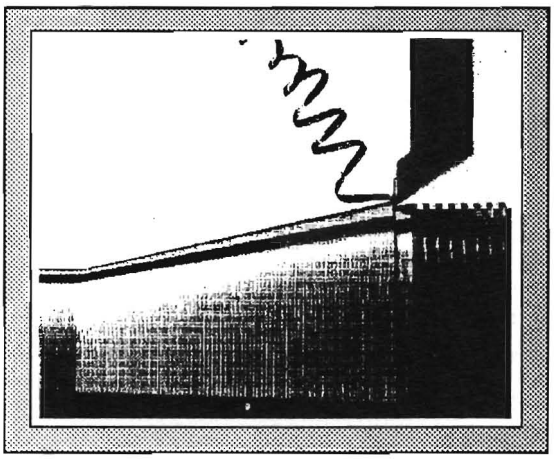
Gerd Kamp

März 1991

Betreuung:

Prof. Dr. Michael M. Richter
Dipl. Inform. Manfred Meyer
Dipl. Ing. Ralf Legleitner

AG Künstliche Intelligenz - Expertensysteme
Universität Kaiserslautern



Danksagung

Mein Dank gilt vor allem Vera Katic, ihre Unterstützung und Aufmunterung machten diese Arbeit erst möglich.

Auch den Mitarbeitern der Arbeitsgruppe Richter und des Projekttes ARC-TEC , ihnen voran Prof. Richter, Manfred Meyer und Peter Spieker bin ich dankbar für ihre Hilfe und die Atmosphäre, die es zu einer Freude machten dort zu arbeiten. Andreas Günter, Roman Cunis und den anderen Mitarbeitern am LKI in Hamburg möchte ich für ihre Geduld bei der Beantwortung von Fragen danken. Für ihre tatkräftige Unterstützung in den letzten Tagen dieser Diplomarbeit danke ich Nada Katic.

Nicht zuletzt danke ich meinen Eltern dafür, daß sie mir das Studium der Informatik und damit auch die Erstellung dieser Arbeit ermöglichten.

Inhaltsverzeichnis

1 Einführung

1.1 Hintergrund

Um den Anforderungen am Markt gerecht werden zu können, sind die Unternehmen gezwungen, immer komplexere, speziell auf den Kunden abgestimmte Produkte möglichst schnell in kleinen Losgrößen herzustellen. Gleichzeitig müssen aber aus Kosten- und Platzgründen die Lagerhaltung und die Anzahl der Maschinen verringert werden.

Diese gegenläufigen Ziele hofft man durch eine größere Flexibilität der Maschinen und eine Verringerung der Durchlaufzeit zu erreichen.

Eine größere Flexibilität wird durch den Einsatz von sogenannten flexiblen Fertigungssystemen oder auch flexiblen Fertigungszellen (FMS[1]) erzielt. Dies sind kleine Gruppen modernster Maschinen (z. B. ein CNC-Drehautomat, eine 5-Achsen CNC - Universalwerkzeugmaschine und eine Schleifmaschine), die mit Hilfe eines Transportsystem oder eines Roboters miteinander verbunden sind. Die FMS sind aufgrund ihrer Anpaßbarkeit durch entsprechende Steuerprogramme und der Vielseitigkeit der Maschinen in der Lage, ein breites Produktspektrum zu fertigen, so daß die Notwendigkeit zur Lagerhaltung weitgehend entfällt.

Um die Durchlaufzeiten zu minimieren, wurden im Rahmen von CIM[2] bis jetzt besonders die Bereiche Konstruktion (CAD[3]) und Bearbeitung (NC[4], CNC[5], DNC[6]) mit Hilfe des Computers unterstützt. Es gelang bis heute aber nicht, den Bereich der Arbeitsvorbereitung und Arbeitsplanung, der die Brücke zwischen CAD und CNC schlägt, zufriedenstellend zu automatisieren (CAP[7],CAPP[8]).

Konventionelle Systeme zur Arbeitsplanung unterstützen den Arbeitsplaner im wesentlichen im Rahmen der Zeitkalkulation, entscheidungstabellenbasierte Systeme eignen sich besonders für bestimmte Bereiche der Variantenplanung. Für das oben skizzierte Szenario ist aber in großem Maße generative Arbeitsplanung notwendig [JF89].

Bis jetzt standen aber die programmtechnischen Möglichkeiten dafür nicht zur Verfügung. Mit Hilfe der Methoden der KI, insbesondere der Expertensysteme, hofft man nun, das Erfahrungswissen und das fertigungstechnische Wissen des Arbeitsplaners erfassen und für die automatische Verarbeitung nutzbar machen zu können.

[1] **FMS** – Flexible Manufacturing Systems
[2] **CIM** – Computer Integrated Manufacturing
[3] **CAD** – Computer Aided Design
[4] **NC** – Numerical Control
[5] **CNC** – Computer Numerical Control
[6] **DNC** – Direct Numerical Control
[7] **CAP** – Computer Aided Planning
[8] **CAPP** – Computer Aided Process Planning

In der letzten Dekade wurden daher eine Reihe von Prototypen zur wissensbasierten generativen Arbeitsplanung entworfen, und eine große Anzahl von Forschungsinstituten arbeitet weltweit an der Lösung der anfallenden Probleme, da dies entscheidende Wettbewerbsvorteile in der Zukunft verspricht. Bis jetzt sind aber bei weitem nicht alle Probleme gelöst, so daß es Sinn macht, sich ebenfalls mit der generativen Arbeitsplanung zu beschäftigen.

1.2 Gliederung

Ziel dieser Arbeit ist es, einen Beitrag zur Automatisierung der generativen Arbeitsplanung zu leisten. Das Hauptaugenmerk liegt auf dem Entwurf und der Realisierung eines Systemes zur Arbeitsplanerstellung für Drehteile.

Da aber kein Entwurf eines neuen Systemes begonnen werden sollte, ohne sich vorher über den aktuellen Stand der Entwicklung zu informieren, enthält die Arbeit auch einen Teil, in dem bestehende Systeme zur Arbeitsplanung vorgestellt und bewertet werden. Weiterhin erschien es für das Verständnis der folgenden Teile notwendig, eine kurze Einführung in den Bereich der Arbeitsplanung zu geben.

Dadurch ergibt sich folgende Gliederung der Arbeit:

- Einführung in die Problemstellung der Arbeitsplanung
- Vergleich bestehender Systeme zur wissensbasierten generativen Arbeitsplanung
- Konzept eines eigenen Systems
- Detaillierung und Realisierung eines Teilbereiches der Problemstellung

1.2.1 Einführung in die Problemstellung der Arbeitsplanung

Für die Leser, die mit den Aufgaben der Arbeitsplanung nicht vertraut sind, wird in Kapitel 2 eine kurze Einführung in den Bereich der Arbeitsplanung und der dort benutzten Terminologie gegeben. Außerdem werden verschiedene Planungsprinzipien und Automatisierungsstufen der Arbeitsplanung vorgestellt.

1.2.2 Vergleich bestehender Systeme zur wissensbasierten generativen Arbeitsplanung

Kapitel 3 enthält einen Vergleich einer Reihe von Prototypen zur wissensbasierten generativen Arbeitsplanung. Dazu werden die Realisierungen einzelne Gesichtspunkte der Problemstellung in den Systemen gegenübergestellt, wobei besonderer Wert auf die eingesetzten KI-Methoden gelegt wird. Besonders interessante Realisierungen der Vergleichpunkte in bestimmten Systemen werden hier vorgestellt.[9]

1.2.3 Konzept eines eigenen Systems

Basierend auf den Erkenntnissen des vorhergehenden Teils, wird in Kapitel 4 das Konzept eines eigenen Systems zur Arbeitsplanerstellung präsentiert.In ihm wird versucht, einige der erkannten Mängel der anderen Ansätze zu vermeiden.

Nach einer Spezifikation der konkreten Problemstellung im Bereich der Arbeitsplanerstellung für Drehteile, wird ein Konzept zur ihrer Lösung vorgestellt.

Dabei wird untersucht, welche Module vollständig neu implementiert werden müssen, und für welche eventuell schon bestehende Software genutzt werden kann. Soll fremde Software benutzt werden, so stellt sich

[9] In Anhang A werden die untersuchten Systeme stichpunktartig vorgestellt, so daß man dort nachschlagen kann, falls man an einem bestimmten System interessiert ist.

die Frage, wie das entworfenene System in dieses Umfeld zu integrieren ist. Eine Analyse erfolgt für den Bereich der Datenbanken und der CAD-Systeme.

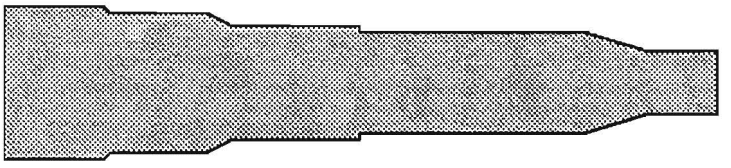
Für die einzelnen Module werden die nach unserer Meinung geeigneten Wissensrepräsentationsformalismen, Inferenzmechanismen und Planungstechniken angegeben.

Nicht in allen Fällen ist die generative Arbeitsplanung das bestgeeignetste Planungsprinzip. Um dieser Tatsache gerecht zu werden, wird eine Integration mit anderen Planungsprinzipien, vorrangig der Variantenplanung, skizziert.

1.2.4 Detaillierung und Realisierung eines Teilbereiches der Problemstellung

Da eine Implementierung des Gesamtsystems den Rahmen einer Diplomarbeit sprengen würde, und eine Planungsshell im Rahmen des ARC-TEC-Projektes sich erst in Entwicklung befindet, wurde der Bereich der Integration externer Programme für die Implementierung ausgewählt.

Hier wurde eine Client-Server Architektur für den netzwerkweiten Anschluß externer Programme entwickelt und implementiert, und als exemplarische Anwendungen ein Solid Modeler und eine objektorientierte Datenbank angeschlossen.



2 Arbeitsplanung - Begriffe und Probleme

2.1 Historisches

Ziel der Arbeitsplanung ist es, bei der Fertigung von Erzeugnissen ein Optimum aus Aufwand und Arbeitsergebnis zu erreichen. So wandelten sich mit der Fertigungstechnik auch die Anforderungen an die Arbeitsplanung. In den ursprünglichen Handwerksbetrieben war eine explizite Planung der Arbeit nicht notwendig. Erst mit der Industrialisierung und dem Aufkommen von Arbeitsteilung und Mechanisierung entstand die Notwendigkeit zur Arbeitsvorbereitung, die zunächst von den Meistern wahrgenommen wurde.

Schwächen der Fertigungstruktur veranlaßten Taylor, Gantt und andere zur Reform der Fabrikorganisation mit der Einführung von Funktionsmeistern und einem Formular- und Berichtswesen. Das Ziel war es, Richtlinien für die Reihenfolge und Dauer der auszuführenden Arbeiten für einen Vorarbeiter zur Verfügung zu stellen. Zur Kontrolle dieser Richtlinien wurde die Akkordarbeit eingeführt. Die Aufgaben der Arbeitsvorbereitung wurden in einem zentralen technischen Büro ausgeführt.

Die ersten Datenverarbeitungsanlagen im Bereich der Arbeitsplanung dienten der Lösung organisatorischer Probleme, wie der Verwaltung der Arbeitspläne. Der nächste Schritt in Richtung einer automatisierten Arbeitsplanung erfolgte durch die Einführung numerisch gesteuerter Werkzeugmaschinen. Für sie wurden maschinelle Programmiersysteme zur Erzeugung der Steuerinformation entworfen.

Durch die Entwicklung grafikfähiger Rechner und der damit verbundenen Unterstützung der Konstruktion (CAD) war es in einem nächsten Schritt möglich, Konstruktionsdaten in die Arbeitsplanung zu übernehmen. Parallel dazu wurden die NC-Programmiersysteme erweitert und verbessert. Eine Standardisierung erfolgte mit EXAPT und CLDATA. Die Entscheidungsfindung wurde durch den Einsatz von Entscheidungstabellensystemen vereinfacht.

Trotz all diesen Hilfsmitteln ist es bis jetzt aber nicht gelungen, die Arbeitsplanung vollständig zu automatisieren. Dies wird aber durch den wachsenden Anteil der Kleinserienfertigung und Bemühungen wie „Just-in-time"-Produktion immer dringlicher. Haupthindernis bei der Automatisierung war bisher der hohe Grad an heuristischen Teilvorgängen, die sich mit herkömmlichen Datenverarbeitungstechniken nicht automatisieren ließen.

Nun hofft man dieses Hindernis mit Hilfe der Methoden der Künstlichen Intelligenz, insbesondere der Expertensysteme zu überwinden. Weltweit werden dazu Systeme zur Arbeitsplanung unter Einsatz von Expertensystemtechniken entwickelt. Eine endgültige Lösung des Problems ist aber noch nicht in Sicht, die bisher vorgestellten Systeme befinden sich im Prototypenstadium und lassen noch einige Fragen offen. Ziel dieser Arbeit ist es, einen kleinen Beitrag zur weiteren Automatisierung der Arbeitsplanung zu leisten.

2.2 Aufgaben der Arbeitsplanung

Wie schon erwähnt, ist es das Ziel der Arbeitsplanung, ein Optimum aus Aufwand und Arbeitsergebnis bei der Fertigung von Erzeugnissen zu erreichen. Eine genauere Beschreibung der Aufgabenstellung gibt AWF/REFA[1]. Danach ist die Arbeitsplanung eines der Subsysteme der Arbeits- und Fertigungsvorbereitung.

Definition 2.1 (Arbeitsplanung) *Die Arbeitsplanung umfaßt alle einmalig auftretenden Planungsmaßnahmen, welche unter ständiger Berücksichtigung der Wirtschaftlichkeit die fertigungsgerechte Gestaltung eines Erzeugnisses oder die ablaufgerechte Gestaltung einer Dienstleistung, sichern [AWF69].*

Ergebnis der Arbeitsplanung ist der Arbeitsplan, dessen Definition nach REFA wie folgt lautet:

Definition 2.2 (Arbeitsplan) *Im Arbeitsplan sind die Ablaufabschnittfolge und die Arbeitssysteme beschrieben, die für eine schrittweise Aufgabendurchführung erforderlich sind. [REF74].*

Im Falle der Fertigungsplanung wird folgende, speziellere Definition gegeben:

Definition 2.3 (Arbeitsplan) *Im Arbeitsplan ist die Vorgangsfolge zur Fertigung eines Teiles, einer Gruppe oder eines Erzeugnisses beschrieben; dabei sind mindestens das verwendete Material sowie für jeden Vorgang der Arbeitsplatz, die Betriebsmittel, die Vorgabezeiten und die Lohngruppen angegeben [REF74, VDI74].*

Grundlage für die Arbeitsplanung ist ein in der Konstruktion definiertes technisches Objekt, für das unter Berücksichtigung der zur Verfügung stehenden Betriebsmittel die für die Fertigung notwendigen Fertigungsunterlagen hergestellt werden. Die dabei anfallenden Aufgaben werden unterteilt in kurzfristige und langfristige Planungsaufgaben. Dabei umfassen die kurzfristigen Planungsaufgaben die Ermittlung der für die Durchführung des Fertigungs- und Montageprozesses notwendigen Anweisungen. Die langfristigen Planungsaufgaben haben das Ziel, für ein zukünftiges Produktspektrum bessere Fertigungsbedingungen und verbesserte Abläufe zu bestimmen [Sp80].

Bild 2.1 zeigt die Aufgaben der Arbeitsplanung, getrennt nach kurz- und langfristigen Teilen.

2.2.1 Arbeitsplanerstellung

Aufgaben

Das Hauptaugenmerk dieser Diplomarbeit[2] liegt im Bereich der kurzfristigen Planungsaufgaben bei der Arbeitsplanerstellung. Daher sollen die dort anfallenden Aufgaben genauer vorgestellt werden. Eine ausführliche Beschreibung der in dieser Arbeit behandelten Problemstellung und eine Spezifikation der Teilaufgaben ist Gegenstand des Kapitels 4.

Die Arbeitsplanerstellung umfaßt nach [SK83] die folgenden Teilaufgaben:

- Ausgangsteilbestimmung
- Arbeitsvorgangsfolgeermittlung
- Maschinenauswahl
- Fertigungshilfsmittelzuordnung
- Vorgabezeitbestimmung

[1] Ausschuß für wirtschaftliche Fertigung e.V./ Verband für Arbeitsstudien und Betriebsorganisation e.V.
[2] und des Projektes ARC-TEC

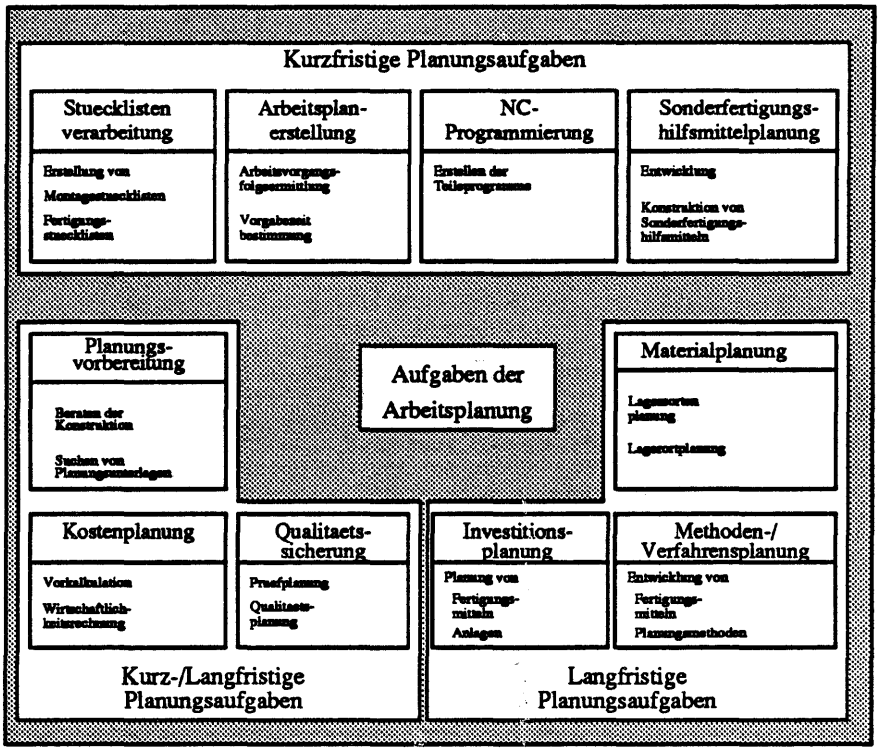


Abbildung 2.1: Teilaufgaben der Arbeitsplanung

Ausgangsteilbestimmung Bei der Ausgangsteilbestimmung können zwei Fälle unterschieden werden:

1. Das Ausgangsteil (Rohteil) wird bei der Konstruktion festgelegt, z. B. bei der Verwendung von Gußteilen.[3]
2. Ein zur Fertigteilbeschreibung passendes Halbzeug wird aus einer vorgegebenen Menge ausgewählt.

Maschinenauswahl Technische und wirtschaftliche Überlegungen spielen bei der Maschinenauswahl eine Rolle. Zu den technologischen Kriterien zählen u. a. die Arbeitsraumabmessungen, das Gewicht und die geforderte Oberflächengüte.

Einfluß auf die Wirtschaftlichkeit einer Maschine nehmen Faktoren wie Losgröße, Maschinenstundensatz, Verarbeitungsgeschwindigkeit und Produktstatus[4].

Ermittlung der Arbeitsvorgangsfolge Die zentrale Aufgabe der Arbeitsplanerstellung ist die Arbeitsvorgangsfolgeermittlung. Für die dort ausgewählten Arbeitsvorgänge werden in den nachfolgenden Schritten Fertigungshilfsmittel, NC-Programme und Vorgabezeiten bestimmt. Zugleich ist sie aber auch die schwierigste Teilaufgabe. Sie entzog sich bisher der Algorithmisierung und wurde durch die Kreativität und das Fachwissen von hochqualifizierten Arbeitsplanern gelöst.

Was ist nun eigentlich die „Arbeitsvorgangsfolge"? Dazu gibt [SK83] die folgenden Definitionen:

Definition 2.4 (Arbeitsvorgangsfolge) *Die Arbeitsvorgangsfolge ist die Reihenfolge der Tätigkeiten, die die technischen Randbedingungen und die Wirtschaftlichkeit der Fertigung berücksichtigt.*

Definition 2.5 (Arbeitsvorgang) *Ein Arbeitsvorgang ist die Bearbeitung, die zusammenhängend auf einer Maschine oder einem Handarbeitsplatz ausgeführt wird.*

[3] Gußteile kommen bevorzugt bei besonders großen Werkstücken und Losgrößen zum Einsatz.
[4] z. B. Protoyp, Vorserie, Serie

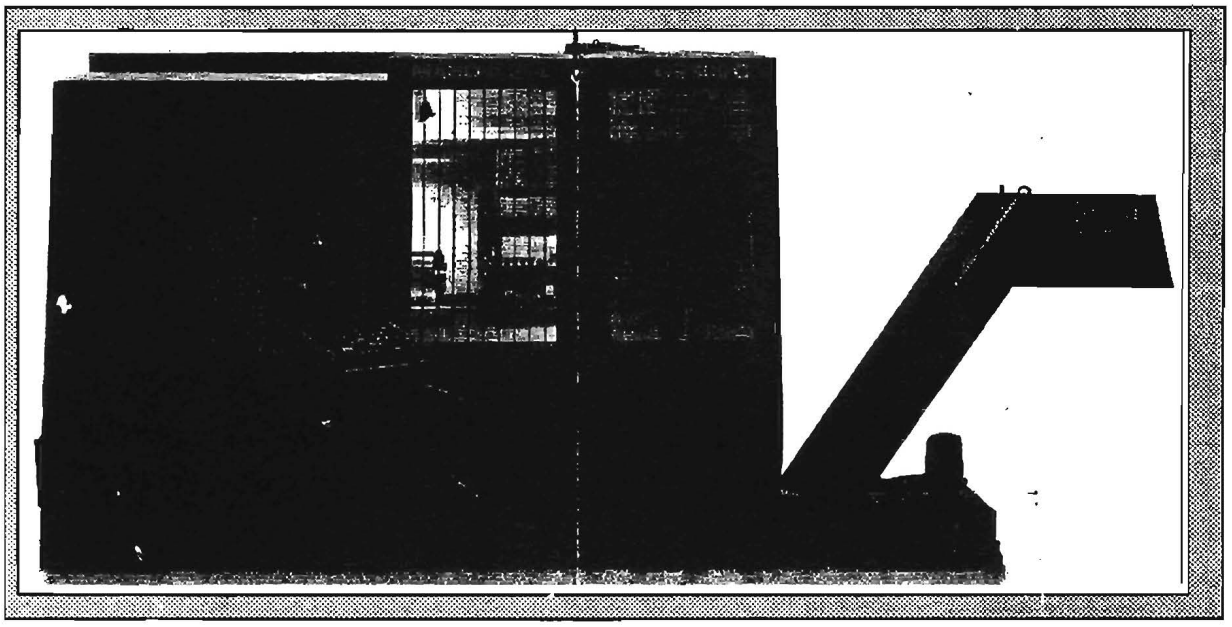


Abbildung 2.2: Eine CNC-Drehmaschine

Die so erreichte grobe Untergliederung des Arbeitsplanes wird meistens weiter verfeinert. Stellt man bei der Unterteilung fertigungstechnische Merkmale in den Vordergrund, so erhält man eine Gliederung in Teilarbeitsvorgänge, die evtl. noch weiter in Verfahrwege zerlegt werden. Beispiele für einen Arbeitsvorgang wären nach dieser Definition Sägen, Drehen, Fräsen, für einen Teilarbeitsvorgang Längsdrehen, Plandrehen, Einstechen etc[5].

Die benötigten Hilfsmittel können ebenfalls zur Aufteilung eines Arbeitsvorganges benutzt werden. Dann läßt sich der Arbeitsvorgang in einzelne Aufspannungen und diese wiederum in Teilaufgaben mit gleichem Werkzeug unterteilen.

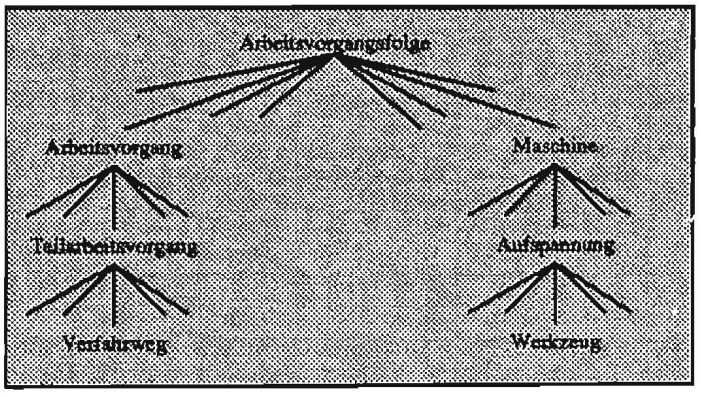


Abbildung 2.3: Unterschiedliche Sichten auf die Arbeitsvorgangsfolge

Die Schwierigkeiten bei der Arbeitsvorgangsfolgenbestimmung liegen in der Vielzahl der möglichen Fertigungsverfahren und Fertigungsmittel[6] und in der Zuordnung dieser Verfahren zu vorhandenen Objektgeometrien.

Erschwerend kommt hinzu, daß meist nicht irgendein, sondern ein „optimaler" Arbeitsplan gewünscht ist.

[5] Durch das Aufkommen von NC-Bearbeitungszentren wird es allerdings möglich, das mehrere Arbeitsvorgänge nach der obigen Definition auf einer Maschine ausgeführt werden können.

[6] So gibt es zum Beispiel ..... Möglichkeiten zur Kombination von Wendeschneidplatte und Werkzeughalter.

Das Optimalitätskriterium kann wie die Auswahl der Fertigungsverfahren je nach Wahl der Eingangsparameter stark variieren. Bestimmende Parameter sind z. B. Losgröße, vorhandenen Resourcen und zur Verfügung stehende Zeit.

Fertigungshilfsmittelzuordnung Unter Fertigungshilfsmitteln werden Werkzeuge, Vorrichtungen und Spannmittel verstanden. Den vorher bestimmten Teilen des Arbeitsplanes müssen nun Fertigungshilfsmittel zugeordnet werden. Dabei gilt es, bestimmte Optimalitätskriterien zu beachten. Ist es für ein Teil des Arbeitsplanes nicht möglich, ein passendes Fertigungshilfsmittel zu finden, muß der Arbeitsplan entsprechend geändert werden.

Bei der Klein- und Mittelserienfertigung ist die wichtigste Teilaufgabe der Fertigungshilfsmittelzuordnung die Werkzeugbestimmung. Diese läßt sich bei der heutzutage üblichen Verwendung von Wendeschneidplatten auf eine Auswahl aus dem zur Verfügung stehenden Sortiment zurückführen.

Werden Sonderhilfsmittel benötigt, so muß für deren Konstruktion und Fertigung gesorgt werden.

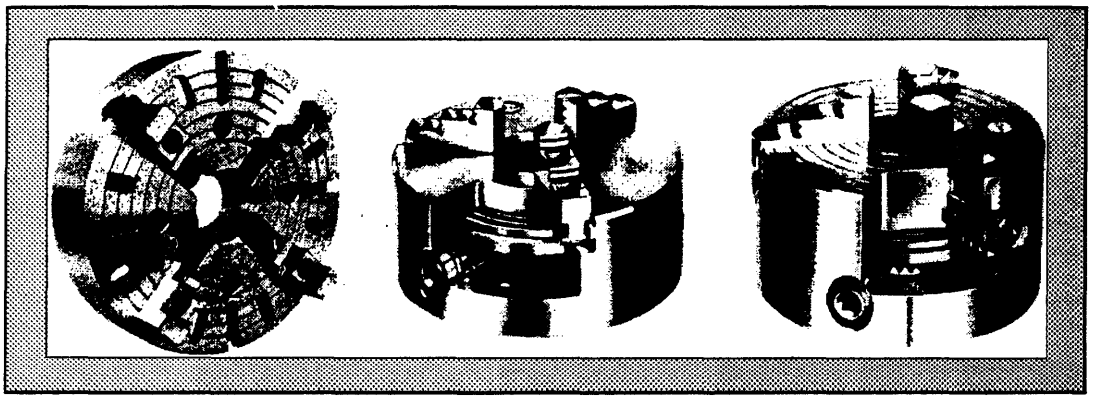


Abbildung 2.4: Verschiedene Spannfutter

NC-Programmierung Aus dem Arbeitsplan müssen im Falle der NC-Bearbeitung die entsprechenden NC-Programme abgeleitet werden. Dieser Prozeß läßt sich heute weitestgehend automatisieren. Das Hauptproblem lag lange Zeit bei den unterschiedlichen Steuerungssystemen der Werkzeugmaschinen. Seit der Einführung von Normen wie CLDATA kann es aber als gelöst betrachtet werden.

Zwischen Arbeitsvorgangsfolgenermittlung und NC-Programmierung steht die Schnittdatenbestimmung, in der Daten wie Schnitttiefe, Vorschub und Schnittgeschwindigkeit festgelegt werden.

Vorgabezeiten Die Bestimmung der Vorgabezeiten ist als Grundlage der Kostenrechnung Teil der Arbeitsplanerzeugung. Während bei Klein- und Mittelserien ihr Gewicht gering ist, ist sie bei der Massenfertigung von besonderer Bedeutung, da die Taktzeit und damit die Investitionsplanung von ihr abhängen.

Die Vorgabezeit setzt sich aus Haupt-, Neben-, Rüst-, Grund- und Verteilzeit zusammen. Mathematisch exakt läßt sich davon nur die Hauptzeit erfassen, Grundlage der anderen sind empirisch ermittelte Richtwerte.

Benötigte Daten

Zur Lösung der Teilaufgaben stehen dem Arbeitsplaner eine Vielzahl von Daten zur Verfügung, deren Inhalt und Struktur jedoch sehr unterschiedlich sind. Tabelle 2.1 zeigt eine bei der manuellen Arbeitsplanerstellung übliche Zuordnung der Daten zu den Teilaufgaben:

Die Rohteilbestimmung erfolgt mit Hilfe von Katalogen der lagerhaltigen Materialien, soweit das Rohteil nicht bereits durch die Konstruktion vorgegeben ist.

Funktion	Planungsunterlagen zur Arbeitsplanerstellung		
Ausgangsteilbestimmung	Materiallagerkatalog	Berechnungshilfen	
Arbeitsvorgangsfolgeermittlung	ähnliche Werkstücke	vergleichbare Arbeitsvorgangsfolgen	Standardarbeitspläne
Maschinenauswahl	Maschinenverzeichnis	Kostenstellen	
Fertigungshilfsmittelzuordnung	Werkzeugkatalog	Vorrichtungskatalog	Meßmittelkatalog
Vorgabezeitbestimmung	Nomogramm	Zeitrichtwerttabelle	Diagramme

Tabelle 2.1: Funktionen der Arbeitsplanerstellung und benötigte Daten

Zur Arbeitsvorgangsfolgenermittlung werden „ähnliche" Arbeitspläne und Standardsarbeitspläne herangezogen.

Maschinenkataloge mit Angaben über die Einsatzgebiete der Maschinen, Kostenstellen und Maschinenstundensätze kommen bei der Maschinenauswahl zum Einsatz.

Bei der Bestimmung der Fertigungsmittel wird auf Werkzeug-, Meß-, und Prüfmitteldateien sowie Vorrichtungungskataloge zurückgegriffen.

Zeiten und Kosten werden mit Hilfe von Richtwerttabellen und Material-, Werkzeug-, und Maschinendaten ermittelt.

Darüberhinaus greifen alle Teile auf eine Werkstückbeschreibung zu, die neben geometrischen auch technologische Daten enthält.

Für eine automatisierte Arbeitsplanerstellung ist es notwendig, all diese Daten in maschineller Form bereitzustellen, bzw. auf die bereits in unterschiedlichen Formen gespeicherten Daten zugreifen zu können.

Eine vollständige Automatisierung erfordert auch die die Formalisierung des Planungswissens des Arbeitsplaners und die Verfügbarkeit in verarbeitbarer Form.. Es ist dies Wissen über Art und Reihenfolge der Bearbeitung, Wissen über „günstige" Maschinen, Werkzeuge etc. Dieses Wissen ist oft heuristischer Art oder Allgemeinwissen. Um es darstellen zu können, werden Methoden der Künstlichen Intelligenz benötigt.

2.3 Arten der Arbeitsplanung

Nachdem wir im vergangenen Abschnitt gesehen haben, welche Aufgaben bei der Arbeitsplanung und der Arbeitsplanerstellung anfallen, sollen nun die Techniken vorgestellt werden, die zur Lösung dieser Aufgaben eingesetzt werden.

Man benutzt unterschiedliche Planungsprinzipien, deren Übergänge jedoch durchaus fließend sind.

Folgende Prinzipien können unterschieden werden :

- Arbeitsplanverwaltung
- Wiederholplanung
- Variantenplanung
- Anpassungsplanung
- Neuplanung

Welches Prinzip zum Einsatz kommt, richtet sich nach den betrieblichen Anforderungen wie z. B. Fertigungsart und Produktspektrums.

2.3.1 Arbeitsplanverwaltung

Die Arbeitsplanverwaltung[7] ist weniger eine Methode zur Erstellung neuer Pläne, sondern zur effizienten Nutzung vorhandener Arbeitspläne.

Kennzeichnend für die Arbeitsplanverwaltung ist das Archivieren von Arbeitsplänen mit Code- oder Klassifizierungssystemen. Benötigt werden dazu auftragsneutrale Arbeitspläne der entsprechenden Losgröße, in die bei Bedarf die entsprechenden Auftragsdaten eingesetzt werden.

Die auftragsneutralen Pläne müssen mit Hilfe anderer Planungsprinzipien hergestellt werden. Für einen erfolgversprechenden Einsatz der Arbeitsplanverwaltung muß die Fertigung eine entsprechende Wiederholhäufigkeit aufweisen.

2.3.2 Wiederholplanung

Wiederholplanung ist ebenfalls kein Planungsprinzip zur Erstellung von Arbeitsplänen neuer Werkstücke. Es kommt zum Einsatz, wenn sich die Fertigungsbedingungen des Teiles ändern.

Dies kann seine Gründe im technischen Fortschritt, der neue Maschinen erlaubt, in einer Änderung der Auftragslage, die eine größere Produktionsrate des Teiles notwendig macht, oder einer Änderung des Entwicklungsstadiums des Teiles, z. B. vom Prototyp zur Vorserie, haben.

Dadurch werden i. a. Änderungen am Arbeitsplan notwendig[8], die die Wiederholplanung vornimmt.

2.3.3 Variantenplanung

Die Variantenplanung kommt als Planungsprinzip bei festumgrenzten Werkstückspektren zum Einsatz. Das Spektrum der zu fertigenden Teile wird in Teilefamilien zerlegt und jeder Teilfamilie ein Standardarbeitsplan zugeordnet. Dieser bildet die Grundlage für die Pläne aller Varianten innerhalb der Teilefamilie.

Die Pläne für die Varianten entstehen aus dem Standardarbeitsplan durch die Änderung von Parameterwerten einzelner Teilarbeitsvorgänge. Die Art und Reihenfolge der Teilarbeitsvorgänge bleibt unverändert, die Variation der Parameterwerte ist gering.

Damit reduziert sich die Planung auf die Bestimmung der geeigneten Teilefamilie und die Berechnung der jeweiligen Parameterwerte. Die Auswahl der Teilefamilie erfolgt mit Hilfe von Klassifizierungsschemata, in die häufig 50 - 60 Daten des Werkstücks Eingang finden [JF89].

Dem Nachteil der geringen Flexibilität steht bei der Variantenplanung der Vorteil kurzer Rechenzeiten gegenüber.

2.3.4 Anpassungsplanung

Die Anpassungsplanung erweitert die Möglichkeiten der Variantenplanung um das Löschen, Hinzufügen und Ändern von Teilarbeitsvorgängen. Dadurch wird eine erheblich größere Flexibilität erreicht, die die Erstellung neuer Planungsdaten möglich macht.

Die Änderungen an den vorhandenen Arbeitsplänen erfolgen entweder durch Algorithmen oder durch den Dialog mit dem Arbeitsplaner.

Für einen erfolgreichen Einsatz der Anpassungsplanung ist die Verwendung eines geeigneten Klassifizierungsschlüssels dringend notwendig.

[7] Auch Regenerierungsprinzip oder Wiederholplanung

[8] Hat z. B. die neue Maschine ein Werkzeugspeicher mit mehr Plätzen, können evtl. speziellere Werkzeuge für die einzelnen Teilarbeitsvorgänge benutzt werden.

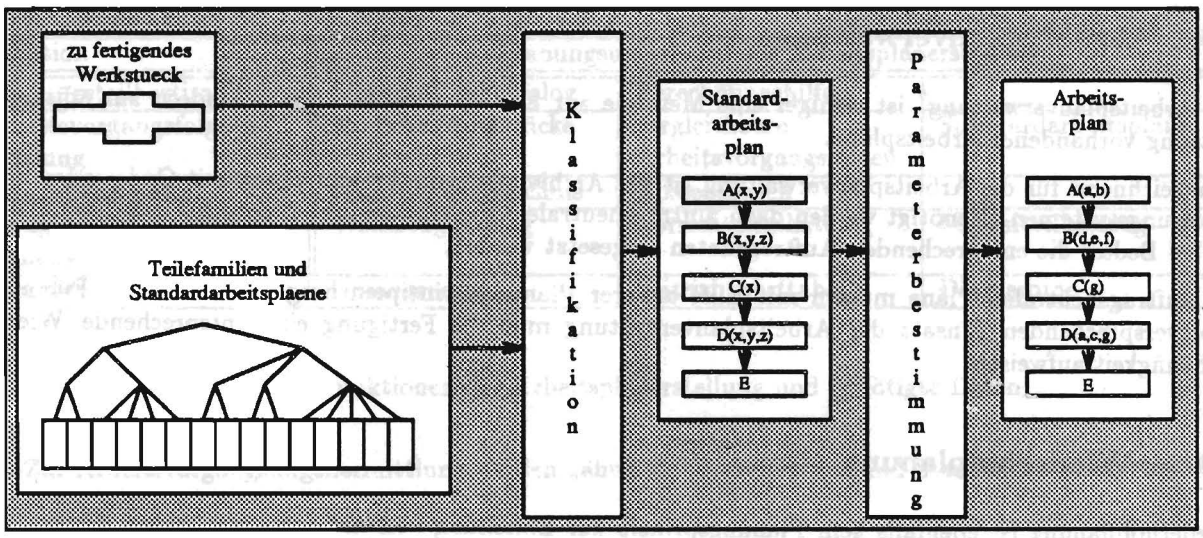


Abbildung 2.5: Das Prinzip der Variantenplanung

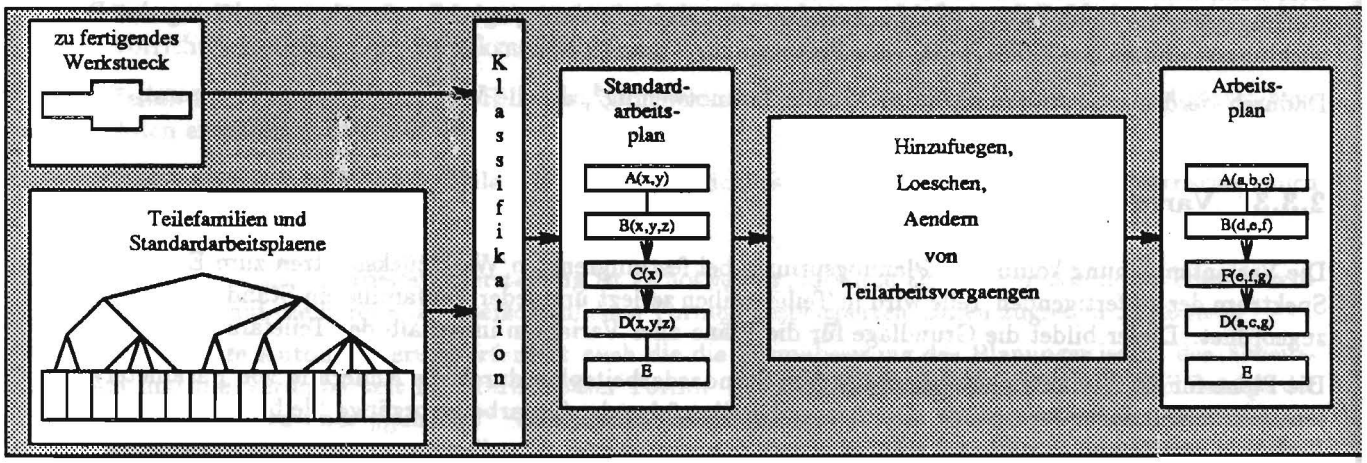


Abbildung 2.6: Das Prinzip der Anpassungsplanung

2.3.5 Neuplanung (Generative Arbeitsplanung)

Neuplanungssysteme basieren auf dem sogenannten Generierungsprinzip, d. h. daß aus den jeweils eingegebenen Daten ein vollständiger Arbeitsplan erzeugt wird. Je nach Eingangsdaten unterscheidet man Neuplanung mit Arbeitsvorgangsbeschreibung und Neuplanung mit vollständiger Roh- und Fertigteilbeschreibung.

Neuplanung mit Arbeitsvorgangsbeschreibung Teilaufgaben der Neuplanung mit Arbeitsvorgangsbeschreibung sind im wesentlichen die Generation von aktuellen technologischen Daten, die Auswahl der Fertigungsmittel und Zeitberechnungen.

Neuplanung mit vollständiger Roh- und Fertigteilbeschreibung Bei der Neuplanung mit vollständiger Roh- und Fertigteilbeschreibung kommt als weitere Teilaufgabe die Zuordnung von Arbeitsoperationen aus Elementen der Werkstückgeometrie hinzu

Im weiteren Verlauf dieser Arbeit wird unter dem Begriff Neuplanung immer Neuplanung mit vollständiger Roh- und Fertigteilbeschreibung verstanden.

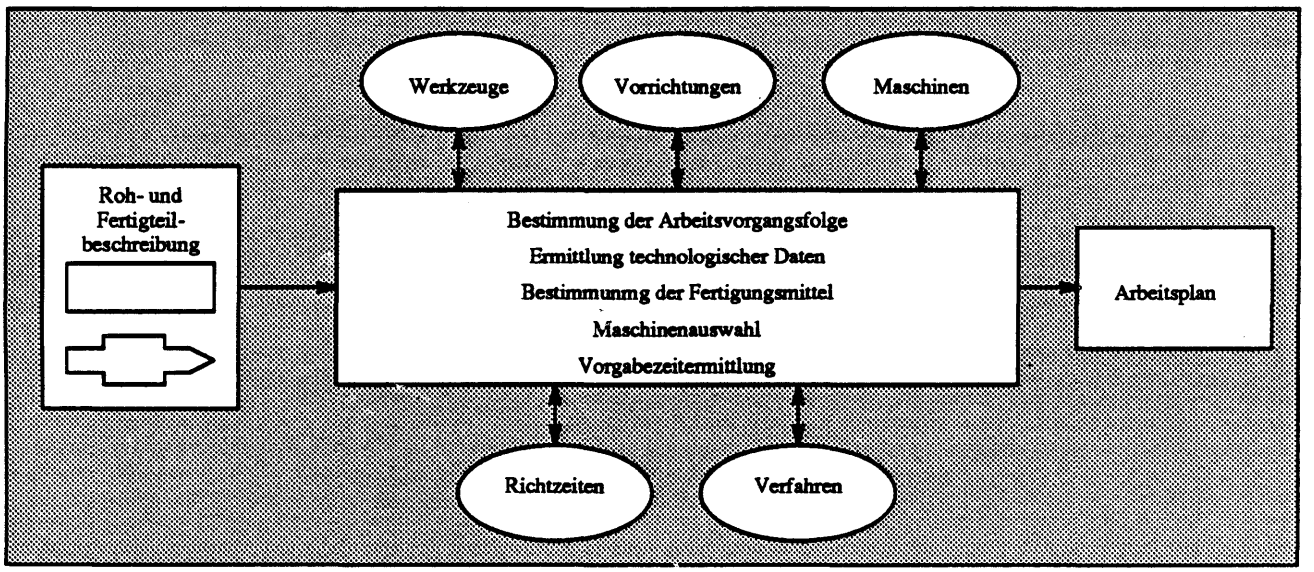


Abbildung 2.7: Das Prinzip der Neuplanung

2.3.6 Kriterien zur Auswahl des Planungsprinzips

Es existieren keine exakten Kriterien, wann welches Planungsprinzip zum Einsatz kommen sollte. Im folgenden wird eine grobe Auswahl von Einflußfaktoren aufgeführt, die [Sp72] entnommen sind:

- Art und Umfang der geplanten Rationalisierung der Arbeitsvorbereitung,
- betriebliches Werkstückspektrum,
- Spektrum der möglichen Fertigungsverfahren sowie vorhandene Betriebsmittel, Variationsmöglichkeit und Variationsbreite der in Betracht gezogenen Werkstücke hinsichtlich Standardarbeitsplänen,
- Kapazität, Eignung und Auslastung vorhandener Datenverarbeitungsanlagen,
- Losgröße der behandelten Werkstücke.

2.4 Automatisierung der Arbeitsplanung

Zur Unterstützung der im letzten Abschnitt vorgestellten Planungsprinzipien wurden schon früh Datenverarbeitungsanlagen eingesetzt. Im Laufe der Zeit haben sich verschiedene Automatisierungsstufen herausgebildet, die im folgenden vorgestellt werden. Danach wird untersucht, welcher Automatisierungsgrad für welches Planungsprinzip benötigt wird, und inwieweit eine entsprechende Rechnerunterstützung bereits vorliegt.

2.4.1 Automatisierungsstufen

Die einzelnen Automatisierungsstufen der Arbeitsplanung sind (in der Reihenfolge zunehmender Automatisierung):

- Manuelle Planung
- Interaktive Planung
- Planung mit Makros
- Halbautomatische Planung
- Vollautomatische Planung

Manuelle Planung

In der traditionellen Vorgehensweise entschlüsselt der Arbeitsplaner Zeichnungen, indem er auf der Grundlage von Informationen wie zum Beispiel Text, Abmessungen, Ansichten und Teile-Charakteristiken entsprechende Entscheidungen trifft und Aktionen auslöst. In diesen scheinbar intuitiv ablaufenden Prozeß werden in hohem Maße Erfahrungswissen und Know-How eingebracht. Bei einer Verfeinerung des Arbeitsplanes werden die „intuitiven" Entscheidungen durch Nachschlagen von Daten, Berechnungen etc. konkretisiert.

Interaktive Planung

Bei dieser Art der Rechnerunterstützung werden dem Planer Fragen gestellt, die er im Dialog beantwortet. Dadurch legt er zum Beispiel Maschine und Art der Arbeit fest. In den nächsten Fragen wird die Auswahl präzisiert. Der Arbeitsplaner hat die vollständige Kontrolle über den Fertigungsplan, er wird jedoch von Routinetätigkeiten wie Nachschlagen in Katalogen und Berechnungen befreit.

Planung mit Makros

Im Laufe der Zeit haben sich für bestimmte Teilprobleme Standardlösungen herausgebildet. Dies können z. B. bestimmte Verfahrensreihenfolgen oder Schnittzyklen sein. Bei der Planung mit Makros kann der Arbeitsplaner diese Standardlösungen im Bedarfsfalle benutzen, evtl. werden sie ihm sogar vorgeschlagen.

Halbautomatische Planung

Bei der halbautomatischen Planung erfolgt die Ermittlung der einzelnen Teile des Arbeitsplanes durch den Rechner, der Arbeitsplaner muß nur noch fehlende Parameter (z. B. Maße) bereitstellen.

Vollautomatische Planung

Die Konstruktionsphase des Werkstücks wird so gestaltet, daß eine vollständige Beschreibung des Teiles vorliegt. Diese wird vom Arbeitsplanungssystem zur Erzeugung des Arbeitsplanes benutzt.

2.4.2 Rechnerunterstützung der verschiedenen Planungsprinzipien

Aufgrund ihrer unterschiedlichen Komplexität sind die verschiedenen Planungsprinzipien mehr oder weniger stark automatisiert. So konnten mit den folgenden DV-Techniken unterschiedliche Planungsprinzipien realisiert werden:

1. Datenbanken

   Am einfachsten fiel die Automatisierung der Arbeitsplanverwaltung, die durch den Einsatz von Datenbanken gelang.

2. Konventionelle Arbeitsplanungssysteme

   Mit Hilfe konventioneller Systeme konnte die interaktive Planung für alle Planungsprinzipien weitestgehend realisiert werden.

3. Entscheidungstabellen

   Entscheidungstabellen halfen, die Steuerung der Fragenreihenfolge zu vereinfachen, was es ermöglichte halbautomatische Variantenplanung zu realisieren. Die Probleme liegen hier hauptsächlich in der Wahl des Klassifizierungsschlüssels.

Mit dem oben beschriebenen Methoden gelang es aber nicht, mehr als eine interaktive Planung für die Wiederholplanung, die Anpassungsplanung oder die generative Planung bereitzustellen.

Dies hat seine Hauptursache darin, daß es mit diesen Hilfsmitteln nicht oder nur sehr schlecht möglich war, die Arbeitsvorgangsfolge festzulegen. Dafür werden aber 30 - 80 % der Planungszeit benötigt, sie stellt somit die zentrale Aufgabe dar.

Man hofft, dieses Problem nun durch die größere Flexibilität der Steuerung, die durch den Einsatz von Expertensystemen erreicht wird, lösen zu können.

Neben diesem Punkt listet [JF89] noch folgende Bereiche auf, in denen der Einsatz wissensbasierter Methoden Vorteile mit sich bringt:

- Generative Arbeitsplanung

  Wissensbasierte Methoden ermöglichen die Merkmalsextraktion aus CAD–Zeichnungen; das Wissen zur Bestimmung der Arbeitsgänge läßt sich besser repräsentieren.

- Alternativplanung

  Alternativplanung (d.h. die Erwägung anderer Materialien, Maschinen etc.) führt bei Entscheidungstabellensystemen zu einem exponentiellen Anwachsen der Anzahl der Regeln und Bedingungen, und ist daher nicht machbar. Mit Expertensystemen kann sie zumindest teilweise realisiert werden.

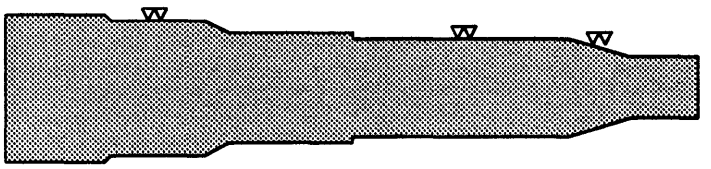
- Planung und Optimierung der Arbeitsvorgangsfolge

  Entscheidungstabellen lassen sich für Planungs- und Optimierungsaufgaben nicht einsetzen. Bei Expertensystemen können diese Aufgaben durch heuristische Suche (z.B. A*-Algorithmus) oder Integration anderer Optimierungsverfahren gelöst werden.

Des weiteren bieten wissensbasierte Systeme Vorteile durch die

- größere Mächtigkeit der Wissensrepräsentation,
- höhere Flexibilität in der Abarbeitung des Wissens,
- Erklärungsfähigkeit des Systems.

Um von einer halbautomatischen zu einer vollautomatischen Planung zu gelangen, ist es notwendig, das Werkstück vollständig zu beschreiben. Dazu ist entweder eine Änderung der Konstruktionphase notwendig, derart, daß alle Daten explizit bereitgestellt werden, oder die benötigten Daten müssen aus den zur Verfügung gestellten extrahiert werden. Beide Ansätze wurden und werden verfolgt; genauer wird auf sie im Abschnitt 3.7 eingegangen.



3 Vergleich und Bewertung bestehender Systeme

In diesem Kapitel werden eine Anzahl von Prototypen zur wissensbasierten generativen Arbeitsplanerstellung vergleichend vorgestellt. Dazu werden in den nachfolgenden Abschnitten verschiedene Aspekte der Systeme auf die verwendeten Methoden und Lösungsansätze untersucht. Dabei wird versucht, die zum Teil doch recht unterschiedlichen Begriffsbildungen der einzelnen Systeme zu vereinheitlichen und grobe Mißverständisse zu bereinigen. Die Vorstellung der Systeme erfolgt hier nur insoweit, als das Aussagen darüber gemacht werden wie die einzelnen Gesichtspunkte in dem jeweiligen System realisiert sind. Möchte der Leser genaueres über ein System erfahren so sei er auf Anhang A verwiesen, in dem die verschiedenen Systeme stichpunktartig vorgestellt werden. Dort sind auch Verweise auf die Originalliteratur zu finden.[1]

Die untersuchten Aspekte umfassen u. a.:

- Hard- und Softwareplattformen
- Umfang des bearbeitbaren Teilespektrums und der bearbeiteten Teilaufgaben
- Integration von Varianten- und Anpassungsplanung
- Integration in das betriebliche Umfeld (insbesondere von Datenbanken und CAD-Systemen)
- Repräsentation des Werkstücks und der Werkstatt
- Verwendete KI-Methoden und Planungstechniken.

3.1 Verwendete Hard- und Softwareplattformen

Die untersuchten Systeme wurden in einem für die Verhältnisse der Informatik recht großem Zeitraum entwickelt. Daher spiegelt sich die allgemeine Entwicklung im Hard- und Softwarebereich natürlich auch in den verwendeten Rechnern und der benutzten Software wieder.

So implementierten die frühen Systeme ihren Inferenzmechanismus noch selbst in einer prozeduralen Programmiersprache (z. B. XPLANE in FORTRAN). Die eingesetzten KI-Methoden beschränkten sich auf die Benutzung von Produktionsregeln. Als Hardware kamen Großrechner (z. B. GARI auf einer HB-68) zum Einsatz.

[1] Eine Reihe weiterer Systeme die nicht in diesen Vergleich aufgenommen wurden, da über sie nur wenig Informationen vorhanden waren, oder die in [JF89] bereits beschrieben wurden werden ebenfalls in Anhang A tabellarisch vorgestellt.

	AVOGEN	CAMEX	CHAMP	KRUPP	FEXCAP	GARI	IUDICA	GENOA	GPPS	KAPLAN	ROUND	XPLANE
Entstehungszeitpunkt:												
Jahr der Veröffentlichung	90	87	88	89	89	81	88	90	89	89	86	88
Software:												
KI-Methoden wurden selbstimplementiert		×	×			×	×	×	×	?		×
Benutzung einer Expertensystemshell	×			×			×					
Hardware:												
Großrechner						×						×
Workstation	×[1]	×[2]	×[1]	×[3]			×[1]	×	×			
PC					×							

Legende:

KRUPP: Abkürzung für das Expertensystem für die spanende Bearbeitung.

IUDICA: Abkürzung für das generative Arbeitsplanungssystem mit CAD-Kopplung.

×: Das entsprechende System hat die genannte Fähigkeit

?: Es ist unbekannt, ob das entsprechende System die genannte Fähigkeit besitzt.

[1]: Rechner der VAX-Familie

[2]: Apollo-Workstation

[3]: Symbolics-Lispmaschine

Tabelle 3.1: Verwendete Hard- und Software

Spätere Systeme implementieren zwar den Produktionsregelinterpreter zwar auch noch selbst, verwandten dazu aber meist eine nichtprozedurale Programmiersprache (z. B. GPPS benutzt PROLOG, CAMEX verwendet LISP). Auch gekaufte Produktionsregelinterpreter kommen zum Einsatz (FEXCAPP mit SierraOPS5). Als Hardware setzten sich immer mehr Workstations durch (meistens VAX oder Apollo).

Neuere Systeme verwenden meistens gekaufte Expertensystemshells (z. B. AVOGEN KnowledgeCraft, das Expertensystem zur spanenden Bearbeitung KEE), oder implementieren selbst eine Shell (MMP im generativen Arbeitsplanungssystem mit CAD-Kopplung). Die Werkzeuge dieser Shells wie Frames, Regeln und Constraints kommen zum Einsatz. Es werden weiterhin primär Workstations eingesetzt, es kommen aber auch KI-Rechner zum Einsatz (KEE wird auf einer Symbolics benutzt).

3.2 Grundannahmen der Systeme

Zur Lösung der Planungsaufgabe machen die studierten Systeme unterschiedliche Grundannahmen. Diese haben starken Einfluß auf die Konzeption und Struktur des Systems, sie wirken sich vor allem auf die Zahl der generierten Pläne aus.

Die meisten Systeme gehen davon aus, daß es unmöglich bzw. zu teuer ist, alle möglichen Pläne zu erzeugen. So wird in diesen Systemen versucht, die Zahl der möglichen Pläne durch einschränkende Heuristiken zu beschränken. Mit Hilfe eventuell notwendig werdenden Backtrackings wird dann nach einer optimalen[2] Lösung gesucht.

Eine prinzipiell andere Annahme macht das System KAPLAN. Dort wird es für das Finden einer guten Lösung als unabdingbar erachtet, daß alle Pläne erzeugt werden. Die gefundenen Lösungen werden bewertet, und eine optimale ausgewählt.

[2] Optimal bedeutet in diesem Zusammenhang optimal im Bezug auf die jeweils gewählte Heuristik. Je nach Art der Bewertungsfunktion und des angewandten Inferenzmechanismus können dies lokale oder globale Optima sein.

Dies bedeutet einen stark erhöhten Rechenaufwand zur Findung aller Lösungen, der sich aber z. B. bei der Alternativplanung wieder auszahlt.

3.3 Bearbeitbares Teilespektrum

Die meisten untersuchten Systeme sind entweder zur Planung der Bearbeitung von prismatischen Teilen oder Drehteilen entworfen. Eine Ausnahme bildet hier GENOA, das zur Planung beliebiger Teile eingesetzt werden soll, der Schwerpunkt dürfte hier aber auch bei der Bearbeitung prismatischer Teile liegen[3].

	AVOGEN	CAMEX	CHAMP	KRUPP	FEXCAP	GARI	IUDICA	GENOA	GPPS	KAPLAN	ROUND	XPLANE
prismatische Teile		×			×	×	×	×				×
Drehteile	×		×	×		×		×	×	×	×	

Tabelle 3.2: Bearbeitbares Teilespektrum

Die Wahl der Teileklasse hat maßgeblichen Einfluß auf die Struktur der Systeme:

Prismatische Teile Systeme zur Behandlung prismatischer Teile sind fast ausschließlich featurebasiert. D. h. einfache Teile werden vollständig durch das Rohteil und eine Menge zu fertigender Features (wie Löcher, Bohrungen, Taschen, Schlitze etc.) beschrieben. So reduziert sich die Operationsplanung im wesentlichen darauf, aus den vorgegebenen Fertigungsmöglichkeiten eines Features die beste auszuwählen, und die Bearbeitungsreihenfolge der Features festzulegen. Dadurch wird aus der Planungsaufgabe mehr eine Konfigurationsaufgabe.

Drehteile Bei Drehteilen ist die Lage etwas anders. Dort werden die Werkstücke meist nicht durch eine Menge von Features beschrieben, sondern durch Angabe der Fertigteilkontur. Hier ist es Aufgabe des Systems, die zu fertigenden Bereiche festzulegen. Davon ausgehend können dann Fertigungsverfahren und Bearbeitungsreihenfolge bestimmt werden. Diese Teilaufgabe ist dann allerdings meist einfacher als bei prismatischen Teilen, da die Zahl der Kombinationsmöglichkeiten aufgrund der eher zweidimensionalen Problemstellung geringer ist.

So sind auch die Schwerpunkte der einzelnen Systeme entsprechend unterschiedlich gelagert. Reale Teile benötigen im allgemeinen aber eine Kombination der obigen Vorgehensweisen.

Prismatische Teile entstehen meistens nicht nur durch Bohren einiger Löcher in einer Platte, diese Platte muß zuerst in eine veränderte Form gebracht werden. Diese Form läßt sich aber nur schwer als ein Feature beschreiben, so daß eine Planung dieses Bearbeitungsprozeßes notwendig wäre. In dieser Richtung sind allerdings in den betrachteten Systemen bis jetzt nur Ansätze realisiert (z. B. in FEXCAPP). Bei komplexeren Teilen wird außerdem die Bestimmung der Aufspannung und der benötigten Vorrichtung viel komplizierter als es bei einfachen Teilen der Fall ist. Dies führte unter anderem zur Entwicklung eigenständiger Systeme zur Lösung dieser Aufgabe(z. B. IDA [Kra88]).

Reine Drehteile sind auch nur selten anzutreffen. Meistens enthalten sie zusätzliche Formelemente die sich leicht als Features beschreiben lassen (z. B. Nuten, Schlitze, Bohrungen). Die Bearbeitung der Features läßt sich dann aber wie bei den prismatischen Teilen planen. Im Gegensatz zu den prismatischen Teilen sind aber Systeme zur Behandlung einfacher rotationssymmetrischer Teile evtl. sinnvoll. Dies liegt daran, daß mit der Drehbank eine Maschine existiert, die nur in der Lage ist, solche Teile zu fertigen.

[3] Diese allgemeine Verwendbarkeit liegt jedoch weniger in GENOA, als in der generellen Verwendbarkeit des unterliegenden Entscheidungstabellensystems (im wesentlichen allerdings für Variantenplanung) begründet

Innerhalb einer Teileklasse ist das Spektrum der in ihr enthaltenen Teile aber immer noch so groß, daß nicht alle Systeme alle möglichen Teile bearbeiten können.

Prismatische Teile Die Systeme für prismatische Teile unterscheiden sich in ihrem Featurespektrum und der Fähigkeit auch weniger featureorientierte Operationen planen zu können. Die meisten dieser Systeme sind zur Bearbeitung von Guß- oder Stahlteilen gedacht, so daß dort die spanende Bearbeitung von Hauptinteresse ist. Eine Ausnahme bildet das generative Arbeitsplanungssystem mit CAD-Kopplung, das zur Bearbeitung von Blechteilen gedacht ist, und daher auch Schneideoperationen planen können muß.

Drehteile Bei den Systeme zur Bearbeitung von Drehteilen reicht das Spektrum von der reinen Außenbearbeitung von Teilen mit einseitig abfallender Kontur (Expertensystem zur spanenden Bearbeitung) über die Außen- und Innenbearbeitung bis zur Bearbeitung auch nichtrotationssymmetrischer Features (z. B. GPPS).

3.4 Bearbeitete Teilaufgaben

Unabhängig von ihrer Konzeption als Integriertes System oder als Stand-alone System variiert der Umfang der Systeme. Allerdings tendieren die integrierten Systeme dazu, eine größeren Umfang zu haben, da sie bestimmte Komponenten wiederverwenden oder mitbenutzen können.

Welche Teilaufgaben sollte nun ein System zur generativen Arbeitsplanung umfassen? In [JF89] werden folgende Aufgabenbereiche genannt, die wenigstens teilweise abgedeckt werden sollten:

- Zuordnung von Betriebsmitteln (Auswahl von Maschinen und Werkzeugen)
- Bestimmung von Prozeßparametern (Fertigungsverfahren, Maschineneinstellungen, Hilfsmittel)
- Planung und Optimierung der Arbeitsgangreihenfolge
- Alternativplanung
- Simultanplanung

Die hier betrachteten Systeme decken im wesentlichen die ersten drei Gesichtspunkte der obigen Aufzählung ab. Eine Alternativplanung findet nur in geringem Umfang statt (z. B. indem man bestimmte Maschinen fest vorgibt etc.). Simultanplanung, d. h. die Bestimmung gleichzeitig ausführbarer Operationen wird von keinem der untersuchten Systeme voll unterstützt. Ansätze finden sich in GARI und dem im Anhang vorgestellten System Hi-Mapp[4].

Weiterhin werden folgende Teilaufgaben von einigen Systemen bearbeitet:

- Generierung der Werkstückbeschreibung
- NC-Teileprogrammgenerierung

Da die Art der Generierung der Werkstückbeschreibung von großer Bedeutung für die Architektur der nachfolgenden Systemkomponenten ist, wird darauf in Abschnitt 3.7 genauer eingegangen.

Die NC-Teileprogrammgenerierung wird von einigen Systemen nicht bereitgestellt, da für sie der Bereich der Arbeitsplanung nach Festlegung der Operationen und Werkzeuge beendet ist. Dies ist insbesondere dann sinnvoll, wenn keine NC-geeigneten Maschinen eingesetzt werden. Oft wird diese Aufgabe aber auch in einer anderen Komponente eines Gesamtsystems ausgeführt.

[4] siehe Tabelle A.2

	AVOGEN	CAMEX	CHAMP	KRUPP	FEXCAP	GARI	IUDICA	GENOA	GPPS	KAPLAN	ROUND	XPLANE
Maschinenauswahl	×	?	×	×		×	×	×[1]	×			×[1]
Werkzeugauswahl	×	×	?	×	×		×	×[1]	×	×[1]	×	×
Wahl der Aufspannungen	×	?		×	×		×		×	×	×	
Planung und Optimierung der Arbeitsvorgangsfolge	×	×	×	×	×	×	×	×	×	×	×	×
NC-Teileprogrammgenerierung	?		×	×	×[1]				×[1]	×[1]		×
Werkstückdefinition	×[1]			×		×	×[2]		×[1]		×	
CAD-Kopplung	×		×	×	×			×	×	×		

Legende:

[1]: Komponente ist extern realisiert [2]: Erfragung fehlender Information [3]: Planung in externer Komponente

Tabelle 3.3: Realisierte Systemkomponenten

3.5 Integration in das betriebliche Umfeld

Bei den betrachteten Systemen können prinzipiell zwei Arten unterschieden werden:

1. Stand-alone Systeme
2. Systeme, die in eine Gesamtumgebung integriert sind.

Beide Arten haben eine Reihe von Vor- und Nachteilen:

Stand-alone Systeme Die Stand-alone Systeme arbeiten weitestgehend unabhängig von anderen Programmsystemen und müssen daher alle anfallenden Aufgaben selbst bewältigen. Dazu ist es notwendig, daß alle relevanten Daten innerhalb des Systemes selbst verfügbar sind. Dies bedingt eine redundante Datenhaltung, da Daten wie etwa verfügbare Maschinen und Werkzeuge auch für andere Teile der Fertigungsplanung relevant sind. Diese Redundanz kann dann leicht zu Fehlern der folgenden Art führen:

> Innerhalb der lokalen Datenhaltung des Arbeitsplanungssystems werden Maschinen ausgewählt, die momentan nicht benutzt werden können, da sie sich in Reparatur befinden. In der globalen Datenbasis, die erfahrungsgemäß von mehr Benutzern benötigt wird und daher besser gepflegt wird, sind die Maschinen schon als nicht benutzbar markiert.

Eine lokale Datenhaltung zwingt also das Arbeitsplanungssystem zu einem regelmäßigen Update der Wissensbasis. Erfahrungsgemäß wird diesem Gesichtspunktbeim der Systeme aber relativ wenig Aufmerksamkeit gewidmet. So müßte dieser Prozeß zumindest teilautomatisiert werden, was in den betrachteten Systemen nicht der Fall ist.

Andererseits kann die lokale Datenhaltung viel mächtigere Repräsentationsformalismen als eine hierarchische Datenbank anbieten, wodurch sich das Planungsproblem besser lösen läßt. In manchen Fällen wird sicherlich erst dadurch eine Lösungsfindung möglich.

Soll das System alle anfallenden Aufgaben selbst bewältigen, so ist es notwendig, auch Teilaufgaben wie die Berechnung der Vorgabezeiten und die Erzeugung des NC-Codes zu realisieren. Oft existiert aber schon konventionelle Software, die diese Aufgaben löst und auch sinnvollerweise benutzt werden sollte. Dazu müßte aber eine saubere Schnittstelle zu diesen Programmen aus dem Planungssystem heraus bestehen, etwa in

Form von RPC's[5]. Die wenigsten Systeme können diese anbieten, manche bieten die Möglichkeit über Schnittstellen Funktionen und Prozeduren anderer Programmiersprachen einzubinden, andere sind auf die Kommunikation mit Hilfe von Dateien angewiesen.

Werden die obigen Teilaufgaben innerhalb des Planungssystem implementiert, so tritt das Problem auf, daß die zugrundeliegenden Mechanismen wie Produktionsregeln zwar gut zur Lösung von Planungsproblemen geeignet sind, jedoch weniger gut für numerische oder prozedurale Fragestellungen. Dadurch ergibt sich eine unnatürliche und ineffiziente Lösung des Teilproblems.

Einige der Systeme (z. B. das generative Arbeitsplanungssystem mit CAD-Kopplung) bieten daher auch die Möglichkeit, Prozeduren als Aktionen von Regeln zu definieren.

Integrierte Systeme Als integrierte Systeme werden hier System betrachtet, die eine mehr oder weniger lose Kopplung mit anderen Programmen besitzen und diese zur Lösung der Planungsaufgabe mitbenutzen.

Bei den analysierten Systemen wurden zwei Arten der Integration angetroffen:

- Erweiterung eines bestehenden Systems zur Arbeitsplanung
- Entwurf des Arbeitsplanungssystems als eine Komponente eines umfangreicheren Systems (z. B. zur Produktionsplanung einer flexiblen Fertigungszelle).

Gemeinsam ist allen integrierten Systemen, das sie Teilaufgaben an andere Komponenten des Gesamtsystems delegieren. Dies kann z. B. die Teilebeschreibung sein, die in einem CAD-System erfolgt, oder die Berechnung der Vorgabezeiten in einem Modul des Entscheidungstabellensystems. Die beiden Arten der Integration unterscheiden sich in der Enge der Kopplung der Komponenten.

Systeme, die als Erweiterung eines bestehenden Systems entstanden, sind auf dieses zugeschnitten, das Basissystem kann nicht gegen ein gleichartiges ausgetauscht werden. Systeme der zweiten Art sind flexibler, der Austausch von Komponenten (z. B. dem CAD-System) ist möglich, solange diese eine gewisse Funktionalität aufweisen.

Durch die Mitbenutzung von Komponenten anderer Programme kann zwar der Umfang des Planungssystems reduziert werden, es müssen aber Kompromisse eingegangen werden. Dies betrifft besonders die Mächtigkeit der einsetzbaren Repräsentationsformalismen, da z. B. eine hierarchische Datenbank bei weitem nicht die Möglichkeiten einer framebasierten Repräsentation bietet.

	AVOGEN	CAMEX	CHAMP	KRUPP	FEXCAP	GARI	IUDICA	GENOA	GPPS	KAPLAN	ROUND	XPLANE
Stand-alone System		×	×	×	×	×	×				×	
Erweiterung eines bestehenden Systems	×							×				
Komponente eines Gesamtsystems									×	×		

Tabelle 3.4: Systemumgebungen

3.6 Integration anderer Arbeitsplanungstechniken

In der Arbeitsplanung kommen auch Planungstechniken wie Variantenplanung, Änderungsplanung und Wiederholplanung zum Einsatz. In Bereichen, in denen diese Planungsarten sinnvoll benutzt werden können,

[5] RPC - Remote Procedure Call

ist ihnen aufgrund der geringeren Komplexität Vorrang vor der generativen Planung einzuräumen. Daher erscheint es sinnvoll, die verschiedenen Planungstechniken zu kombinieren.

Bei den untersuchten Systemen ist allerdings eine solche Kombination selten, und nur, wenn überhaupt, bei den integrierten Systemen vorhanden.

Entstanden die Systeme als Erweiterung eines bereits bestehenden Planungssystems (AVOGEN, GENOA), so können die dort realisierten Planungstechniken benutzt werden. So bietet es sich an, Komponenten wie Ähnlichteilsuche, die Entscheidungstabellen oder das konventionelle System bereitstellt, zur Variantenplanung einzusetzen. Das wissensbasierte System deckt den Bereich der generativen Planung ab.

Bei Systemen dieser Art ist die Kopplung der verschiedenen Planungstechniken lose, der Rückgriff auf die bestehenden Komponenten erfolgt nicht automatisch, er muß erzwungen werden.

Enger ist die Kopplung bei Systemen, die schon beim Entwurf des Gesamtsystems die Integration anderer Arbeitsplanungstechniken berücksichtigten (z. B. GPPS).Sie benutzen nur dann die generative Arbeitsplanung, wenn keine verwendbaren Varianten zur Verfügung stehen. Es besteht aber auch hier keine Rückkopplung zwischen generativer und Variantenplanung.

3.7 Repräsentation des Werkstücks

Zur Lösung des Planungsproblems muß eine Beschreibung des Roh- und des Fertigteils in einer für den Planungsmechanismus geeigneten Form vorliegen.

Dazu genügen die geometrischen Daten einer CAD-Zeichnung nicht, es werden auch technologische Daten (z. B. Material, Toleranzen etc.) benötigt. Außerdem sind für den Planungsprozeß oft nicht geometrische Entitäten wie z. B. Linie von A nach B, Kreis um C mit Radius D wichtig; das Werkstück wird vielmehr durch fertigungstechnische und funktionale Einheiten, sogenannte *Features*, beschrieben.

Für manche Teilaufgaben ist eine abstraktere Teilebeschreibung besser geeignet, die ganze Fülle an Details ist zu ihrer Lösung nur hinderlich. Eine Teilebeschreibung auf mehreren Abstraktionsebenen bieten die analysierten Systeme aber nicht, einzig GPPS beschreibt das Teil auf der Topologieebene.

Auch wenn die Systeme alle eine mehr- oder weniger featureorientierte Werkstückbeschreibung benutzen, so ist die Art der Erzeugung und die Art der verwendeten Features doch unterschiedlich.

3.7.1 Erzeugung der Werkstückbeschreibung

Bei der Erzeugung der Werkstückbeschreibung können drei Vorgehensweisen unterschieden werden:

1. Interaktive Eingabe der Werkstückbeschreibung innerhalb des Planungssystems
2. Extraktion der Features aus den Daten einer CAD-Zeichnung
3. Produktionsorientierte Konstruktion mit modifizierten CAD-Systemen

Interaktive Eingabe der Werkstückbeschreibung Bei einigen Systemen (insbesondere den Standalone Systemen) ist es nicht möglich, Daten aus CAD-Systemen zu übernehmen. Dort erfolgt die Werkstückbeschreibung durch die Auswahl und Parametrisierung von vordefinierten Elementen innerhalb des Systems. Dies hat den Vorteil, daß die Beschreibung gleich in dem vom Planungssystem gewünschten Format vorliegen.

Allerdings bietet diese Art der Erzeugung auch eine Reihe von Nachteilen:

1. Die Art der Konstruktion von Teilen ist für den Benutzer unbekannt und stößt auf Ablehnung[JF89].
2. Konstruktionen, die mit einem CAD-System erstellt wurden, können nicht wiederverwendet werden.

	AVOGEN	CAMEX	CHAMP	KRUPP	FEXCAP	GARI	IUDICA	GENOA	GPPS	KAPLAN	ROUND	XPLANE
Featureerkennung		×	×		×		×					×
Interaktive Definition				×		×					×	
produktionsorientierte Konstuktion	×							×	×	×		

Tabelle 3.5: Erzeugung des Werkstücks

3. Es bestehen in den untersuchten Systemen dieser Art nur geringe Visualisierungsfähigkeiten[6]

4. Es wird kein standardisiertes Format für die Daten verwendet. Dadurch wird der Austausch von Daten mit anderen Stellen unmöglich.

Featureerkennung aus CAD-Daten Andere Systeme versuchen die featureorientierte Beschreibung aus den Daten einer CAD-Zeichnug zu gewinnen. Dazu werden Algorithmen benutzt, die in den linienorientierten CAD-Zeichnungen Features (z. B. Löcher und Bohrungen) erkennen sollen.

Diese Aufgabenstellung ist aber selbst so komplex, daß sie bisher nur unzureichend gelöst wurde. Meistens werden dazu ebenfalls wissensbasierte Methoden verwandt, die dann einen Großteil des Gesamtsystems in Anspruch nehmen[7]. Es gelingt auch nicht, alle Features zu erkennen, und Fehlerkennungen sind an der Tagesordnung. Dadurch wird eine Nacheditierung der erkannten Features notwendig.

Um die Featureerkennung effizient und sicher zu realisieren, ist es notwendig, sie auf bestimmte Modellierer (oder Modellierertypen) zuzuschneiden[8].

Verstärkt wird dieser Effekt durch die fehlende Standardisierung von Graphikschnittstellen, was bei dem Expertensystem zur spanenden Bearbeitung dazu führte, daß auf eine Featureerkennung verzichtet wurde, und man das Werkstück interaktiv beschreibt.

Hinzu kommt, daß mit Hilfe der Featureerkennung eigentlich Wissen rekonstruiert wird, das zum Zeitpunkt der Konstrukton schon vorhanden war, und daher Arbeit doppelt getan wird.

Produktionsorientierte Konstruktion Die in den letzten beiden Abschnitten angeführten Probleme stießen die Entwicklung von sogenannten „technologischen Modellierern“[9] an [HPS89, HPS90]. Die Werkstücke werden mit Hilfe von technologischen Primitiven konstruiert, die Art der Konstruktion entspricht der Vorgehensweise im vorletzten Abschnitt. Darüberhinaus stellen sie aber die Visualisierungsmöglichkeiten eines geometrischen Modellierers zur Verfügung.

Die meisten dieser Systeme befinden sich noch in der Entwicklung, das mit ihnen modellierbare Teilespektrum ist eingeschränkt.

Weiterhin sind Bemühungen in Gange gekommen, ein standardisiertes Format für die produktionsrelevanten Daten eines Werkstücks zu definieren. Die entsprechenden Empfehlungen werden derzeit im Rahmen von STEP (**ST**andard for the **E**xchange of **P**roduct model data) erarbeitet. Erste Analysen [Ble90] zeigen jedoch, daß diese noch verbesserungsfähig sind, da u. a. noch inkonsistente Teilebeschreibungen möglich sind.

[6] Ansprechende Visualisierung eines Teiles (z. B. 3-D Darstellung, Hidden-Line und Hidden-Surface, Rendering) würden die Neuimplementierung eines geometrischen Modellierers gleichkommen. Dies wäre mit einem Aufwand verbunden, der von den Entwicklern der Systeme nicht zu leisten ist.

[7] So sind ungefähr 75% der Regeln in FEXCAPP Featureerkennungsregeln.

[8] FEXCAPP benötigt für die Featureerkennung z. B. eine BREP-Darstellung

[9] Andere in diesem Zusammenhang benutzte Begriffe sind „produktionsorientierte Modellierer“ und „feature based modeling“.

3.7.2 Verwendete Featurearten

Die untersuchten Systeme haben unterschiedliche Vorstellungen, was ein Feature ist. Dies hängt einerseits mit den unterschiedlichen Teilespektren zusammen, andererseits differerieren auch die Teilebeschreibungen von Systemen mit ähnlicher Aufgabenstellung. Bestimmte Aspekte des Werkstücks lassen sich relativ leicht durch Features beschreiben, hier benutzten die Systeme auch im wesentlichen die gleichen Featuretypen. Andere Aspekte lassen sich schwerer durch Featurebeschreibungen erfassen, dort sind die Unterschiede zwischen den Systemen größer.

Die Eigenschaft, leicht oder schwer durch Features beschreibbar zu sein, ist nun interessanterweise eng daran gekoppelt, ob der betrachtete Aspekt des Werkstücks „ materiallos“ oder „materialhaltig“ ist.

Materiallose Features Relativ einfach fällt eine Featureklassifikation materialloser Komponenten[10] des Teiles. Dies sind bei prismatischen Teilen z. B. Bohrlöcher, Schlitze und Taschen; bei Drehteilen sind es die nichtrotationssymmetrischen Komponenten wie Paßfedernuten.

Für solche Features läßt sich leicht ein Skelettplan angeben, da Art und Reihenfolge der Arbeitsvorgänge relativ fest sind.

Materialhaltige Features Schwieriger ist die Definition von materialhaltigen Features[11] zur Beschreibung der Hauptkontur des Werkstücks. Dort sind komplexere Geometrien (z. B. Splines, Freiformflächen) möglich, die einen größere Zahl von Fertigungsmöglichkeiten zuläßt.

Relativ leicht fällt die Klassifikation noch bei Drehteilen, da die Kontur in zwei Dimensionen beschreibbar ist. Es werden in den meisten Fällen Featuretypen wie Zylinder , Konus etc. definiert. Aber auch hier bestehen Unterschiede in Bezug auf die Granularität der Features. Benutzt AVOGEN Features wie Zylinder, Konus und Vierkant, so werden in GPPS etwa Gewindeende und Flansch als Feature angesehen[12].

Bei prismatischen Teilen mit dreidimensionaler Beschreibung ist eine vollständige Auflösung in Features noch nicht gelungen. Daher beschränken sich die betrachteten Systeme in irgendeiner Weise. CAMEX betrachtet nur Teile die in 2 1/2 Dimensionen beschreibbar sind, andere Systeme beschränken sich auf die Bearbeitung materialloser Features in einem festgelegten Grundkörper.

3.8 Probleme einer vollständigen Featurebeschreibung

Abgesehen von den Problemen, die bei der Erzeugung einer vollständigen Featurebeschreibung entstehen, stellt sich die Frage, ob eine solche auch immer sinnvoll ist. Eine featureorientierte Beschreibung erleichtert einerseits die Erstellung eines Arbeitsplanes, andererseits werden durch die Festlegung der Arbeitsschritte innerhalb des Skelettplanes des Features die Freiheitsgrade bei der Planung evtl. zu sehr eingeschränkt.

Bei den betrachteten Systemen erfolgt diese Festlegung in unterschiedlichen Graden. In GARI wird jedes Feature generell mit einem Schrupp- und einem Schlichtschritt bearbeitet. Andere Systeme lassen zwar die Anzahl der Operationen für ein Feature fest, sie kann aber von Feature zu Feature unterschiedlich sein. Bei dritten ist die Zahl der Operationen pro Feature völlig frei.

Durch die Kopplung der Operationen an die Features werden aber eventuell optimale Arbeitspläne nicht gefunden, wie das folgende Beispiel zeigt.

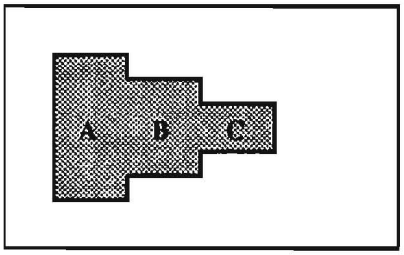
Beispiel 3.1 *Angenommen, als Feature zur Beschreibung eines rotationssymmetrischen Teiles stehe der Zylinder zur Verfügung. Es sei weiter festgelegt, daß zur Bearbeitung eines Zylinders ein Schrupp- und ein Schlichtschritt notwendig sind, und der Schrupp- vor dem Schlichtschritt erfolgt.*

[10] In den untersuchten Systemen werden hierfür auch die Begriffe „Nebenformelement“ und „sekundäres Feature“ benutzt

[11] Auch „Hauptformelemente“ oder „primäre Features“

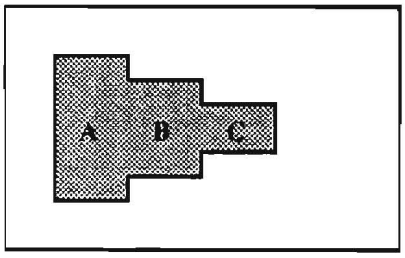
[12] Dies beruht auf der Tatsache, daß in GPPS eine Kopplung zu Gruppentechnologieverfahren besteht, und die Features an die dort benutzten Komponenten angelehnt sind. Für eine Übersicht übder die benutzten Features sei auf die in GPPS definierte MGF (Matrix of Geometric Features) verwiesen.

Dann würde für ein einfaches Teil der folgende Plan generiert:



1. Schruppen Zylinder C
2. Schlichten Zylinder C
3. Schruppen Zylinder B
4. Schlichten Zylinder B
5. Schruppen Zylinder A
6. Schlichten Zylinder A

Ein besserer Plan wird sicher durch einmaliges Schruppen aller Zylinder und darauffolgendes Schlichten erreicht.



1. Schruppen Zylinder A, B und C
2. Schlichten Zylinder A, B und C

Es sind also featureübergreifende Operationen für die Erzeugung eines optimalen Arbeitsplanes notwendig.

Die im Beispiel dargestellte Situation ließe sich sicher durch Definition eines komplexeren Featuretyps „Treppe" beheben, die prinzipielle Problematik bliebe davon allerdings unberührt, da sie nun auf einer höheren Ebene auftreten würde.

3.9 Integration von Datenbanken

3.9.1 Notwendigkeit der Integration

Die Arbeitsplanung steht innerhalb des Betriebsablaufes nicht allein, sie ist vielmehr eng mit Komponenten wie Konstruktion, Fertigung, Fertigungsplanung, Produktionsplanung und Angebotserstellung gekoppelt. Es wird daher genauso auf Daten dieser Bereiche zugegriffen, wie diese Daten aus der Arbeitsplanung benötigen[Hä89].

Deshalb müssen Schnittstellen zur Realisierung dieses Datentransfers innerhalb des Arbeitsplanungssystems bereitgestellt werden. Dies bedeutet, daß irgendeine Art von Datenbankzugriff möglich sein muß.

Die Notwendigkeit der Integration von Datenbanken wird noch durch die Tatsache verstärkt, daß der Umfang der benötigten und erzeugten Daten ein gleichzeitiges Halten aller Daten im Hauptspeicher unmöglich macht, und eine Auslagerung von Daten notwendig wird.

3.9.2 Arten der Integration

Es lassen sich drei Integrationsgrade bei den untersuchten Systemen unterscheiden:

1. Es ist keine Schnittstelle zu Datenbanken vorhanden
2. Die Datenbank dient als Datenpuffer zwischen den einzelnen Komponenten des Systems. Innerhalb der Komponenten erfolgen nur Zugriffe auf lokale Daten.
3. Während des Systemlaufes wird dauernd auf Daten der Datenbank zugegriffen, sie ist integraler Bestandteil des Systems.

	AVOGEN	CAMEX	CHAMP	KRUPP	FEXCAP	GARI	IUDICA	GENOA	GPPS	KAPLAN	ROUND	XPLANE
keine Integration		x	x	x		x	x					
Datenbank als Puffer	x							x		x		
Datenbank als integraler Bestandteil					x				x		x	x

Tabelle 3.6: Datenbankintegration

3.9.3 Anwendungsgebiete

Ist eine Integration von Datenbanken vorhanden, so eignen sich einige Aspekte der Problemstellung besser als andere zur Unterstützung durch Datenbanken. Hier sollen die zwei wichtigsten kurz vorgestellt werden.

Werkstattmodell Häufig wird das Werkstattmodell durch Zugriffe auf Datenbanken realisiert. Dies hat seine Ursache darin, daß Daten wie verfügbare Maschinen und Werkzeuge auch für die Fertigungssteuerung benötigt werden, und daher schon in Datenbanken gespeichert sind.

Schnittdatenbestimmung Erfahrungswerte bei der Zerspanung eignen sich besonders zur Ablage in Datenbanken. Dort können dann die optimalen Schnittparameter in Abhängigkeit vom Material des Werkstücks und des Werkzeugs erfragt werden.

Diese Werte werden oft innerhalb einer Firma bestimmt, in vielen Ländern existieren sogar zentrale Datenbanken die Schnittparameter bereitstellen. In Deutschland sind dies z. B. die Datenbank INFOS an der TH Aachen und der Schnittwertspeicher Magdeburg [SK83].

3.9.4 Probleme der Integration von Datenbanken

Das zentrale Problem bei der Integration von Datenbanken ist die unterschiedliche Mächtigkeit der Beschreibungssprachen. Sie ist zwar bei relationalen Datenbanken größer als bei hierarchischen, aber weitaus niedriger als bei Frames oder Objekthierarchien.

Daher wurden in letzter Zeit große Anstrengungen unternommen, um diese Lücke zu schließen. Dies geschah durch die Entwicklung von Objektorientierten- und Nonstandarddatenbanken. Die meisten Systeme sind noch Forschungssysteme, z. B. PRIMA, das an der Universität Kaiserslautern entwickelt wird. Es existieren bis jetzt nur wenige Produkte, wie etwa STATICE von Symbolics oder G-BASE von Graphael.

Solche Datenbanken erlauben eine Darstellung der Daten in ähnlicher Weise wie mit Frames oder Objekten. Im Falle von STATICE ist der Zugriff auf die Datenbank sogar völlig transparent.

3.10 Verwendete KI-Methoden

In den nachfolgenden Abschnitten sollen nun die in den verschiedenen Systemen eingesetzten KI-Methoden etwas genauer beleuchtet werden.

3.10.1 Wissensrepräsentationsformalismen

Praktisch jedes System bietet Produktionsregeln zur Repräsentation des Planungswissens an. Dabei kommen sowohl Produktionsregelinterpretern mit Forward- als auch mit Backward-chaining zum Einsatz (z. B. durch Benutzung von PROLOG oder OPS5).

Den in der Literatur gemachten Angaben ist allerdings in diesem Punkt mit etwas Skepsis zu begegnen, da dort Zitate wie etwa „Rückwärtsverkettung (backtracking)“ und „Realisierung des backward-reasoning mit forward-chaining in der Programmiersprache PROLOG“ vorkommen

	AVOGEN	CAMEX	CHAMP	KRUPP	FEXCAP	GARI	IUDICA	GENOA	GPPS	KAPLAN	ROUND	XPLANE
Produktionsregeln	×	×	×	×	×	×	×		×	×	?	×
Frames	×			×			×					
Constraints	?			×			×					

Tabelle 3.7: Wissensrepräsentationsformalismen

Erst neuere Systeme benutzen Formalismen wie Frames bzw. Objekte oder Constrain s. Frames dienen zur Repräsentation der vorhandenen Betriebsmittel wie Maschinen und Werkzeugen. Constraints kontrollieren die Operationsreihenfolge[13].

Regeln Die meisten Systeme benutzen „klassische“ Produktionsregeln mit Bedingungs- und Aktionsteil. In GARI wurden diese um Gewichte und getypte Variablen erweitert, FEXCAPP erlaubt die Eingabe der Regeln in „strukturiertem“ Englisch. In XPLANE werden die Regeln nach Fortran compiliert.

Die Bildung modularisierter Regelbasen ist in vielen Systemen möglich. Der Wechsel zwischen den Modulen wird durch ausgezeichnete Aktionen ausgeführt.

In einigen Systemen können Metaregeln zur Steuerung des Kontrollflußes definiert werden.

Constraints In den neueren Systemen kommen Constraints bei der Reihenfolgebestimmung zum Einsatz. Dazu werden entweder entsprechende Komponenten bestehender Shells benutzt, oder selbst Algorithmen zur Constraint-Propagierung implementiert.

Inwieweit andere Teilaspekte der Problemstellung mit Constraints modelliert wurden, ist nicht bekannt. Zumindest die Werkzeug- und Maschinenauswahl würden sich jedoch dafür anbieten.

In früheren Systemen wurden die jetzt mit Constraints gelösten Probleme durch Regeln realisiert. Hier fehlt jedoch die automatische Propagierung und die damit verbundenen Rückkopplungen innerhalb des Constraint-Netzes, die den Constraint-Ansatz gerade erst interessant macht.

Frames Frames haben in der letzten Zeit wegen der Vererbung und der Möglichkeit, über Dämonen aktive Strukturen zu definieren, die Entwicklung komplexer Systeme vereinfacht[14].

So haben auch in den betrachteten Systemen Frames die „flachen“ Darstellungen der Wissensbasis verdrängt. Ob jedoch immer die volle Funktionalität von Frames oder „nur“ Objekthierarchien bereitgestellt werden, ist leider nicht bekannt.

[13] Statt Constraints wurden in den älteren Systemen Produktionsregeln benutzt, die den Suchraum einschränkten. Die Anwendung dieser Regeln wird in [JF89] ebenfalls als constraint-propagation bezeichnet.

[14] Nicht umsonst findet die "Objektorientierung" auch in prozeduralen Programmiersprachen immer mehr Beachtung.
Von besonderem Interesse im Zusammenhang mit der hier betrachteten Problemstellung sind Objektorientierten Datenbanken (OODB) und objektorientierten CAD-Systeme

3.10.2 Planungstechniken

In den untersuchten Systemen kamen eine Reihe von Planungstechniken zum Einsatz, die in gewisser Weise die Entwicklung von Planungsexpertensystemen in den letzten 10 Jahren widerspiegeln. Die wichtigsten davon sind[15] :

Graphensuche Die meisten Systeme verwenden eine mehr oder weniger informierte Graphensuche auf dem durch das Planungsproblem aufgespannten Suchraum, als primären Planungsmechanismus. Von einem aktuellen Zustand wird mit Produktionsregeln eine Menge von Nachfolgezuständen erzeugt oder die Menge der Nachfolgezustände eingeschränkt. Typische Aktionen solcher Regeln sind die Veränderung des Werkstückzustandes entsprechend der Anwendung einer Operation, die Bestimmung von Reihenfolgerestriktionen auf den Operationen, die Einschränkung der Menge der benutzbaren Werkzeuge für eine Operation etc.

Bei einigen Systemen erfolgt die Suche uninformiert, andere benutzen Bewertungsfunktionen um die Suche zu steuern. Läuft das System bei der Planung in eine Sackgasse, wird mittels Backtracking zu einem früheren Zustand zurückgekehrt und dort weitergeplant. Die verwendeten Backtrackingmechanismen und Bewertungsfunktionen werden in den Abschnitten 3.10.4 und 3.10.5 vorgestellt, daher soll an dieser Stelle eine Detaillierung unterbleiben.

Hierarchische Planung Einige der Systeme verwenden Hierarchische Planung zur Strukturierung des Planungsproblem. Verwandte Ebenen der Planung sind dort zum Beispiel die Ebene der Maschinen, der Aufspannungen, der einzelnen Features etc. Innerhalb einer Einheit der jeweiligen Ebene wird dann von den notwendigen Aktionen innerhalb dieser Einheit abstrahiert, um zuerst eine Lösung des Problems in dieser Ebene zu finden.

Die Realisierung der Hierarchie erfolgt auf unterschiedliche Weise. Einige Systeme verwenden Metaregeln, andere sogenannte „Szenarios", dritte erreichen eine Art der hierarchischen Planung durch Regeln die Modulwechsel ausführen, bei manchen ist die Hierarchie der Planung festcodiert.

Weitere Planungstechniken Als weitere Planungstechniken kommen nichtlineare Planung und Skelettplanung zum Einsatz. Nichtlineare Planung wird vom System GARI zur Planung simultan ausführbarer Aktionen benutzt, das generative Arbeitsplanungssystem mit CAD-Kopplung benutzt Skelettplanung.

	AVOGEN	CAMEX	CHAMP	KRUPP	FEXCAP	GARI	IUDICA	GENOA	GPPS	KAPLAN	ROUND	XPLANE
Graphensuche	×	×	×	×	×	×	×	×	×		×	×
Hierarchische Planung	×			×		×	×		×		×	
Nichtlineares Planen						×						
Skelettplanung							×					

Tabelle 3.8: Verwendete Planungstechniken

[15] in [JF89] werden die Systeme ebenfalls auf die benutzten Planungstechniken untersucht, dort erfolgt unserer Meinung nach aber eine Vermischung der Planungtechniken mit ihren Implementierungsaspekten. Daher werden hier nur eine Teilmenge der dort genannten Begriffe in diesem Zusammenhang verwandt.

3.10.3 Planungsrichtung

Sowohl Forward- als auch Backward-Reasoning kommen bei den analysierten Systemen zum Einsatz. Forward-Reasoning bedeutet in diesem Zusammenhang , das vom Roh- zum Fertigteil geplant wird, beim Backward-Reasoning erfolgt die Planung vom Fertig- zum Rohteil.

Für die Verwendung der jeweiligen Planungsrichtung werden von den Autoren die folgenden Vorteile angeführt:

Forward-Reasoning

- Forward-Reasoning entspricht der Vorgehensweise des Arbeitsplaners [JF89].

Backward-Reasoning

- Backward-Reasoning bietet Vorteile bei der Implementation, da davon ausgegangen wird, daß die schwierigen und teuren Operationen am Ende des Arbeitsplanes liegen. Durch die Rückwärtsplanung wird eine frühe Fokussierung auf die wichtigen Entscheidungen erreicht [ASP90].
- Backward-Reasoning ist der Logik und der Vorgehensweise des Technologen am ähnlichsten [PS89].
- Die Flächen, die das Werkzeug führen, sind bei der Rückwärtsplanung bekannt. Dadurch wird die Erkennung des nächsten zu bearbeitenden Features bei einfachen Rohteilen und komplexen Fertigteilen einfacher [JF89].

	AVOGEN	CAMEX	CHAMP	KRUPP	FEXCAP	GARI	IUDICA	GENOA	GPPS	KAPLAN	ROUND	XPLANE
Forward-reasoning		×	×	×	×	×	×	×		×	×	×
Backward-reasoning	×								×			

Tabelle 3.9: Benutzte Planungsrichtung

3.10.4 Backtrackingmechanismen

Die meisten Systeme benutzen chronologisches Backtracking zur Lösung eines Konfliktes. Aber auch dependency-directed Backtracking kommt in einigen Systemen zum Einsatz.

Bei den Systemen mit dependency-directed Backtracking nimmt das System GARI eine Sonderstellung ein. Nachdem es schon zu einem sehr frühen Zeitpunkt (1981) mit einer komplizierten Form der Abhängigkeitsverwaltung experimentiert hatte, wurde dies wegen seines Aufwandes verworfen und eine einfachere Lösung benutzt. Im Nachfolgeprojekt XPS-E (siehe Tabelle A.2) wurde dies aber zumindest teilweise wieder rückgängig gemacht[16].

Das Expertensystem zur spanenden Bearbeitung und das generative Arbeitsplanungssustem mit CAD-Kopplung setzen beide ein ATMS ein. Im ersten System wird dabei das in KEE bereitgestellte benutzt, während im zweiten ein eigenes ATMS implementiert wurde.

Leider wird in beiden Systemen nicht erwähnt, wie das ATMS genau eingesetzt wird, d. h. welche Inferenzen als Assumptions behandelt werden etc.

[16] Dies wurde wahrscheinlich durch den Fortschritt in der Hard- und Softwareentwicklung möglich

	AVOGEN	CAMEX	CHAMP	KRUPP	FEXCAP	GARI	IUDICA	GENOA	GPPS	KAPLAN	ROUND	XPLANE
Chronologisches Backtracking	×	×	×		×			×	×		×	×
Dependency-directed Backtracking						×						
ATMS				×			×					

Tabelle 3.10: Verwendete Backtrackingmechanismen

3.10.5 Bewertungsfunktionen

Einige der untersuchten Systeme steuern die Graphensuche mit Hilfe von Bewertungsfunktionen, andere Systeme bewerten die generierten Lösungen um aus ihnen eine optimale auszuwählen. Dabei ist bei manchen Systemen die Bewertungsfunktion fest, andere lassen Variationen der Funktion zu.

Werden die Bewertungsfunktionen bei der Graphensuche eingesetzt, so dienen sie der Auswahl des nächsten zu expandierenden Zustands. Die hierzu benutzten Funktionen haben eine sehr unterschiedliche Komplexität, sie reichen von einer 8-wertigen Bewertungstabelle in FEXCAPP bis zu einer kaum zu verstehenden Formel in ROUND.

Betrachtet man die Bewertungsfunktionen, so ist anzuzweifeln, ob sie immer zu einer optimalen Lösung führen (z. B. ob sie für das A*-Verfahren immer eine optimistischen Schätzung liefern). Teilweise scheinen die Wirkungsweise und die Grenzen solcher Bewertungen nicht ganz verstanden zu sein, wenn z. B in FEXCAPP behauptet wird, daß mit Hilfe einer 8-wertigen Funktion und Hillclimbing eine optimale Operationsreihenfolge bestimmt werden kann.

Die Bewertungsfunktionen zur Auswahl der besten Lösung greifen auf während der Lösungsfindung gesammelte Informationen zurück. In GARI ist dies die Summe der Gewichte der Regeln, die zu dieser Lösung führten, bei KAPLAN die Summe von sogenannten Scores, die für diese Aufspannung ermittelt werden.

3.10.6 Einschätzung der eigenen Kompetenz

Wie viele KI-Systeme haben auch die hier analysierten kaum die Möglichkeit zu erkennen, ob sie kompetent genug sind, das gestellte Problem zu lösen. Dies bedeutet, das auch bei Teilen, die mit den vorhandenen Betriebsmitteln nicht gefertigt werden können, versucht wird eine Lösung zu finden. Erst nachdem alle Möglichkeiten versucht worden sind, wird mit Mißerfolg abgebrochen.

Einen Ansatz zur Erkennung der eigenen Kompetenz bietet allein das System KAPLAN. Dort werden nichtrotationssymmetrische Teile des Werkstücks mit einer zylindrischen Hülle versehen, die das Werkstück auf einer Drehbank fertigbar macht.

3.10.7 Erklärungen

Erklärungen können nur in manchen Systemen gegeben werden. Sie beschränken sich auf die Wiedergabe des Regeltraces, um How- und Why-Fragen zu beantworten.

Dazu werden entweder eigene Mechanismen implementiert, wie in XPLANE, oder die zur Verfügung gestellten Mechanismen einer Shell genutzt, wie im Expertensystem zur spanenden Bearbeitung.

3.10.8 Unschärfe

Unschärfe, obwohl sie bei der gegebenen Problemstellungan vielen Stellen auftritt, wird nur von zwei Systemen behandelt.

In GARI soll das Gewicht der Regeln dazu dienen, Unschärfe zu modellieren; GPPS stellt Intervalle und Operationen auf diesen Intervallen zur Verfügung. Damit sollen Toleranzen besser behandelt werden können, ein Einsatz für andere Arten von Unschärfe ist aber auch möglich.

3.11 Zusammenfassung

Bei den hier untersuchten Systemen ist eine starke Streuung der Anwendungsgebiete, Problemstellungen und Realisationen zu verzeichnen, so daß kein einheitliches Bild über das Problemfeld „Arbeitsplanung" gezeichnet werden kann.

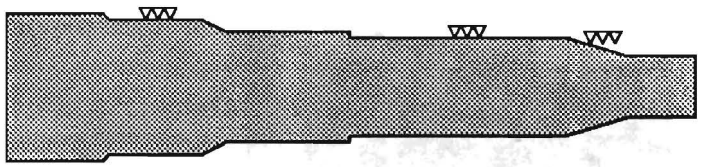
So ist die Bandbreite sowohl der verwendeten Hard- und Software als auch der bearbeiteten Problemstellungen enorm, sie reicht vom PC bis zum Großrechner und von der Bearbeitung von Drehteilen mit einseitig abfallender Kontur bis zum System zur Bearbeitung von prismatischen Teilen. Auch werden längst nicht alle Teilaufgaben von allen Systemen gleich stark behandelt.

All dies zeigt uns, daß es für die Entwicklung und den Entwurf eines „eigenen" Systems von großer Wichtigleit ist, die Problemstellung genau zu spezifizieren und nicht etwa ein „System zur Arbeitsplanung" zu entwerfen.

Zu beachten ist bei einem solchen Entwurf, daß das resultierende System nur dann akzeptiert wird, wenn es in das betriebliche (CIM) Umfeld eingebettet ist. Dieser Aspekt wird wegen seiner hohen Relevanz und relativen Unabhängigkeit von der Implementierung des Restsystems im verbleibenden Teil dieser Diplomarbeit eine große Rolle spielen, Kapitel 5 ist ihm vollständig gewidmet.

Bei der Werkstückbeschreibung und Erzeugung derselben sind zwei Tendenzen zu erkennen. Einerseits die Erzeugung der Beschreibung im Arbeitsplanungssystem selbst oder mit Hilfe eines speziellen Modellierers, so daß sie gleich im richtigen Format vorliegt. Andererseits die Interpretation von Beschreibungen die mit herkömmlichen CAD-Programmen erzeugt wurden.

Die verwendeten KI-Methoden hängen im wesentlichen vom Entstehungszeitpunkt des jeweiligen Arbeitsplanungssystems ab. Generell ist hier jedoch zu erkennen, daß die zu diesem Zeitpunkt gängigen Methoden benutzt wurden. Neuere Systeme verwenden Objekthierarchien zur Repräsentation der Domäne, Constraints zur Einschränkung des Suchraums und Regeln zur Inferenz.



4 Konzept eines Systems zur Arbeitsplanerstellung für Drehteile

Im vorherigen Kapitel wurde eine Reihe von Systemen zur Arbeitsplanung vorgestellt und bewertet. Dabei wurden einige Mängel festgestellt.

In diesem Kapitel soll nun das Konzept von TUPPSY[1], einem Systems zur Arbeitsplanerstellung für Drehteile vorgestellt werden, das versucht einige der entdeckten Mängel zu beheben.

Dazu wird im ersten Abschnitt kurz die Problemstellung konkretisiert und formalisiert. Dem schließt sich ein Abschnitt an, in dem die Gesamtarchitektur und die funktionalen Module von TUPPSY vorgestellt werden.

Der restliche Teil dieses Kapitels ist zweigeteilt. Im ersten Teil wird **PLAKON**, eine im Rahmen des BMFT-Verbundprojektes TEX-K erstellte Planungsshell vorgestellt, die als Grundlage für die Realisierung von TUPPSY dient.[2]

Inhalt des zweiten Teils ist die Realisierung der funktionalen Module von TUPPSY mit Hilfe der von **PLAKON** zur Verfügung gestellten Mechanismen. Die Abschnitte 4.5 – 4.12 enthalten eine Beschreibung der Abbildung der einzelnen Module auf die verschiedenen Systemkomponenten. Ausführlich konnte allerdings nur das Produktmodell dargestellt werden, die restlichen Teile werden skizziert.

4.1 Konkretisierung der Problemstellung

Im vorherigen Kapitel wurden sowohl Systeme zur Planung prismatischer und rotationsymmetrischer Teile vorgestellt, die doch zum Teil recht unterschiedliche Problemstellungen bearbeiteten.

In diesem Abschnitt soll nun die Problemstellung genauer beschrieben werden, für deren Lösung im Rest des Kapitels ein Konzept erarbeitet wird.

Die Problemstellung läßt sich folgendermaßen charakterisieren :

> Gegenstand des Systems ist die *Arbeitsplanerstellung für weitgehend rotationssymmetrische Drehteile der Klein- und Mittelserienfertigung.*

[1] TUPPSY- TUrnpiece Process Planning SYstem

[2] **PLAKON** steht in diesem Zusammenhang als Vertreter für eine Reihe von modernen Expertensystemwerkzeugen, es wurde primär aus Gründen der Verfügbarkeit ausgewählt. Genausogut könnte ein anderes Werkzeug benutzt werden, das die gleichen oder ähnliche Mechanismen bereitstellt.

Unter weitgehend rotationssymmetrischen Drehteilen sollen in dieser Arbeit Drehteile verstanden werden, die aus rotationssymmetrischen Grundelementen bestehen, und die evtl. um nichtrotationssymmetrische Nebenelemente (wie z. B. Paßfedernuten) erweitert wurden.[3]

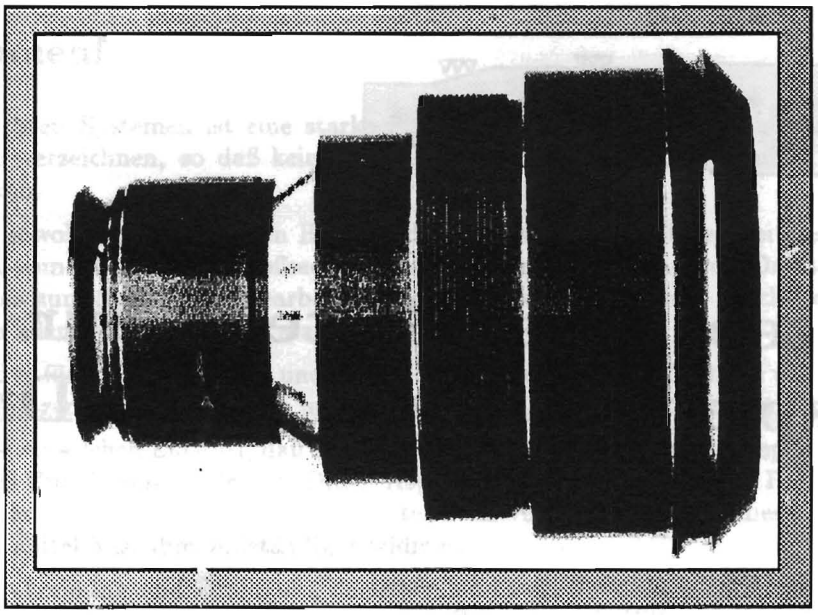


Abbildung 4.1: Ein weitgehend rotationssymmetrisches Drehteil

Die Problemstellung wird auf den Bereich der Klein- und Mittelserienfertigung eingeschränkt, da in diesem Umfeld eine „Standardwerkstatt" vorausgesetzt werden kann. Insbesondere wird auf eine Betrachtung der in der Großserienfertigung üblichen Sonderwerkzeug- und Sondermaschinenfertigung verzichtet.

Arbeitsplanerstellung wird als der Prozeß verstanden, in dessen Ablauf die folgenden Teilaufgaben bearbeitet werden:

- Maschinenauswahl
- Aufspannungsermittlung
- Arbeitsvorgangsfolgenbestimmung
- Werkzeugauswahl
- Schnittwertermittlung
- NC-Programmgenerierung

Nachfolgend sollen die einzelnen Teilaufgaben charakterisiert werden. Der Schwerpunkt liegt auf der Vorstellung der Einschränkungen die gegenüber einer möglichst allgemeinen Lösung und Kapitel 2 vorgenommen werden.

Maschinenauswahl Ziel der Maschinenauswahl ist es, unter den momentan zur Verfügung stehenden Maschinen eine für die Fertigung des Werkstücks geeignete zu finden. Stehen mehrere Maschine zur Auswahl, so sollte die zur Abwicklung des aktuellen Auftrags bestgeeignetste ausgewählt werden[4]

[3] Im weiteren Verlauf werden die hier verwendeten Begriffe Grund- und Nebenelement definiert.
[4] Das zur Bewertung herangezogene Kriterium kann so von Auftrag zu Auftrag variieren.

Es wird angenommen, daß es möglich ist das Teil auf *einer* Maschine zu fertigen. Ist keine Maschine in der Lage, das komplette Teil zu fertigen, so sollte dies festgestellt werden können. Unterstützung der Mehrmaschinenfertigung sollte nur soweit erfolgen, daß eventuell Vorschläge für Zwischenzustände des Werkstücks gemacht werden.

Eine weitere Einschränkung ist die Festlegung auf Drehmaschinen mit 1 Spindel, um so Probleme der Parallelplanung zu vermeiden.

Aufspannungsermittlung Die Aufspannungsermittlung umfaßt die Ermittlung der Zahl der Aufspannungen und die Auswahl der benötigten Spannmitteln und Vorrichtungen.

Dabei ist es von Vorteil, daß für die Fertigung von rotationssymmetrischen Teilen im Normalfall maximal 2 Aufspannungen gebraucht werden. Spannmittel und Vorrichtungen werden aus Katalogen unter Beachtung von Randbedingungen wie Verfügbarkeit, Oberflächengüte und maximaler Spannkraft ausgewählt. Falls eine Lünette benötigt wird, wird dies ebenfalls berücksichtigt.

Arbeitsvorgangsfolgenermittlung Ziel der Arbeitsvorgangsfolgenermittlung ist die Bestimmung von Art und Reihenfolge der auszuführenden Bearbeitungsschritte, um vom Roh- zum Fertigteil zu gelangen. Die Unterteilung des zu zerspanenden Volumens erfolgt unter Beachtung von Reihenfolgebedingungen. Dabei wird angenommen, daß für jeden bestimmten Bereich ein passendes Werkzeug gefunden werden kann[5].

Ein weiterer Gesichtspunkt der Arbeitsvorgangsfolgenermittlung ist die Beachtung von Optimalitätskriterien wie Anzahl der Werkzeuge, benötigte Zeit etc.

Werkzeugauswahl Hier werden die für die einzelnen Bearbeitungsschritte geeigneten Werkzeuge bestimmt. Es sind nur Werkzeuge nach DIN 4983 und 4987 zugelassen, d. h. Werkzeughalter mit Wendeschneidplatten. Die Auswahl der Werkzeuge erfolgt aus dem Sortiment der momentan verfügbaren. Evtl. muß auch ein geeignetest Scheidöl oder Kühlmittel ausgewählt werden.

Nach Bestimmung der für einen Arbeitsvorgang geeigneten Werkzeuge wird die Gesamtzahl der benötigten Werkzeugen über alle Arbeitsgänge minimiert, Werkzeugkosten werden nicht berücksichtigt.

Schnittdatenbestimmung Für die Arbeitsvorgänge werden die Schnittparameter Schnittiefe, Vorschub und Schnittgeschwindigkeit werden ermittelt.

Randbedingungen die durch die verwendete Maschine entstehen werden beachtet, Werkzeugkosten werden nicht berücksichtigt.

NC-Programmgenerierung Mit Hilfe der bestimmten Arbeitsvorgänge, Werkzeuge und Schnittparameter wird ein NC-Programm nach DIN 66 255 erzeugt.

4.2 Formalisierung

Abstrakt läßt sich die Problemstellung also wie folgt beschreiben:

Gegeben :

Die Mengen $\mathcal{M}, \mathcal{W}, \mathcal{B}, \mathcal{S}, \mathcal{WK}$ sowie zwei Werkstückbeschreibungen w_{roh} und w_{fertig} mit :

$\mathcal{M}$	$=$	$\{m_1, \ldots, m_{n_1}\}$	einer Menge von *Maschinen*,
$\mathcal{W}$	$=$	$\{w_1, \ldots\}$	einer Menge von *Werkstückbeschreibungen*,
$\mathcal{B}$	$=$	$\{b_1, \ldots, b_{n_2}\}$	einer Menge von *Bearbeitungsverfahren* mit $\forall k = 1, \ldots, n_2 : b_k : \mathcal{W} \times \mathcal{N}^\ni \rightarrow \mathcal{W}$,
$\mathcal{S}$	$=$	$\{s_1, \ldots, s_{n_3}\}$	einer Menge von *Spannmitteln*,
$\mathcal{WK}$	$=$	$\{wk_1, \ldots, wk_{n_3}\}$	einer Menge von *Werkzeugen*.

[5] Diese Heuristik ist durch entsprechende Erfahrungen aus der Praxis gerechtfertigt.

Weiterhin sei $Z = Vol(w_{roh}) - Vol(w_{fertig})$ das *zu zerspanende Volumen*, wobei $Vol : \mathcal{W} \rightarrow N^3$ und $w_{roh}, w_{fertig} \in \mathcal{W}$.

Gesucht :

Ein *Arbeitsplan* $AP = (M, ASp, AVF, Wk, SP)$ mit :

M	$=$	$\{m_1, \ldots, m_{l_1}\}$	einer Folge von *benutzten Maschinen*, wo $\forall j = 1, \ldots, l_1 : l_j \in \mathcal{M}$
ASp	$=$	$\{(a_1, j_1), \ldots, (a_{l_2}, j_{l_2})\}$	einer Folge von *Aufspannungen* mit $l_2 \leq 2l_1$; $\forall k = 1, \ldots, l_2 : j_k \in 1, \ldots, l_1$; $j_k \leq j_{k+1}$; $\forall k = 1, \ldots, l_2 : a_k \in \mathcal{S}$
AVF	$=$	$\{(av_1, j_1), \ldots, (av_{l_3}, j_{l_3})\}$	einer Folge von *Arbeitsvorgängen* mit $\forall k = 1, \ldots, l_3 : av_k = (b_k, v_k)$; $b_k \in \mathcal{B}$; $v_k \subseteq N^3$; $\bigcup z_k = Z$; $\forall k = 1, \ldots, l_3 + 1 : w_{k+1} = b_k(w_k, v_k)$; $w_1 = w_{roh}$; $w_{l_3+1} = w_{fertig}$; $\forall k = 1, \ldots, l_3 : j_k \in 1, \ldots, l_2$; $i_k \leq i_{k+1}$
Wk	$=$	$\{(wk_1, V_1), \ldots, (wk_{l_4}, V_{l_4})\}$	einer Menge von *benutzten Werkzeugen* mit $\forall k = 1, \ldots, l_4 : wk_k \in \mathcal{WK}$ $\forall k = 1, \ldots, l_4 : V_k$einer Menge von Arbeitsvorgangsindizes mit : $V_k = avi_1, \ldots, avi_{m_k}$ $\bigcup V_k = \{1, \ldots, l_3\}$; $\nexists l \in 1, \ldots, l_3 \text{mit} l \in V_{k_1}, V_{k_2} \text{und} k_1 \neq k_2$
SP	$=$	$\{(v_1, t_1, \omega_1), \ldots, (v_{l_3}, t_{l_3}, \omega_{l_3})\}$	einer Menge von *aktuellen Schnittparametern* mit $\forall k = 1, \ldots, l_3 : v_k$der aktuelle Vorschub für av_k; $t_k$die aktuelle Schnittiefe für av_k; $\omega_k$die aktuelle Schnittgeschwindigkeit für av_k;

4.3 Gesamtarchitektur und funktionale Module von TUPPSY

Insgesamt ergeben sich aus der Problemstellung heraus folgende Architekturüberlegungen für TUPPSY:

1. Verschiedene funktionale Module werden von einer Steuerung kontrolliert. Diese bestimmt, welches Modul aktiv ist, und welche Aktionen innerhalb des Moduls ausgeführt werden.
2. Die Daten die die einzelnen Module benötigen entstammen einem Produktmodell, das sowohl statische als auch dynamische Daten enthält.
3. Die Beeinflussung der Steuerung erfolgt über Strategien, die auf der globalen Ebene, der Ebene der funktionaler Module und innerhalb der Module definiert werden können und somit eine hierarchische Planung realisieren.
4. Die einzelnen funktionalen Module können sich externer Programme bedienen, deren Anschluß durch eine Erweiterung des unterliegenden Planungssystem PLAKON bewerkstelligt wird.
5. Die Teilaufgaben der Problemstellung entsprechen funktionalen Modulen in TUPPSY.
6. Hinzu kommen Module zur Werkstückdefinition, bzw. zur Erstellung der internen Werkstückrepräsentation aus CAD-Daten und zur Simulation des NC-Programms.

Abbildung 4.2 zeigt die Gesamtarchitektur von TUPPSY.

Bevor ab 4.5 auf die Inhalte einiger funktionaler Einheiten eingegangen wird, soll zuerst die Planungsshell PLAKON vorgestellt werden. Sie dient als Basis für die weiteren Überlegungen dieses Kapitels.

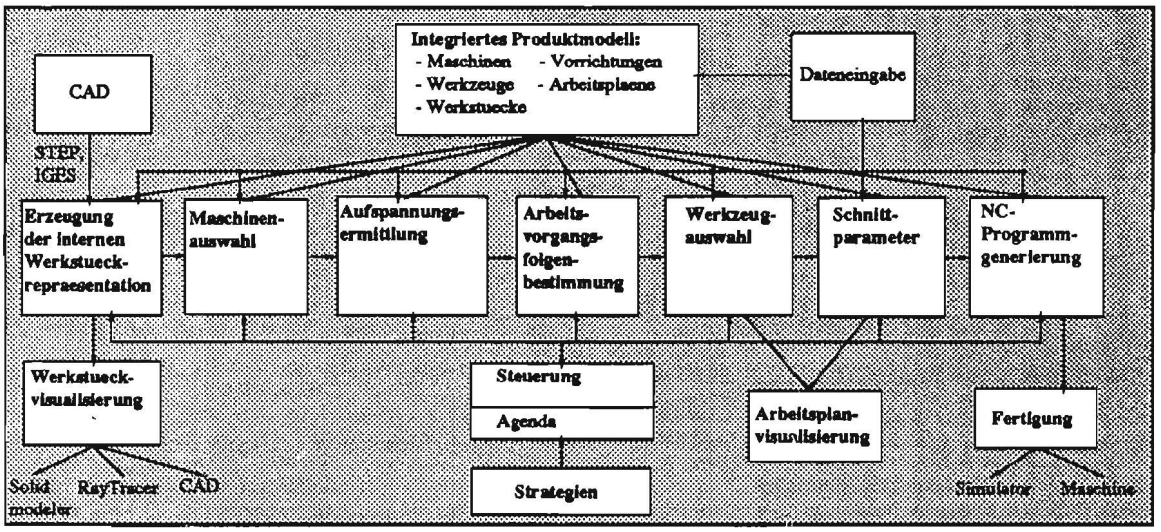


Abbildung 4.2: Die Gesamtarchitektur von TUPPSY

4.4 Die Planungsshell PLAKON

TUPPSY beschreibt die funktionalen Einheiten eines Systems zur Arbeitsplanerstellung für Drehteile. Dieses System muß mit einem KI-Planungssystem realisiert werden. In diesem Abschnitt soll die Planungsshell PLAKON vorgestellt werden, die als Grundlage für die Realisierung von TUPPSY dient.[6]

PLAKON wurde für den Bereich der Planungs- und Konfigurierungsaufgaben, speziell in technischen Anwendungen, entwickelt. Dieser Aufgabenbereich ist dadurch gekennzeichnet, daß aus Einzelkomponenten (Planschritten, Bauteilen) eine Gesamtheit (ein Plan, eine Konfiguration) unter Berücksichtigung einer Menge von Randbedingungen *konstruiert* werden soll. Diese Gesamtheit wird in PLAKON als *Konstruktion* bezeichnet, ihre Komponenten als *Konstruktionsobjekte.* Planung und Konfigurierung werden unter dem Begriff *Konstruktionsvorgang* zusammengefaßt. Ein Konstruktionsvorgang besteht aus einer Sequenz von *Konstruktionsschritten.*

Für PLAKON wurden unter anderem drei wesentliche Eigenschaften angestrebt:

- umfassende, den Konstruktionsvorgang unterstützende Repräsentationsmöglichkeiten für Konstruktionsobjekte,
- leistungsfähige Darstellungs- und Verarbeitungsmechanismen für numerische und nicht-numerische Abhängigkeiten innerhalb einer Konstruktion,
- flexible Ablaufsteuerung zur Realisierung typischer Konstruktionsstrategien (interaktiv, hierarchisch, opportunistisch).

4.4.1 Das Konzept von PLAKON

Ausgehend von einer losen Menge von Objekten, die von einer Aufgabenstellung gefordert sind, wird die Konstruktion schrittweise aufgebaut. Dazu stehen die folgenden Konstruktionsschritttypen zur Verfügung:

[6] Weitere Informationen über PLAKON findet man in [CGS91], dem auch Teile dieses Abschnittes entstammen.

- Objekte *zerlegen*: Dekomposition; dies entspricht der Zerlegung einer Gesamtaufgabe in Teilaufgaben.
- Komponenten *zusammenfügen*: Integration (Aggregation); dies entspricht einem Bottom-Up-Vorgehen.
- Objekte *spezialisieren*: Verfeinerung in der taxonomischen Hierarchie der Objekte.
- Objekte *parametrieren*: Dies entspricht der Festlegung von Parametern und Eigenschaften (z. B. Länge, Radius eines Werkstücks etc.) unter Berücksichtigung von Randbedingungen.

Das dazu notwendige konzeptuelle Domänenwissen wird deklarativ in einer Begriffshierarchie (siehe 4.4.2) dargestellt. Sie enthält eine kompositionelle Hierarchie (für Zerlegung und Aggregation) und eine taxonomische Hierarchie (für Eigenschaftsbeschreibung und Spezialisierung). Instanzen der dort beschriebenen Konzepte werden zum Aufbau der Konstruktion verwendet.

Randbedingungen der Konstruktion werden mit Hilfe von Constraints formuliert (siehe 4.4.3), zur Steuerung des Ablaufes und der Auswahl des nächsten Konstruktionsschrittes steht der *begriffshierachie-orientierte Kontrollansatz* (siehe 4.4.5) zur Verfügung. Dieser benutzt unter anderem Produktionsregeln (siehe 4.4.4), die von FRESKO einer CLOS-ähnlichen objektorientierten Erweiterung von COMMONLISP bereitgestellt werden.[7] Für die Arbeitsplanerstellung ist weiterhin die Bereitstellung von dimensionierten Zahlen sehr nützlich.

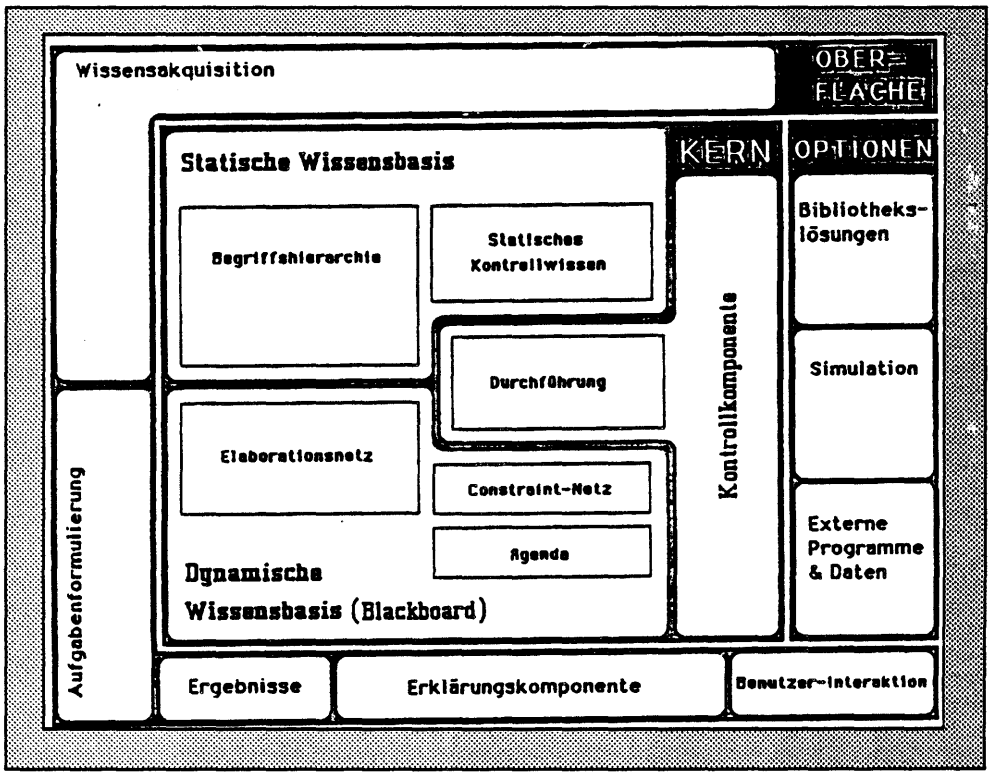


Abbildung 4.3: Die Architektur von PLAKON

Hier sollen die verschiedenen Repräsentationsformalismen nur soweit skizziert werden wie es für das weitere Verständnis notwendig ist. Für eine weitergehende Darstellung der grundlegenden Ideen sei auf die einschlägige Expertensystemliteratur (z. B. [Pup88, Ric89]) verwiesen, die Realisierung in Plakon kann [CGS91] entnommen werden.

4.4.2 Objekte und Begriffshierarchie

Idee Die Menge der Objekte $\mathcal{O}$ ist in *Klassen* $\mathcal{O} = \{K_1, \ldots, K_n\}$ partitioniert, d. h. $\bigcup K_i = \mathcal{O}$, $M_i \cap M_j = \emptyset$ für $i \neq j$. Eine Klasse K_i besteht aus einer Menge von Objekten $\{m1, \ldots, m_k\}$ den *Instanzen* dieser Klasse.

[7] FRESKO wurde ebenfalls im Rahmen von TEX-K an der Universität Hamburg entwickelt.

Auf der Menge der Klassen ist eine Partialordnung $\sqsubseteq$ definiert. Einschränkend wird die Zahl der direkten Vorgänger $dv(K_i)$ einer Klasse K_i (d. h. die Menge $\{K_j \mid K_i \sqsubseteq K_j \wedge \nexists K_k \text{mit} K_i \sqsubseteq K_k \sqsubseteq K_j, K_k \neq K_i, K_j\}$) auf maximal 1 eingeschränkt ($\mid dv(K_i) \mid \leq 1$) und es existiert genau eine Klasse K_w mit $\mid dv(K_i) \mid = 0$, die *Wurzelklasse.*[8]

Die so definierte Hierarchie wird zur ökonomischen Datenhaltung benutzt, indem sie eine *Vererbungshierarchie* impliziert. Statt bei jedem Objekt all seine Eigenschaften abzuspeichern, werden nur individuelle Eigenschaften bei dem Objekt selbst verzeichnet, während allgemeinere Eigenschaften den Vorgängern in der Hierarchie zugeordnet und an die Nachfolger vererbt werden. Bei Abfragen wird der Wert der Eigenschaft zunächst in dem Objekt selbst gesucht, dann in seinem Vorgänger etc.

Die Klassen werden mit einer Klassenbeschreibung spezifiziert. Eine Klassenbeschreibung besteht aus einem Klassennamen, einem Superklassennamen, und einer Menge von *Slots.* Ein Slot besteht wiederum aus einem Slotnamen und einem Slotwert. Optional können Slots einen *Defaultwert* und eine Reihe von *Facetten* besitzen.

Durch *Instantiierung*, d. h. der partiellen Slotbestimmung werden konkrete Objekte (*Instanzen*) gebildet.

In den *Slots* werden die Eigenschaften gespeichert. Sie können mehrere *Facetten* besitzen, die ebenfalls einen Wert besitzen können. Eine vordefinierte Facette dient zur Speicherung eines *Defaultwertes*, andere Facetten können z. B. den Wertebereich definieren oder aktive Werte (d. h. zugeordnete Prozeduren `if-added`, `if-needed`, ...) realisieren.

Realisierung in PLAKON Objekte sind in Plakon auf zwei Ebenen realisiert:

- FRESKO

  Das Framerepräsentationskonzept FRESKO deckt sich in wesenlichen Zügen mit CLOS und kann mit diesem als Basis implementiert werden. Es erweitert dieses um das Konzept der Facetten an Slots und der erweiterbaren Frames.[9] Darüberhinaus stellt es zwei Pakete zur Behandlung von Windows und zur Propagation von Regeln bereit, die als Zusätze mit den Mitteln von FRESKO definiert sind, und die durch die Anforderungen von PLAKON initiiert wurden.

  In FRESKO wird jedes Datenelement als Objekt aufgefaßt, wobei neben den COMMONLISP-Datentypen noch *Frames* zur Verfügung gestellt werden. Ein Frame ist eine Struktur, die beliebig viele Slots enthält. Jeder Slot hat einen Bezeichner, einen Wert und eine Reihe von Facetten, die wiederum aus Bezeichner und Wert bestehen.

  Die Funktionalität eines Frames wird durch *generische Funktionen* bestimmt. Dies sind Funktionen die unterschiedliche Prozedurrümpfe, die *Methoden*, in Abhängigkeit der Klassen ihrer Parameter ausführen. Die Klassen der Parameter wirken als *Diskriminatoren*, wird eine generische Funktion aufgerufen, so wird die speziellste Methode ausgewählt, deren Diskriminatoren den Argumenten des Aufrufs entspricht. Dadurch ergibt sich eine implizite Methodenvererbung, ohne daß sich dies in einer Veränderung der Syntax wie bei message-parsing Mechanismen (z. B. in `Smalltalk-80`) niederschlägt.

Beispiel eines Frames:

```
(create-class fahrzeug
     :super frame-object
     :meta hashtable-class
     :instance-slots
         (hersteller
         (motor otto-motor (zylinder 4) (einspritzung nil))
         (geschwindigkeit 200)
         (tueren 4))
     :class-slots
```

[8] Dadurch wird die prinzipiell mögliche Klassenheterarchie zu einer Klassenhierarchie.

[9] Außerdem ist in FRESKO im Gegensatz zu CLOS das Konzept der Metaklassen klar definiert, was sich in Kapitel 5 noch als nützlich erweisen wird.

```
((definiert-von 'meier)))
```

Beispiel einer generischen Funktion

```
;; Definition der Argumentliste der generischen Funktion

(defgeneric gewicht (teil1 &rest weitere-teile &allow-other-keys))

;; Definition einer Methode

(defmethod gewicht ((motor teil1) &rest weitere-Teile)
    (+ (* motor (slot-value motor 'spez-gewicht))
       (apply #'+ weitere-teile)))
```

Die interne Repräsentationsform eines Frames wird durch seine *Metaklasse* bestimmt. In FRESKO kann ein Frame zusätzlich zu seiner primären Metaklasse noch sogenannte *funktionale Metaklassen* haben, die z. B. *Dämonen* realisieren. Diese Erweiterung kann zudem nur temporär sein, um z. B. das Debugging zu erleichtern.

Das FRESKO-Regelkernsystem bietet einen vorwärtsverkettende Regel- und Matchmechanismus in Art und Umfang von OPS-5 an, während das FRESKO-Windowkernsystem eine Hierarchie von Fenstertypen zur Verfügung stellt.

- BHIBS

  Mit Hilfe der BegriffsHIerarchie-BeschreibungsSprache (BHIBS) wird die PLAKON-Begriffshierachie formuliert. In der Begriffshierarchie werden alle Objekte der Anwendungsdomäne und die Abhängigkeiten zwischen ihnen konzeptuell beschrieben. Diese Konzepte bilden eine taxonomische (*is-a*-)Hierarchie, die dazu dient, Objekte zu typisieren, Klassen und Spezialisierungen (Unterklassen) von Objekten zu beschreiben und deren Eigenschaften festzulegen. Eigenschaften werden innerhalb der Hierarchie vererbt. Konzepte werden durch einen Bezug zu einem übergeordneten Konzept und durch eine Menge von Eigenschaften oder Parametern beschrieben, die das Konzept charakterisieren (und für die Konstruktion relevant sind). *Objektdeskriptoren* beschreiben die zulässigen Wertemengen für Eigenschaften. Zusätzliche Angaben an den Eigenschaften können genutzt werden, um z. B. Defaults oder Werte-Berechnungsfunktionen vorzugeben. Intern wird solch ein Konzept als ein *Frame* dargestellt, dessen *Slots* (Felder) die Objektdeskriptoren aufnehmen und in dem *Facetten* an den Slots die zusätzlichen Angaben aufnehmen. Objektdeskriptoren können Anzahlmengen, Zahlintervalle, Prädikate oder wiederum Konzepte von anderen Objekten sein. Semantisch repräsentieren sie *einen* Wert, für den nur bekannt ist, daß er in der angegebenen Menge enthalten ist.

Beispiel für eine Konzeptdefinition

```
;; ein Auto ist ein Konstruktionsobjekt mit den Eigenschaften...
(ist! (ein Auto)
      (ein Konstruktionsobjekt
           (Geschwindigkeit [0km/h 300km/h]
                (Default 100km/h))
           (Farbe {rot gruen blau schwarz}) ; einer der Werte...
           (Hersteller (eine Autofirma))
           (Has-Parts (:set (ein Motor) ; eine Menge bestehend aus...
                            (eine Karosserie)
                            (ein Fahrgestell)))))
```

Spezialisierende Konzepte dürfen vorhandene Slotbeschreibungen nur einschränken, nicht aber überschreiben, und zusätzliche Eigenschaften einführen:

Eine Konzeptspezialisierung

```
;; ein LKW ist ein langsames,schweres Auto mit einem Dieselmotor
(ist! (ein LKW)
      (ein Auto
           (Geschwindigkeit [0km/h 120km/h])
           (Hoechstlast [0t 30t])
           (Has-Parts (:set (ein Dieselmotor)
                            (eine Karosserie)
                            (ein Fahrgestell
                                 (Achslast [0t 40t])))))))
```

Dadurch stellt die taxonomische Hierarchie eine strenge Spezialisierungshierarchie dar, die sich an Ideen von KL-ONE orientiert. Sie ermöglicht den Einsatz eines Klassifikators (*Classifiers*) zur *dynamischen Spezialisierung* und zur Unterstützung der Wissensakquisition.

Für die taxonomische Hierarchie gilt die *„Closed-World-Assumption"*, bei der angenommen wird, daß der beschriebene Weltausschnitt im Sinne der Aufgabenstellung *vollständig* ist.[10]

Über diese taxonomische Hierarchie wird eine *kompositionelle Hierarchie* gelegt, die die Zerlegung von Konstruktionsobjekten in Komponenten beschreibt. Die *has-parts-* (bzw. *part-of-*) Beziehung ist die Grundlage für die Erstellung der fertigen Konstruktion. In den Konzept-Frames wird die *has-parts-*Relation durch mengenwertige Slots dargestellt, die Verweise auf die Konzepte der Teilkomponenten enthalten (s. Beispiele oben). Um mehrfaches Vorkommen von Komponenten eines Typs in einem Aggregat ausdrücken zu können, kann mit dem entsprechenden Konzept eine Anzahlangabe assoziiert werden.

Die kompositionelle Hierarchie

```
(ist! (ein Motor)
      (ein Autoteil
           (has-parts (:set (:some (ein Zylinder) 2 12)
                            (ein Vergaser) ...))))
```

Zusätzlich können in der Begriffshierarchie mit den Objektkonzepten Nebenbedingungen assoziiert werden, die die Wertzuweisung an die Parameter eines Objektes kontrollieren und die Zusammenhänge zwischen Parametern eines oder mehrerer Objekte beschreiben. Diese Nebenbedingungen werden in der Begriffshierarchie durch *konzeptuelle Constraints* (siehe 4.4.3) ausgedrückt.

4.4.3 Constraints

Als weiterer Wissensrepräsentationsformalismus werden in PLAKON *Constraints* bereitgestellt. Sie dienen der Darstellung von Randbedingungen, die eine Lösung erfüllen muß. Durch jedes Constraint wird der

[10] Diese kritische Forderung kann in technischen Domänen guten Gewissens aufgestellt werden, weil dort die Zahl der Objekte einigermaßen begrenzt ist und alle Komponenten und ihre Eigenschaften bekannt sein *müssen*, um eine Konstruktion durchführen zu können.

Lösungsraum zusätzlich eingeschränkt. Constraints sind im Gegensatz zu Regeln ungerichtet und können als mathematische Gleichungen aufgefaßt werden. Je nach Definitionsart können extensionale, intentionale und funktionale Constraints unterschieden werden.

Durch gemeinsame Teile werden einzelne Constraints zu einem *Constraintnetz* zusammengesetzt. Durch Propagierung innerhalb des Constraintnetzes können sich die Einschränkungen gegenseitig aufschaukeln und so den Lösungsraum stark einschränken.

Im allgemeinen beschreiben Constraints über eine *Constraint-Relation* Abhängigkeiten zwischen einer Menge von *Constraint-Variablen*. Die Constraint-Relation definiert dabei die Art der Abhängigkeit. Werden bestimmte Variablen eines Constraints mit Werten belegt, so sorgt es dafür, daß für die übrigen Anschlüsse Werte entsprechend der Constraint-Relation generiert werden. Werden dagegen von vornherein alle Anschlüsse mit Werten belegt, so kann das Constraint feststellen, ob die Anschlußbelegung in Bezug auf die Constraint-Relation konsistent ist [Gü87].

In PLAKON wird ein dreistufiges Constraint-Modell verwendet. Auf der obersten Stufe stehen die sogenannten *Constraint-Klassen*. Sie stellen domänenunabhängige Spezifikationen von Abhängigkeitsbeziehungen zur Verfügung (z.B. Adder, Multiplier). Die Constraint-Klassen werden durch ein in PLAKON integriertes, domänenunabhängiges Constraint-System bereitgestellt, das dem an der GMD entwickelten CONSAT [Gü87] nachempfunden wurde.

Aufbauend auf den Constraint-Klassen können auf der zweiten Stufe des Constraint-Modells sogenannte *konzeptuelle Constraints* definiert werden, die die in der betrachteten Domäne bestehenden Abhängigkeiten beschreiben. Sie stellen eine Zuordnung von Objekten bestimmter Konzepte und ihren Parametern zu Constraint-Klassen dar.

Beispiel eines konzeptuellen Constraints

„Der Hubraum eines Motors ist gleich dem Produkt aus Zylinderanzahl und Hubraum der einzelnen Zylinder"

```
(constrain ((#?M (ein Motor))
            (#?Z (ein Zylinder (part-of #?M))))
   (multiplier (#?Z Hubraum) (#?M Zylinderzahl) (#?M Gesamthubraum)))
```

Die Definition besteht aus zwei Teilen. Der erste Teil, Domain-Pattern genannt, spezifiziert diejenigen Objekte, an die das Constraint gebunden werden soll. Der zweite Teil der Definition beschreibt die Zuordnung der Constraint-Klasse `Multiplier` an die spezifizierten Eigenschaften bzw. Slots der Objekte.

Immer, wenn während des Konstruktionsvorganges Objekte erzeugt werden, die das Domain-Pattern erfüllen, werden die zugehörigen konzeptuellen Constraints instantiiert und an die Slots dieser Objektinstanzen gebunden. Dadurch, daß mit jedem Instanzenslot mehrere Constraint-Variablen verbunden sein können, entsteht eine Vernetzung der instantiierten Constraints zu einem *Constraint-Netz*. Dieses Netz bildet die dritte Stufe im verwendeten Constraint-Modell und spiegelt die Menge der Abhängigkeiten innerhalb der aktuellen Konstruktion wider. Es wächst dynamisch mit der Erzeugung neuer Objekte.

Der Kontrollmechanismus von PLAKON kann zu jeder Zeit die Propagation des Constraint-Netzes anstoßen und damit Parameterfestlegungen durchführen oder überprüfen:

- Unbekannte Slotwerte können anhand von Constraints aus bereits bekannten Slotwerten abgeleitet werden, und Mengen zulässiger Werte für einen Parameter können weiter eingeschränkt werden.
- Bei gegebener Abhängigkeitsstruktur können Slotwerte auf ihre Konsistenz überprüft werden.

Außer Abhängigkeiten zwischen Eigenschaften von Objekten gibt es noch eine weitere, häufige Art von Einschränkungen, die die Existenz von Objekten fordern oder verbieten [DL81]. Um diese Art von Constraints repräsentieren und evaluieren zu können, sind zwei spezielle Constraint-Klassen `Exist` und `Not-Exist` eingeführt worden.

Beispiel eines `Exist`-Contraints

„Wenn ein Auto nach Österreich exportiert werden soll muß es einen Katalysator haben"

```
(constrain ((#?A (ein Auto (exportiert-nach 'Oesterreich))))
   (exist (ein Katalysator (part-of #?A))))
```

4.4.4 Regeln

PLAKON selbst benutzt Regeln nur innerhalb der Kontrollkomponente (siehe 4.4.5) und stellt keinen allgemeinen Regelmechanismus zur Verfügung. Dieser ist aber in FRESKO implementiert, er wird nur nicht in seiner vollen Mächtigkeit nach oben durchgereicht. Wenn man ihn benutzten will muß dies auf Umwegen geschehen, es ist aber möglich. FRESKO bietet *Vorwärtsregeln*, keine *Rückwärtsregeln*. Die Vorwärtsregeln sind ganz normale Produktionsregeln. Sie bestehen aus einem Bedingungs- und einem Aktionsteil. Der Bedingungsteil, der aus einer Konjunktion von Prädikaten besteht beschreibt ein Muster das durch Übereinstimmung mit Teilen der Wissensbasis erüllt wird. Es wird so eine Situation beschrieben, in der die Aktionen ausgeführt werden sollen.

Die Regel $\mathcal{R}$ sind in Regelmengen $\mathcal{R} = \{R_1, \ldots, R_m\}$ unterteilt, die jeweils eine Reihe von Regeln umfassen. Wahlweise kann der Matchmechanismus daten- oder regelgetrieben erfolgen. Ein *datengetriebenes* Regelsystem ist dadurch charakterisiert, daß die Instantiierung der Regeln durch Neueinträge und Änderungen im Datenbestand ausgelöst werden. Der Matchmechanismus wird durch Kopplung einer Datenmenge mit einer Regelmenge in Gang gesetzt. Bei der funktionalen Regelauswertung wird eine Regelmenge auf eine Datenmenge angewandt und keine Information über teilweise erfolgreiche Matches gehalten. Dem Nachteil des erhöhten Matchaufwandes steht hier der Vorteil der geringeren Verwaltungskosten gegenüber.

Als Zusatz wurde in FRESKO eine OPS-5 nachgebildete Regelsprache implementiert. Ein Beispiel eine Regel in dieser Regelsprache zeigt das nachfolgende Beispiel

Eine Regel in P-Syntax

```
(p beispiel-regel
     (#?FIRMA (auto-firma (name #?N)))
     (> 3 (fahrzeug (fahrzeugtyp #?N) (farbe 'rot))) --> (format t Firma ~ A liebt rote Autos." #?N))
```

4.4.5 Der begriffshierarchie-orientierte Kontrollansatz

Im *begriffshierarchie-orientierten Ansatz* verwendet der Konstruktionsprozeß die Begriffshierarchie als Leitfaden. Teilkonstruktionen, die ja Teil-Instantiierungen der Begriffshierarchie verkörpern, werden ausschließlich entsprechend der darin enthaltenen *syntaktischen* Information weiterentwickelt. Auf diese Weise wird sichergestellt, daß die resultierende Konstruktion immer ein Element der Menge der zulässigen Konstruktionen repräsentiert. Sie wird außerdem garantiert alle Einschränkungen, die mit der initialen Teilkonstruktion vorgegeben wurden, erfüllen, vorausgesetzt daß eine Lösung existiert, weil die Endkonstruktion aus der initialen Teilkonstruktion in einem strikt monotonen Verfeinerungsprozeß abgeleitet wird.

Beim begriffshierarchie-orientierten Kontrollkonzept wird eine klare Trennung zwischen unterschiedlichen Wissensinhalten erzwungen. Kontrollwissen wird explizit und deklarativ repräsentiert. Der Konstruktionsvorgang von PLAKON wird nur durch separate, explizite Kontrollentscheidungen gesteuert.

Der *zentrale Zyklus* des Konstruktionsvorganges besteht aus folgenden Schritten:

1. Es wird die aktuelle Teilkonstruktion bestimmt.

2. Die ausgewählte Teilkonstruktion wird daraufhin untersucht, welche Konstruktionsschritte noch durchgeführt werden müssen. Wie bereits ausgeführt, kennt PLAKON folgende Typen von Konstruktionsschritten:

   - *Zerlegen* (Dekomponieren) von Objekten gemäß der *has-parts*-Hierarchie durch Instantiieren der Komponenten.
   - *Spezialisieren* von Objekten entlang der *is-a*-Kanten.
   - *Integration* von Teilkomponenten entlang der *has-parts*-Kanten, d.h. Instantiierung von Aggregaten und fehlenden Teilen, sobald ausreichend Teilkomponenten eines Konstruktionsobjektes vorhanden sind. Dies enthält außerdem das *Verschmelzen* verschiedener Instanzen in einer Teilkonstruktion, die eindeutig dasselbe reale Objekt beschreiben.
   - *Parameterwerte* festlegen oder einschränken. Dabei spielt die Propagation von Nebenbedingungen eine wesentliche Rolle.

   Zu jedem dieser Konstruktionsschritt-Typen existiert ein „Spezialist", der feststellt, ob an einem Konstruktionsobjekt ein entsprechender Konstruktionsschritt durchgeführt werden muß, und wenn ja, einen entsprechenden Eintrag in eine *Agenda* generiert. Durch eine Menge von *Auswahlkriterien* wird aus der Agenda ein Eintrag (entspricht einem Konstruktionsschritt) ausgewählt. Die Auswahlkriterien beinhalten Wissen darüber, in welcher Reihenfolge die Konstruktionsschritte am erfolgversprechendsten ausgeführt werden können.

3. Der ausgewählte Konstruktionsschritt wird durchgeführt. Falls kein eindeutiger Wert als Resultat feststeht, können verschiedene Bearbeitungsverfahren zur Festlegung angewendet werden:

   - einen Defaultwert einsetzen,
   - eine Berechnungsvorschrift oder -regel anwenden,
   - den Benutzer fragen,
   - einen Wert aus einer bereits erfolgten und in einer *Bibliothek* gespeicherten Konstruktion übernehmen, falls „nichts dagegen spricht",

   Bei der Durchführung eines Konstruktionsschrittes werden eine neue Teilkonstruktion und eine Elaborationskante generiert.

4. Abschließend können durch Propagation des Constraint-Netzes

   - aus der Durchführung resultierende Einschränkungen an andere Objekte und Slots der Teilkonstruktion weitergereicht werden,
   - Konflikte, die aus einer inkonsistenten Teilkonstruktion resultieren, erkannt werden.

4.4.6 Dimensionierte Zahlen

Dimensionierte Zahlen sind Zahlen mit Maßangaben wie z. B `3cm`, `4KW`, `3s`. FRESKO bietet nun die Möglichkeit, dimensionierte Zahlen wie normale Zahlen zu benutzen und mit ihnen zu rechnen. Gerade für den Bereich der Arbeitsplanung mit seiner Vielzahl von Maßen und den großen Unterschieden in der Genauigkeit ist die Benutzung von dimensionierten Zahlen von großem Nutzen[11].

Dazu werden die folgenden Funktionen angeboten:

- Definieren von Maßeinheiten und den Beziehungen zwischen ihnen.
- Repräsentation und Eingabe von Zahlen mit Maßeinheiten.
- Beschreiben eines Ausgabeformates für definierte Größen.
- Löschen von Maßeinheiten.

[11] So wäre zum Beispiel die Eingabe einer Oberflächengüte von 2μm als 0,000002 für viele Benutzer inakzeptabel

4.4.7 Externe Schnittstelle

Im Rahmen dieser Diplomarbeit wurde PLAKON um eine externe Schnittstelle erweitert[12]. Die externe Schnittstelle dient dem Anschluß existierender Programme und Daten, um so zu verhindern, daß einerseits schon vorhandene Software (evtl. schlechter) reimplementiert und andererseits vorhandene Daten (evtl. fehlerhaft) redundant gespeichert werden.

Die Schnittstelle besteht aus einem (oder mehreren) Servern auf den Anwendungsrechnern, den als Klienten betriebenen Anwendungen und einem Satz von Kommunikationsfunktionen im Lisp-System. Das Lisp-System wendet sich über einen Satz definierter Nachrichten an die Anwendungen. Die empfangenen Nachrichten werden in Aktionen umgesetzt und deren Werte an das Lisp-System zurückübermittelt.

Aufbauend auf den Kommunikationsfunktionen werden die Anwendungen als Objekte in die Objekthierarchie eingegliedert, so daß sie innerhalb des Lispsystems transparent behandelt werden können.

Verschiedene Architekturvarianten der externen Schnittstelle, ihre Diskussion und die Auswahl einer konkreten Variante sind ebenso Teil von Kapitel 5 wie die Realisierung zweier Arten von Datenbankanbindung und dem Anschluß eines Solid Modelers als Beispielanwendungen der externen Schnittstelle, so daß an dieser Stelle auf eine genauere Vorstellung verzichtet wird.

4.5 Das Produktmodell

Produktmodelle vereinigen Geometrie- und Strukturbeschreibungen des Werkstücks mit einem Werkstattmodell und Teilen eines Prozeßmodells. Im Vergleich zu allgemeingültigen Produktmodellen[13] wie STEP/IPIM erfolgt in TUPPSY eine Einschränkung des Produktmodells auf die für die Problemstellung relevanten Teile[14]

Es sind dies im Rahmen des Werkstückmodells die Teilmodelle:

- Bemaßung
- Darstellung
- Geometrie
- Technologie
- Funktionselemente

Das Werkstattmodell umfaßt die Teile:

- Werkzeuge
- Maschinen
- Spannmittel
- Werkstoffe

Das Prozeßmodell besteht im wesentlichen aus einem Arbeitsplanmodell.

Weiterhin wird das Produktmodell auf die Beschreibung von Einzelteilen beschränkt, so daß die in [BKL90] genannten Teilmodelle zur Beschreibung zusammengesetzter Produkte wie Komponentenmodell etc. verzichtet werden kann.

Realisiert wird das Produktmodell in TUPPSY als Teil der Objekthierarchie, wobei der immanenten Generalisierungshierarchie eine Aggregationshierarchie mit Hilfe von `part-of` Slots überlagert wird.

Hinzu kommen Constraints zur Konsistenzerhaltung bzw. Slotwertbestimmung [15]

[12] Die Erweiterung ist allerdings nicht nur für PLAKON benutzbar, es wird nur ein Lispsystem mit einer objektorientierten Erweiterung benötigt.

[13] Und damit auch großen, unübersichtlichen und unhandlichen

[14] Entsprechend der Großbereiche Allgemeine Produktdefinierende Daten und NC-Programmierung von ARC-TEC.

[15] Aus Effizienzgründen ist eine Realisierung der Slotwertbestimmung mit Dämonen denkbar, was jedoch die Reihenfolge der Slotwertbestimmung inflexibler macht

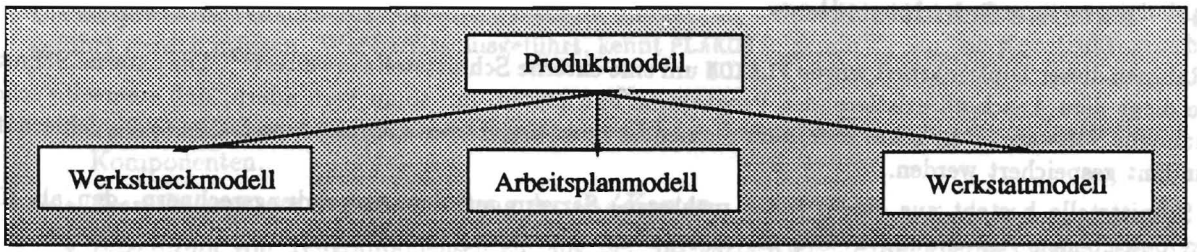


Abbildung 4.4: Die Aggregationshierarchie des Produktmodells

4.5.1 Werkstückmodell

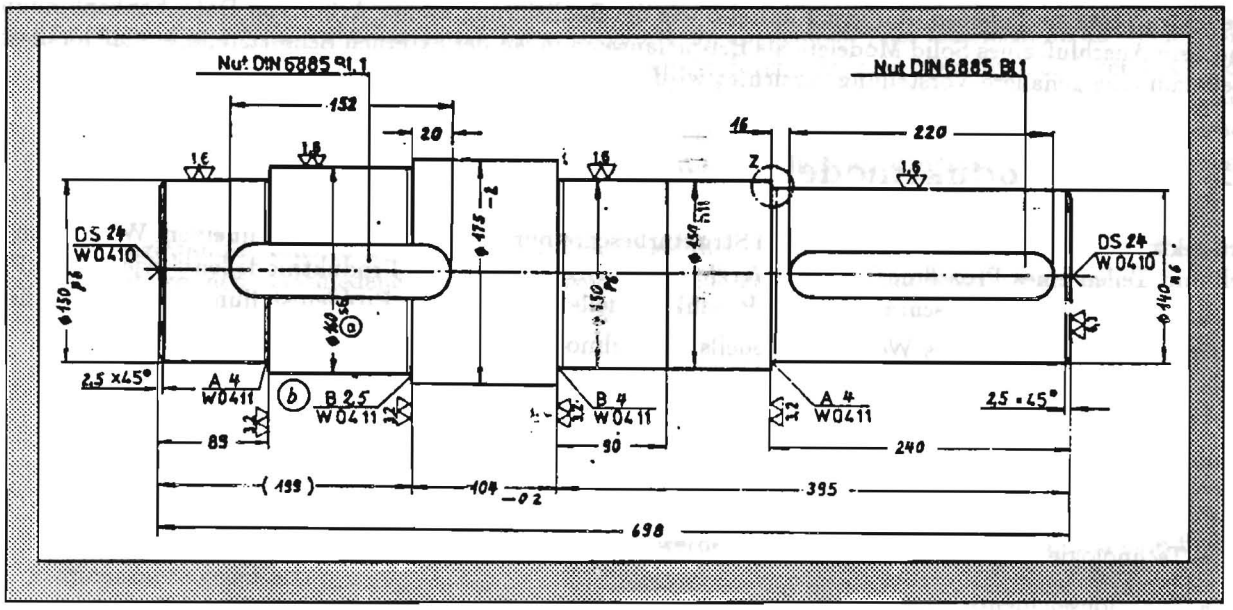


Abbildung 4.5: Eine Werkstückzeichnung

Die Struktur des Werkstückmodells nimmt zentralen Einfluß auf das Aussehen und die Qualität des darauf aufbauenden Arbeitsplanungssystems. Daher sollte seinem Entwurf besondere Aufmerksamkeit gewidmet werden. So werden vor der Vorstellung eines eigenen Modellvorschlags verschiedene Werkstückmodelle diskutiert, wobei der Schwerpunkt auf sich momentan in Kaiserlautern in Entwicklung befindlichen Systemen liegt. Dabei verwischen sich die zunächst scharf gezogenen Grenzen zwischen den Teilmodellen zugunsten einer pragmatischen Mischform, in der von einer von der Geometrie dominierten Begriffshierarchie auf die technologische Elemente etc. verwiesen wird.

Existierende Werkstückmodelle

TechMo An der Universität Kaiserslautern wurde im Rahmen des Projektes TechMo (einem DB-gestützten technischen Modellierer) das folgende Werkstückmodell entworfen.[HPS90, HPS89]

Ein Werkstück ist in `TechMo` ein *technisches Aggregat*, das aus `Haupt-`, `Neben` und `Funktionselementen` besteht.

`Hauptelemente` bezeichnen rotationssymetrische Fonelemente wie, z.B. Absätze, Kegel etc. und enthalten neben den geometrischen Daten auch technologische wie Werkstoff, Oberflächengüte, Toleranzen usw.

`Nebenelemente` haben gegenüber Hauptelementen attributierenden Charakter. Sie werden aus einer möglichen Anzahl von Varianten ausgewählt und einem Hauptelement zugeordnet. Nebenelemente können eigene

Informationen besitzen und unter Umständen genormt sein. Beispiele für Nebenelemente sind Freistiche und Einstiche.

Ein **Funktionselement** repräsentiert den Bereich eines Einzelteils, durch den eine Funktion eines Einzelteils realisert wird. Es ist eine funktionsbedingte Gesamtheit von Haupt- und Nebenelementen.

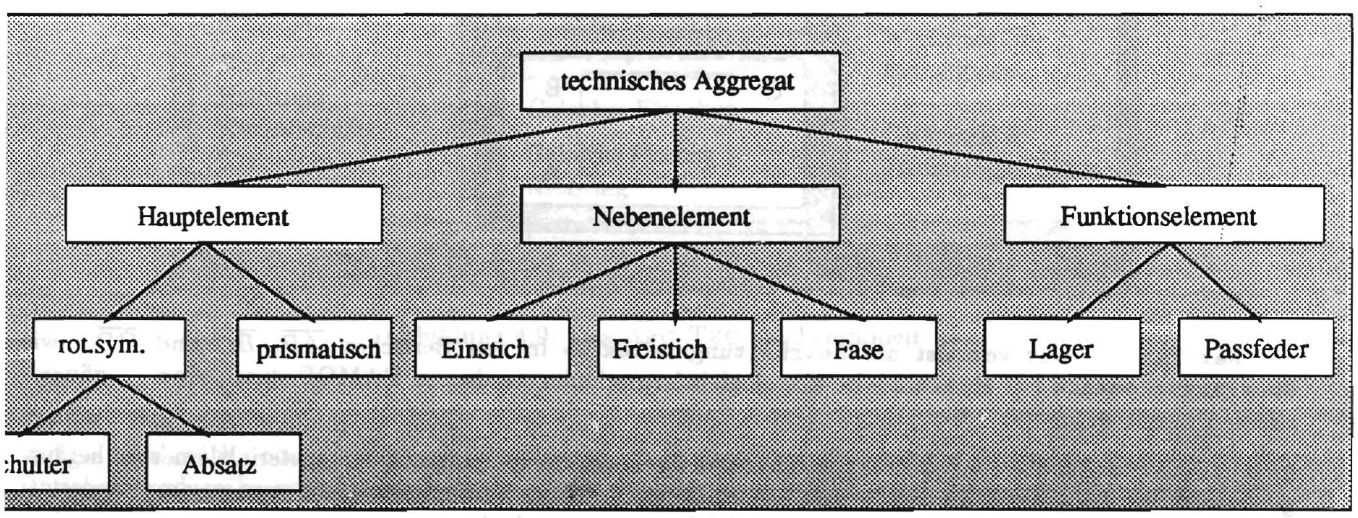


Abbildung 4.6: Das Werkstückmodell von TechMo (Ausschnitt)

BAMOF Am CCK (CIM Center Kaiserslautern) wird ebenfalls an einem integrierten Produktmodell namens BAMOF[16] gearbeitet, von dem im folgenden die *formelementbasierte Werkstückrepräsentation* [Hum91] kurz vorgestellt werden soll.

Ein Werkstück wird dort mit der in Bild 4.7 wiedergegebenen Aggregationshierarchie repräsentiert:

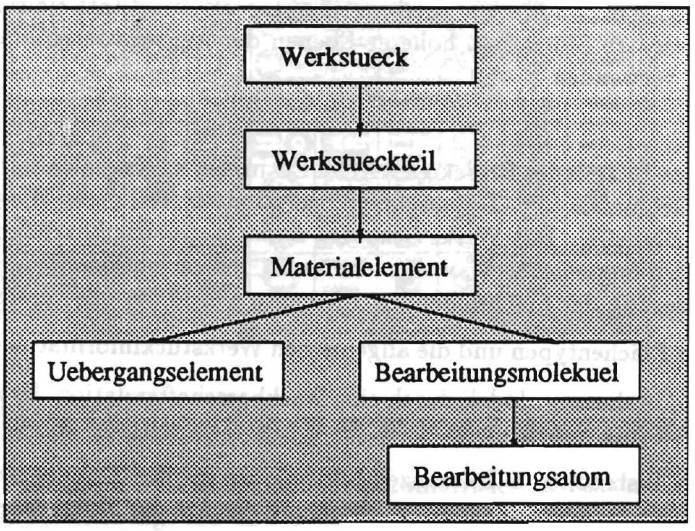


Abbildung 4.7: Die formelementbasierte Werkstückrepräsentation von BAMOF

Ein **Werkstück** enthält neben geometrischen, fertigungstechnischen und technologischen Angaben Verweise auf bis zu 3 **Werkstückteile**: links, horizontal, rechts.

[16] Bauteilmodell FBK/CCK

Werkstückteile zeigen auf eine Reihe von **Materialelemente**. **Materialelemente** sind additive Formelemente, die in ihrer Summe eine Umhüllende des Fertigteils bilden. Sie entsprechen in etwa den Hauptelementen von TechMo. Es gibt 4 verschiedene Typen von **Materialelementen**: Kegel, Zylinder, Konvex, Konkav. Eine geometrische Beschreibung erfolgt durch Angabe einer Reihe von Parametern für den in Bild 4.8 wiedergegenen Streckenzug $\overline{ABCD}$.

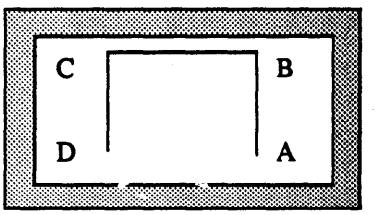


Abbildung 4.8: Das Schema für ein BAMOF-Materialelement

Ein **Materialelement** verweist auf **Bearbeitungsmoleküle** in den Bereichen $\overline{AB}$, $\overline{BC}$ und $\overline{CD}$ sowie **Übergangselemente** bei B, C und D. Topologieinformationen werden in BAMOF durch eine Vorgängerrelation repräsentiert.

Bearbeitungsmoleküle sind subtraktive Elemente die durch Zerspanung aus den Materialelementen herausgenommen werden sollen. Sie sind wiederum aus einer Reihe von **Bearbeitungsatomen** zusammengesetzt. Bearbeitungsmoleküle sind genau einem Materialelement zugeordnet.

Bearbeitungsatomen ist ein ähnliches Schema zugrundegelegt wie den Materialelementen. Damit wird die Geometrie des Atoms beschrieben.

Übergangselemente beschreiben Übergänge von einem Formelement zu einem anderen oder innerhalb von Formelementen. Es gibt 2 Arten von Übergangselementen: Fasen und Rundungen.

Kritik Insgesamt erscheint das Modell zu starr. Ein Werkstück kann aus maximal 3 Teilen bestehen, es gibt genau 4 Typen von Materialelementen etc. Die Realisierung ist in einigen Teilen unnatürlich, z. B. die Kodierung der verschiedenen Materialelemente in einen Streckenzug statt einer Subklassenbildung. Slotwerte unterer Ebenen werden zu Slotwerten höherer Ebenen der Aggregationshierarchie durchgereicht statt generische Funktionen zu verwenden.

ARC-TEC Im Rahmen des R-Teils des Projektes ARC-TEC im DFKI Kaiserslautern wurde ein Formalismus zur Repräsentation von Geometrie und Technologieinformation als Teil eines Wissenbasierten Produktmodells entwickelt [BKL90, BKL90a]. Das Werkstück wird durch die es begrenzenden Oberflächen dargestellt und mit weiteren Attributen angereichert[17], wobei eine an STEP-Konventionen angelehnte Syntax verwendet wird. Auf eine Hierarchisierung der beschreibenden Elemente wurde bewußt verzichtet.

4.9 zeigt die verschiedenen Flächentypen und die allgemeinen Werkstückinformationen dieser Repräsentation.

Topologieinformation wird in diesem Modell durch eine Nachbarschaftsrelation dargestellt, die angibt welche Flächen eine gemeinsame Kante haben.

Technologieinformationen, insbesondere Toleranzen bilden einen großen Teil des beschriebenen Modells, auf eine Beschreibung wird an dieser Stelle jedoch verzichtet, auf die Art der Toleranzrepräsentation wird jedoch in Abschnitt 4.5.1 zurückgekommen.

Kritik Die gewählte BREP-Repräsentation erscheint ungeeignet, um darauf direkt die Arbeitsplanung aufzusetzen, höchstwahrscheinlich ist es auch gar nicht intendiert.

[17] also eine erweiterte BREP-Beschreibung

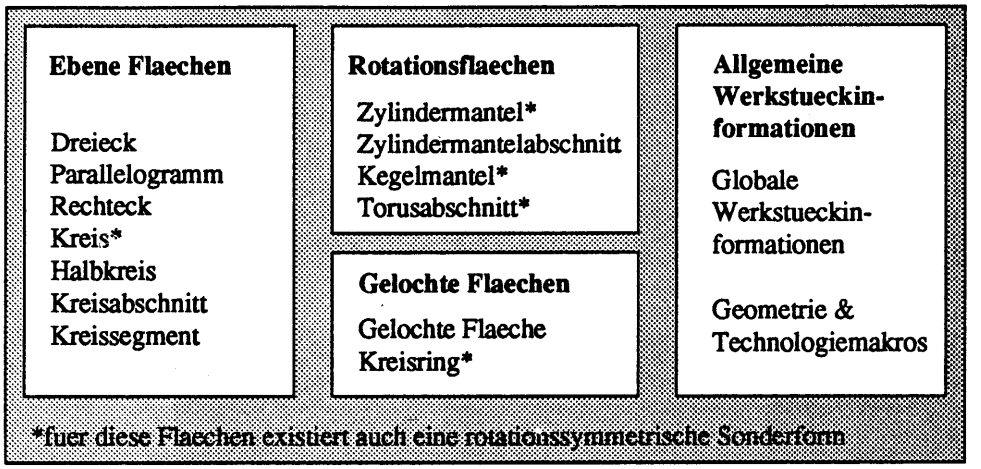


Abbildung 4.9: Das ARC-TEC Flächenmodell

GPPS Das im Rahmen dieser Arbeit vorgestellte Arbeitsplanungssystem GPPS [PS89] benutzt ein mehrstufiges Werkstückmodell, in dem in einer niedrigeren Ebene das Teil mit seiner vollen Geometriedaten beschrieben wird, während auf einer höheren Ebene die Topologie des Teils mit Hilfe der Relationen `left-of` und `carries` beschrieben wird.

Die Teilebeschreibung erfolgt mit Hilfe eines speziellen Entwurfssystems unter Zuhilfenahme der in Abb. 4.10 dargesstellten Featurematrix.

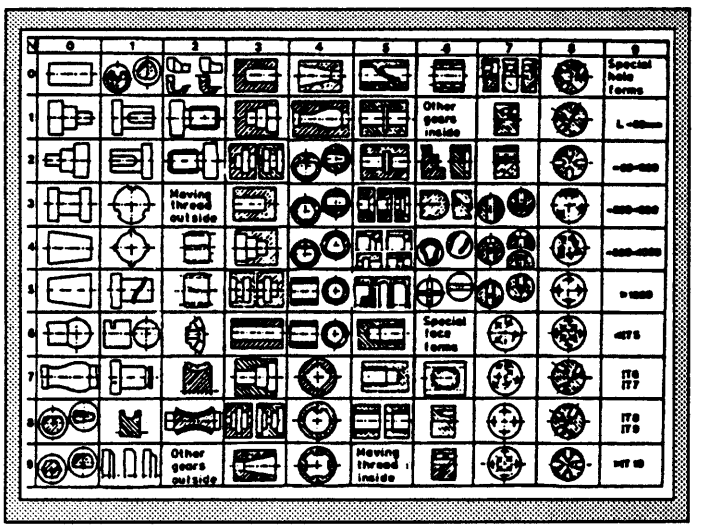


Abbildung 4.10: Die Matrix der geometrischen Features von GPPS

Das „Engineerungmodel" genannte Modell umfaßt die parametrisierten Featurebeschreibungen, die topologische Struktur des Teils, Toleranzangaben und Oberflächeneigenschaften. Seine Struktur ist in Abb. 4.11 wiedergegeben. Die Features selbst sind in verschiedenen Klassen unterteilt: externe Features (entsprechend den Hauptelementen), interne Features von links und interne Features von rechts (für Innenbearbeitung). Hinzu kommen rotations- bzw. nichrotationssymmetrische Abweichungen der obigen Klassen und Bohrungen.

Aus diesen Informationen wird der Topologiegraph für ein konkretes Teil entwickelt, der wiederum die Grundlage der Arbeitsplanung bildet.

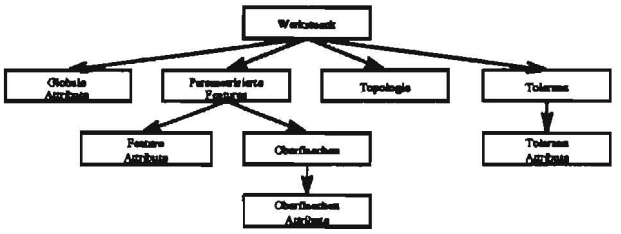


Abbildung 4.11: Das 'Engineeringmodel' von GPPS

Zusammenfassung Insgesamt läßt sich die bei den untersuchten Systemen (auch bei den in Kapitel 3 verglichenen) folgendes feststellen:

- Bei den meisten Systemen ist eine Trennung in Haupt- und Nebenelemente (auch materialhaltige bzw. materiallose Elemente (siehe auch 3.7)) festzustellen. Oft ist jedoch das Spektrum der Elemente fest vorgegeben. Um eine bessere Anpaßbarkeit an neue Verfahren und spezifische Gegebenheiten zu erreichen sollte es aber möglich sein das Werkstückmodell um beliebige Elemente erweitern zu können. Ein weiterer Grund das Modell flexibel zu halten liegt darin, daß „gelernte" Elemente in das Werkstückmodell eingebracht werden können sollten[18]. [19]
- Einhergehend mit dem Einsatz von Objekten hat sich eine hierarchische Darstellung der Aggregation des Werkstücks durchgesetzt, wobei allerdings der Grad variiert. Eine Festlegung der Hierarchietiefe sollte allerdings nicht erfolgen.
- Eine Beschreibung der räumlichen Beziehungen der Werkstückelemente erfolgt meist nur auf der Geometrischen Ebene (d.h. auf Basis der exakten Abmessungen). Die Topologie des Werkstücks wird nur rudimentär beschrieben. Hier wäre eine Erweiterung der Topologiebeschreibung sinnvoll.

Diese Erfahrungen flossen in den Entwurf des Werkstückmodells von TUPPSY ein, der nun vorgestellt werden soll.

Spezialisierungs- und Aggreationshierarchie des Werkstückmodells

Ähnlich wie in TechMo und anderen Systemen entsteht das Werkstückmodell von TUPPSY durch eine Überlagerung einer Spezialisierungs- und einer Aggregationshierarchie. Die Spezialisierungshierarchie wird implizit durch die Objekthierarchie realisiert, während die Aggregationshierarchie explizit durch den `has-parts` Slot beschrieben wird, der die Menge der Werkstückteile beschreibt aus denen das Werkstück besteht. Werkstückteile selbst werden ebenfalls durch eine Spezialisierungshierarchie definiert.

Die Werkstückspezialisierung erfolgt im wesentlichen über eine Einschränkung des has-parts Slots, kann aber auch anderweitig geschehen (zusätzliche Slots, Einschränkung anderer Slots, ...)[20]

Konzeptdefinition(Werkstück) (Teil A)

```
(ist!  (ein werkstück) (ein produktmodellteil
       ;;; allgemeine Werkstückdaten
       : 21
       'has-parts ...)
       (plaene ...)))
```

[18] Sei es manuell oder automatisch

[19] Besonderes Interesse erlangt die Flexibilität des Werkstückmodells im Zusammenhang mit der Verknüpfung von Elementen mit Skelettplänen

[20] z. B. bei der Unterscheidung Welle ↔ Scheibe über das Verhältnis von Länge und maximalen Radius

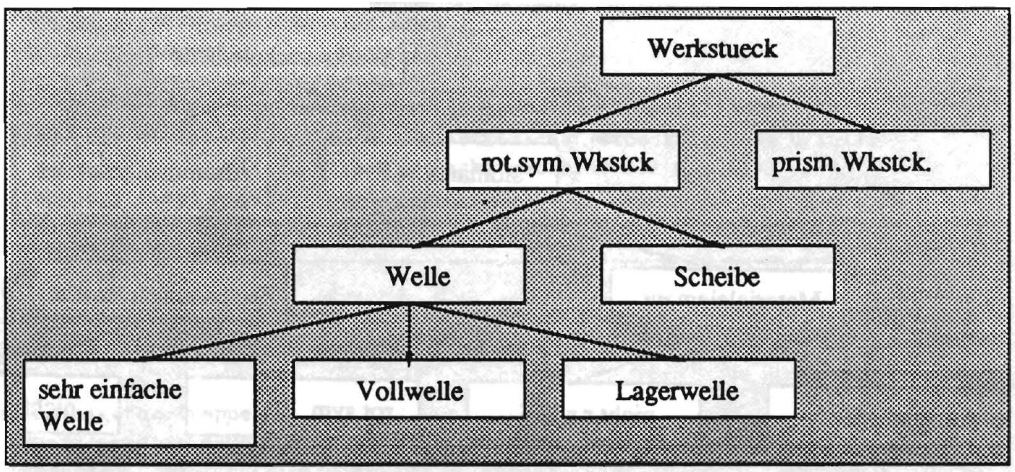


Abbildung 4.12: Ein Ausschnitt aus der Werkstückspezialisierungshierarchie

Anmerkung: Hier wurde wie im ganzen Produktmodell versucht, die Anzahl der Slots einer Konzeptdefinition so gering als möglich zu halten, um so die Datenredundanz zu verringern[22]. Statt neuer Slots werden in dem hier vorgestellten Modell generische Funktionen (bzw. Methoden) benutzt.

So unterscheidet sich z. B. die Definition des `has-parts` Slots eines `Werkstücks`:
`(has-parts (:set #[(ein Werkstückelement 0 inf])))` von der einer `sehr-einfachen-Welle`[23]:
`(has-parts (:set #[(ein Zylinder) 1 3]))` einerseits durch eine Spezialisierung der zugelassenen Werkstückelemente, andererseits durch eine Einschränkung der Elementanzahl.[24]

Um die im letzten Abschnitt geforderte Flexibilität bzgl. der Erweiterung des Werkstückmodells zu erreichen, sind die Spezialisierungshierarchien für Werkstück und Werkstückteile offen. Für die Arbeitsplanung ist hier insbesondere die in (Abb.4.13) gezeigte Struktur der Werkstückelementspezialisierungshierarchie wichtig.

Grundsätzlich gibt es 2 Arten von Werkstückelementen:

- *atomare Werkstückelemente*
- *zusammengesetzte Werkstückelemente.*

Atomare Werkstückelemente Atomare Werkstückelemente entsprechen in etwa der in BAMOF gezeigten Hierarchie von Material- und Bearbeitungsmolekülen, die zusammengesetzte Werkstückelementen dienen den „dynamischen" Aufbau einer Sammlung von Features entsprechend der MGF von GPPS und den Funktionselementen von TechMo.

Atomare Elemente stellen also die Grundbausteine zur Verfügung aus denen das Werkstück und die zusammengesetzten Werkstückelemente aufgebaut werden. In ihnen werden die zu ihrer Beschreibung notwendigen geometrischen und technologischen Angaben gemacht. Darüber hinaus verfügen sie über einen Verweis auf mögliche Arbeitspläne die zu ihrer Fertigung benutzt werden können.

Konzeptdefinition(Werkstückelement):

[22] Dies geschieht im Gegensatz zu einigen der verglichenen Werkstückmodelle wo zum Beispiel der maximale Radius des Werkstücks von unten durch die Aggregationshierarchie propagiert wird und in jeder Ebene als Slotwert festgehalten wird.

[23] Die Einführung dieser Werkstückklasse erfolgt ausschließlich aus didaktischen Gründen, sie hat keine Entsprechung in der realen Welt

[24] Entsprechend etwa der value-restriction und number-restriction in KL-ONE.

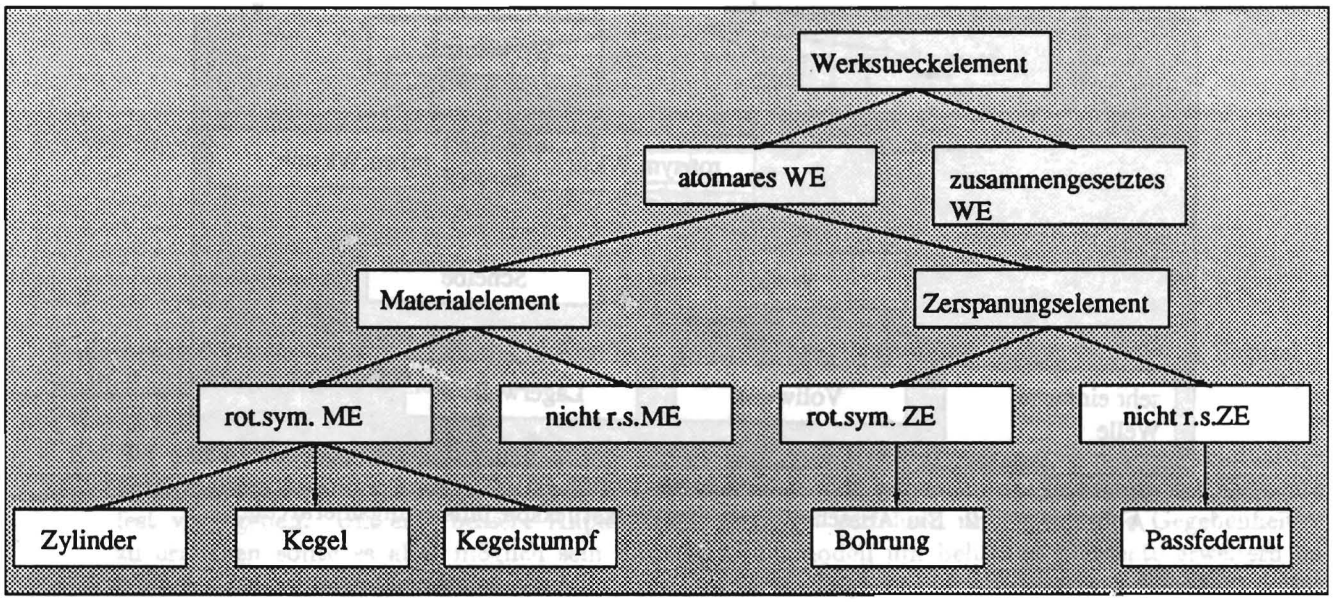


Abbildung 4.13: Spezialisierungshierarchie der Werkstückelemente(Initialzustand)

```
(ist! (ein Werkstueckelement)
      (ein Werkstueckmodellteil
           (Plaene (:set #[(ein Arbeitsplan) 0 inf]))
   (Flaechen (:set #[(ein Wort) 0 inf]))
           (Graphik {0 1} (br-fcn 'br-fcn-graphik))))
```

Klassendefinition(Zylinder):

```
(ist! (ein Zylinder)
      (ein Rot-Sym-Materialelement
        ;;; vererbte Slotwerte
                (Laenge [1mm 1m])
                (Volumen [1mm3 1m3])
                (Offset [0mm 1m])
                (Plaene ~Zylinder-Plan-1)
        (Flaechen (:set 'links 'mantel 'rechts))
        ;;; eigene Slotwerte
        (Radius [1mm 1m])))
```

Anmerkung: ~ soll hier und im weiteren Anzeigen, daß es sich um eine Instanz handelt.

Zusammengesetzte Werkstückelemente Zusammengesetzte Elemente können nicht nur aus atomaren Elementen sondern auch aus weiteren zusammengesetzten Elementen bestehen. Damit können beliebige Aggreationshierarchien und so auch neue Featurearten definiert werden.

Klassendefinition(Zusammengesetztes Werkstückelement):

```
(ist! (ein Zusammengesetztes-Werkstueckelement)
      (ein Werkstueckelement
           (part-of (:or (ein Werkstueck)
                         (ein Zusammengesetztes-werkstueckelement)))
           (has-parts (:set #[(ein Zusammengesetztes-werkstueckelement) 0 inf]
                            #[(ein Atomares-werkstueckelement) 0 inf]))
           (topology (:set #[(eine Topologiebeschreibung) 0 inf]))))
```

Klassendefinition(Ansteigende Treppe):

```
(ist! (eine Ansteigende-Treppe)
      (ein Zusammengestztes-Werkstueckelement
           (has-parts (:set #[(ein Zylinder) 0 inf]))))
```

Anmerkung: Wie man sieht ist die Tatsache, daß es sich um eine ansteigende Treppe handelt nicht aus der Klassendefinition ersichtlich. Diese Tatsache wird durch ein entsprechendes Constraint formuliert, etwa:

```
(constrain ((#?w (eine Ansteigende-Treppe)
           (#?c1 (part-of #?w))
           (#?c2 (part-of #?w))
           (equal (topology #?w #?c1 #?c2) 'meets)))
    (<= (#?c1 radius) (#?c2 radius)))
```

Wesentlich sind im Zusammenhang mit zusammengesetzten Werkstückelementen:

1. Die gleiche Funktionalität wie bei atomaren Werkstückelementen wird durch Realisierung entsprechender Methoden erreicht. Dies wird in der Regel die Anwendung einer Funktion auf die Ergebnisse der Methodenaufrufe auf die Teile sein, z.B.: Bestimmung des maximalen Radius als Maximum der maximalen Radii der Teile, oder zulässige Arbeitspläne als Vereinigung der zulässigen Arbeitspläne mit dem Kreuzprodukt der Arbeitspläne der Teile)

2. Durch die realisierte Aggregationshierarchie verschwimmen die Grenzen zwischen einem Werkstück und einem zusammengesetzten Werkstückelement. So kann es Werkstücke geben, die als einziges Teil ein zusammengesetztes Werkstückelement haben[25]. Dies ist jedoch nicht die Intention dieser Unterscheidung. Werkstückelemente sind die Bausteine aus denen Werkstücke zusammengesetzt werden.

3. Über die Definition zusammengesetzter Werkstücke wird die obige Forderung nach Darstellung des Werkstücks auf mehreren abstrakten Ebenen erfüllt. Und zwar auf folgende Art und Weise:[26]

   Im Slot `Topologie` wird die Topologie eines zusammengesetzten Werkstücks als Graph von Allen-Intervallen dargestellt [All83][27]. Grundsätzlich sind nun 2 Arten der topologischen Beschreibung denkbar, je nachdem ob 1 oder 2 Achsen eines rotationssymmetrischen Werkstücks beschrieben werden sollen. Die Erweiterung der Allen-Beschreibung in höhere Dimensionen kann auf kanonische Weise durch Tupelbildung erfolgen.[28]. Abbildung 4.14 zeigt beide Möglichkeiten anhand eines Beispielwerkstücks.

[25] Dies kann sogar in der Definition eines Werkstücks gefordert werden.

[26] Einschränkung auf rotationssymmetrische Werkstücke

[27] Allen-Intervalle wurden ursprünglich für die Beschreibung von Zeitintervallen benutzt, sind aber natürlich für alle Intervallarten geeignet

[28] Zu diesem Thema siehe auch [Gü89]

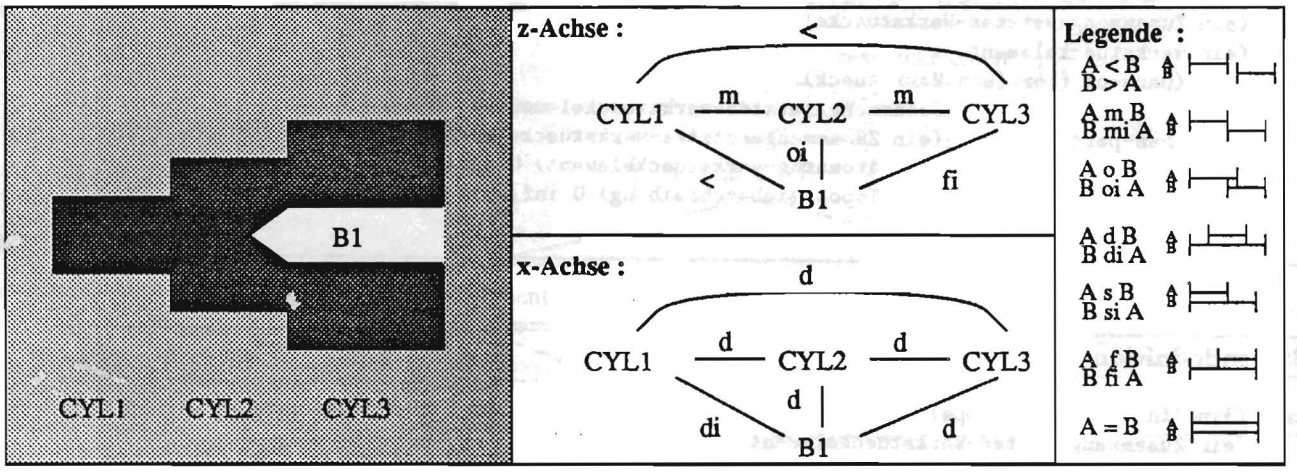


Abbildung 4.14: 1(2)-Dimensionaler Topologiegraph eines Werkstücks

Besteht ein zusammengesetztes Werkstückelement wiederum aus zusammengesetzten Werkstückelementen, so werden nur die topologischen Beziehungen zwischen diesen beschrieben, so daß so eine hierarchische Topologiebeschreibung entsteht.

Welche der beiden Möglichkeiten sollte nun verwendet werden: Die einfache (und schneller zu gewinnende) 1-dimensionale oder die komplexere (und evtl. aussagekräftigere) 2-dimensionale Beschreibung oder etwa beide?

Für einfache Teile (ohne oder mit wenig Innenbearbeitung) reicht eine 1-dimensionale Beschreibung aus, für komplexere Teile gilt dies nicht, die Relation auf der x-Achse wird aber

(a) Schon durch die Termini wie Innenbearbeitung etc. beschrieben, und ist

(b) in der Regel einfacher als die Relation auf der z-Achse.

Anmerkung: Die obige Feststellung gilt so nur für Wellen, weniger für Scheiben. Dort ist es aber umgekehrt: Die Schwerpunkte liegen hier auf der Innenbearbeitung und der x-Achse.

Dies führt in dem hier betrachteten Problembereich also zu einer 1-dimensionalen Topologiebeschreibung der z-Achsenrelation. Die so gewonnene Topologiebeschreibung ist im Verhältnis zu den bisher betrachteten expliziter und umfangreicher und dennoch ebenso einfach zu gewinnen. Sie kann als Grundlage für die Klassifikation eines Werkstücks dienen. So ist zum Beispiel der Topologiegraph für die beiden Werkstücke in Abb. 4.15 identisch, die Arbeitspläne sind zumindest ähnlich.[29]

Die Topologie eines Teiles kann aber nicht immer alleinige Grundlage für die Auswahl eines Arbeitsplanes sein, dies wird im Arbeitsplanmodell (4.5.3) berücksichtigt. Andererseits können auch verschiedene Topologien zu einem Arbeitsplan führen, dies wird durch die Möglichkeit mehrere Topologiebeschreibungen im `topology`-Slot eines zusammengestzten Werkstücks berücksichtigt.

Die Definition eines zusammengestzten Werkstückelementes erfolgt also durch Angabe von Aggregations- und Topologiestruktur, evtl. noch durch Angabe von Constraints.

[29] Diese Beobachtung ist zum Beispiel bei GPPS Grundlage der Arbeitsplanerzeugung

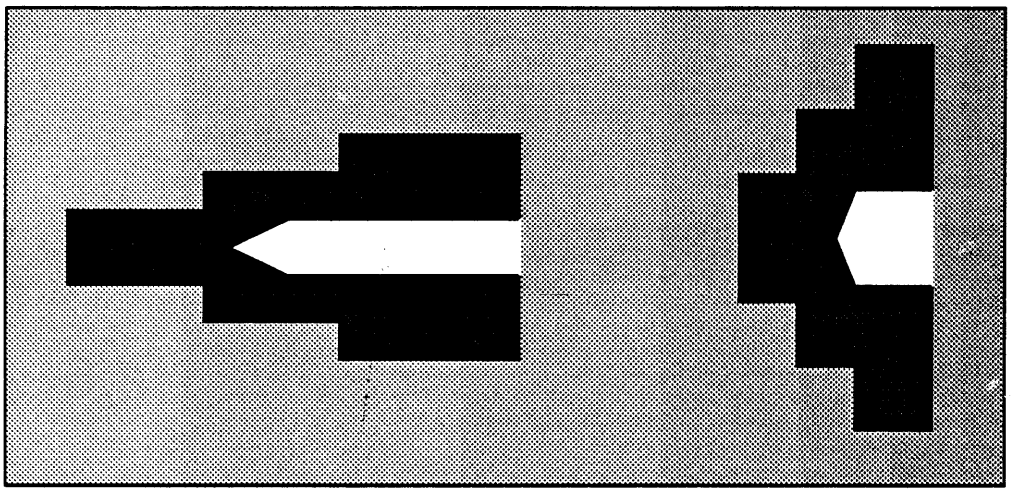


Abbildung 4.15: Ähnliche Arbeitspläne bei gleicher Topologie

Allgemeine Werkstückdaten

Folgende allgemeine Daten werden an einem Werkstück festgehalten.

Konzeptdefinition(Werkstueck): (Teil B)

```
;; allgemeine Werkst"uckangaben
   (werkstoff (ein Werkstoff))          ;; ein Werkstoff aus dem Werkstoffmodell
   (name (ein Wort))                    ;; Name des Werkst"ucks
   (zeichnung (ein Wort))               ;; Verweis auf die CAD-Zeichnung
   (gewicht [1g 10t])                   ;; das Gewicht des Werkst"ucks
   (default-toleranz (eine Toleranz))   ;; die Defaulttoleranz f"ur nicht tolerierte Ma\3e
   (default-oberflaechenguete (eine Oberflaechenguete))
                                        ;; die Default-Oberfl"acheng"ute f"ur nicht gekennzeichnete Fl"ache
   (haerte (eine Haerte))               ;; Ben"otigte Werkst"uckh"arte
```

Technologiedaten und Technologiemodell

Werkstückelemente enthalten einen Verweis auf technologische Anforderungen an dieses Element. Technologische Anforderungen, sind hier Toleranzen, Oberflächengüte und Härteangaben.

Konzeptdefinition(Werkstueckelement): (Technologische Angaben)

```
;;; technologische Daten
(oberflaechengueten (:set #[(eine Oberflaechenguete) 0 inf]))
(toleranzen (:set #[(eine Toleranz) 0 inf]))
(waermebehandlungen (:set #[(eine waermebehandlung) 0 inf]))
(haerten (:set #[(eine haerte) 0 inf]))
```

Die Technologiedaten eines Werkstücks werden mit Hilfe von Methoden aus den Technologiedaten der Elemente gewonnen, während Toleranzen etc. selbst im Technologiemodell beschrieben werden.

Toleranzangaben Toleranzen beziehen sich immer auf Flächen der Geometrieebene. Daher müssen Funktionen zur deren Benennung und Ermittlung bereitgestellt werden. Dies geschieht über den Slot `Flaechen` in der Klasse Werkzeugelement. Er enthält eine Liste von Flächennamen des Werkstückelements. Erwünscht wäre weiterhin eine Methode die für ein Werkstückelement und einen Flächennamen eine entsprechende Flächenrepräsentation liefert. Inwieweit dies für allgemeine Werkstückelemente realisierbar ist, müßte noch untersucht werden.

Als Beispiel für eine Toleranz soll hier die Klasse `Passung` vorgestellt werden.

Konzeptdefinition(Passung):

```
(ist! (eine Passung)
      (eine Toleranz
           (Werkstueckelement (ein Werkstueckelement))
           (Tolerierte-Flaeche (ein Wort))
           (Innenpassmass (ein Passmass))
           (Aussenpassmass (ein Passmass))))

(ist! (ein Passmass)
      (ein Mass
           (Toleranzfeldlage (ein Wort))
           (Toleranzklasse (ein Zahlwert))))
```

Oberflächengüten Auch Oberflächengüten beziehen sich auf Flächen die in Werkstückelementen benannt sind. Daher ist die Repräsentation ähnlich . Als Grundlage für die Repräsentation dient `DIN/ISO 1302`. In den Unterklassen von `Oberflächengüte` werden dann die eigentlichen Maßgrößen eingetragen.

Konzeptdefinition(Oberflächengüte):

```
(ist! (eine Oberflaechenguete)
      (ein Technologieelement
           (Werkstueckelement (ein Werkstueckelement))
           (Bezugsflaeche (ein Wort))))

(ist! (ein Mittenrauhwert)
      (eine Oberflaechenguete
           (Rauheitswert [0mm 1mm])))
```

Ein Problem bei der Angabe von Oberflächengüten ist das Fehlen einer allgemeinen Umrechnungsformel zwischen den verschiedenen Meßwerten. Dies muß in den Teilen des Arbeitsplanungssystems, in denen auf sie referiert wird berücksichtigt werden.

Härten Wie Toleranzen und Oberflächengüten beziehen sich auch Härteangaben auf Flächen, eine Umrechnung zwischen den verschiedenen Härteangaben ist ebenfalls nicht möglich.[30] Gekennzeichnet werden Härten meist durch ein Kurzzeichen.

Konzeptdefinition(Vickershärte):

[30] Nur in geringerm Maße gemäß der Umwertungstabelle nach DIN 50150

```
(ist! (eine haerte)
      (ein Technologieelement
   (Kurzzeichen (ein Wort))
   (Werkstueckelement (ein Werkstueckelement))
   (Bezugsflaeche (ein Wort))))

(ist! (eine Vickershaerte)
     (eine Haerte
   (Pruefkraft [0N 100N])
   (Einwirkdauer [0s 100s])))
```

Darstellungsmodell

TUPPSY benutzt zur Darstellung des Werkstücks `IRIT`, einen 3-D CSG-Solid-Modeller, der über die externe Schnittstelle angeschlossen ist[31]. Die Werkstückelemente und Werkstücke verfügen dazu über einen Slot `graphic-representation`, in dem ein Verweis auf das assoziierte `IRIT`-Objekt abgelegt wird.

4.5.2 Das Werkstattmodell

Wie schon oben erwähnt umfaßt das Werkstattmodell die 4 Teilmodelle:

- Werkzeugmodell
- Spannmittelmodell
- Maschinenmodell
- Werkstoffmodell

die wiederum in Untermodelle zerfallen. Je nach Spezialisierung der Teilmodelle kommen evtl. noch Teile hinzu.

Abbildung 4.16 zeigt die Aggregationshierarchie des Werkstattmodells.

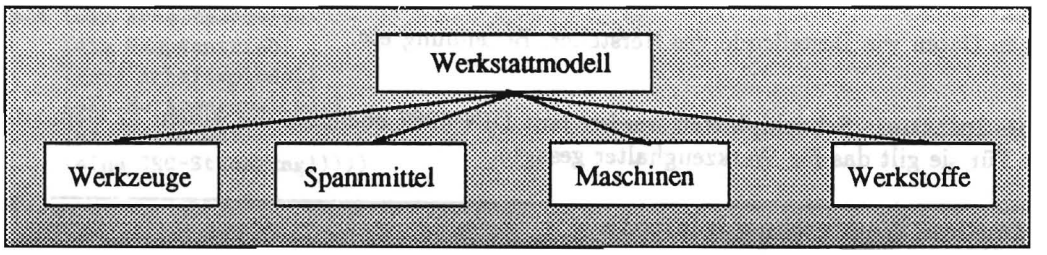


Abbildung 4.16: Die Aggregationshierarchie des Werkstattmodells

Werkzeugmodell

Mit der in der Einführung gemachten Einschränkung besteht das Werkzeugmodell aus 2 Teilen: Wendeschneidplatten und Werkzeughalter. Meist werden von den Werkzeugherstellern wie SPK und Hertel sogenannte Werkzeugsysteme angeboten, die geeignete Kombinationen aus Schneidplatte und Halter für bestimmte Bearbeitungsarten darstellen. Daher existiert eine Klasse `Werkzeugsystem` deren Instanzen (bzw.

[31] Näheres zum Anschluß und der Integration in die Objekthierarchie findet man in 5.3

Instanzen ihrer Subklassen) auf die jeweiligen Instanzen der Schneidplatten und Halter zeigen und in denen die Kriterien für die Auswahl dieses speziellen Werkzeugssystems abgelegt werden können.

Die Klassen **Schneidplatte** und **Werkzeughalter** dienen der Repräsentation der von den Herstellern angebotenen Varianten.

Werkzeughalter Werkzeughalter werden laut DIN 4983 durch eine 11-elementigen Zahlen-Buchstabencode beschrieben. Dabei stehen die einzelenen Stellen für Daten wie **Art der Befestigung, Form der Schneidplatte, ...** Diese Daten wurden durch die gewählte Repräsentation explizit gemacht, zusätzlich werden jedoch der DIN-Code sowie Konvertierungsroutinen zwischen den beiden Formaten bereitgestellt.

Konzeptdefinition(Werkzeughalter):

```
(ist! (ein Werkzeughalter)
      (ein Werkzeugteil
           (Kurzzeichen (ein Wort))
           (Benennung (ein Wort))
           (DIN-Hauptnummer (ein Zahlwert))
           (Schneidplattenbefestigung
              {'|von oben geklemmt|
               '|von oben und ueber Bohrung geklemmt|
               '|ueber Bohrung geklemmt|
               '|durch Bohrung aufgeschraubt|})
           (Plattenform {'secheckig 'achteckig 'fuenfeckig 'quadratisch 'dreieckig 'rhombisch-80Grad
                         'rhombisch-55Grad 'rhombisch-75Grad 'rhombisch-86Grad 'rhombisch-35Grad
                         '|dreieckig mit vergroessertem Eckenwinkel| 'rechteckig 'rhomboidisch-85Grad
                         'rhomboidisch-82Grad 'rhomboidisch-55Grad 'rund})
                         (Halterform ...)
                         (Normalfreiwinkel ...)
                         (Ausfuehrung ...)
                         (Hoehe-Schneidenecke ...)
                         (Schaftbreite ...)
                         (Halterlaenge ...)
                         (Plattengroesse ...)))
```

Hinzu kommen Daten wie Bestellcode des Hersteller, Benennung etc.

Wendeschneidplatten Schneidplatten werden laut DIN 4987 ebenfalls mit einem 12 elementigen Code beschrieben. Für sie gilt das für Werkzeughalter gesagte.

Konzeptdefinition(Wendeschneidplatten):

```
(ist! (eine Wendeschneidplatte)
      (ein Werkzeugteil
           (Kurzzeichen (ein Wort))
           (Benennung (ein Wort))
           (DIN-Hauptnummer (ein Zahlwert))
           (Grundform ...)
           (Normalfreiwinkel {'3 '5 '7 '15 '20 '25 '30 '0 '11})
           (Toleranzklasse ...)
           (Spanflaeche ...)
           (Groesse ...)
           (Dicke ...)
           (Schneidenecke ...)
```

```
(Schneide ...)
(Vorschubrichtung ...)
(Schneidstoff ...))))
```

Für die einzelnen Werkzeugsysteme existieren entsprechende Subklassen von **Werkzeughalter** und **Schneidplatte**, die weitere Merkmale definieren können. Die Instanzen der Klassen **Werkzeughalter** und **Schneidplatte** werden über die externe Schnittstelle auf einer Datenbank gehalten und beim Start von TUPPSY geladen. Es erfolgt ein loser Datenbankanschluß im Sinne von 5.4.2

Maschinenmodell

Da TUPPSY der eingeschränkten Aufgabenstellung der Arbeitsplanerstellung für weitgehend rotationssymmetrische Drehteile gerecht werden soll, wurde das Maschinenmodell daran angepaßt. So ist bis jetzt nur der Bereich der Drehmaschinen ausgearbeitet, eine Erweiterung auf CNC-Center ist aber leicht möglich.

Eine **CNC-Drehmaschine** enthält neben technischen Daten wie **Drehzahlbereich, Anschlußwert** und **Gewicht** Verweise auf seine Teile **Arbeitsbereich, Hauptantrieb, Werkzeugsystem, Steuerung, Spindel, Reitstock**

Konzeptdefinition(CNC-Drehmaschine):

```
(ist! (eine CNC-Drehmaschine)
    (eine Drehmaschine
    (Drehzahlbereich [1U/min 10000U/min])
    (Eilgang-X-Achse [1m/min 100m/min])
    (Eilgang-Z-Achse [1m/min 100m/min])
    (Vorschubkraft-X-Achse [1N 100kN])
    (Vorschubkraft-Z-Achse [1N 100kN])
    (Wegmessystem-X-Achse {'linear 'drehwertgeber})
    (Wegmessystem-Z-Achse {'linear 'drehwertgeber})
    (Anschlusswert [1W 100kW])
    (Gewicht [100kg 100t])
    (has-parts
        (:set (ein Arbeitsbereich)
              (ein Hauptantrieb)
              (eine Arbeitspindel)
              (ein Werkzeugspeicher)
              (ein Reitstock)
              (eine CNC-Steuerung)))))
```

Arbeitsbereich, Hauptantrieb, Slots enthalten die spezifischen technischen Daten wie z.B. **Werkzeuganzahl** und **Revolverkopftyp** bei **Werkzeugsystem**

Spannmittelmodell

Spannmittel können in Hauptspannmittel und Nebenspannmittel unterschieden werden. Bei den Hauptspannmittel können außen, innen und axialspannende unterschieden werden, während bei den Nebenspannmitteln zwischen Reitstock, Zentrierspitze und Linette differenziert werden muß. Abb. 4.17 zeigt die Spezialisierungshierarchie des Spannmittelmodells.

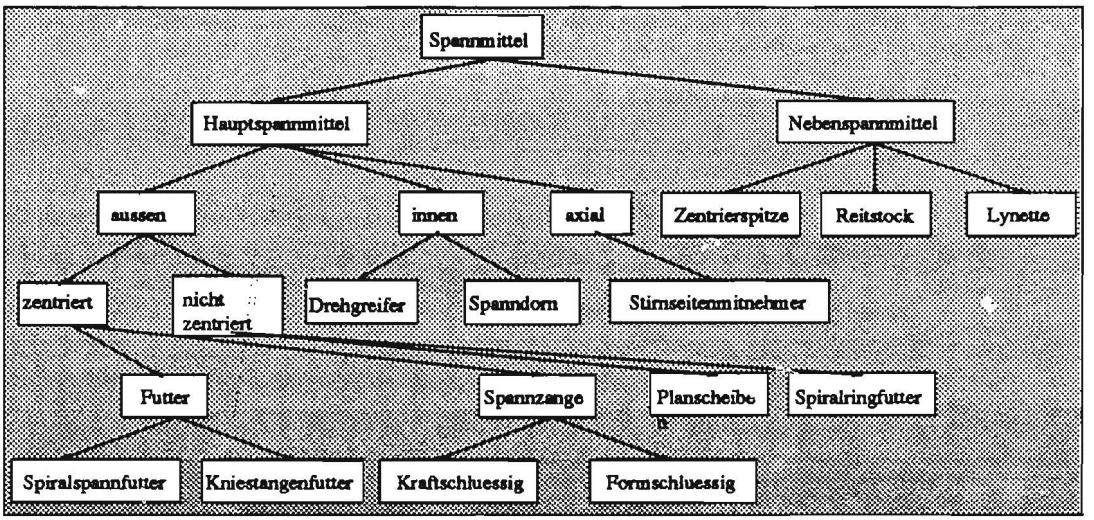


Abbildung 4.17: Die Spannmittelhierarchie

4.5.3 Das Arbeitsplanmodell

Wie wir in Abschnitt 4.5.1 gesehen haben, können Arbeitspläne mit Werkstücken und Werkstückelementen assoziert w rden. Dies dient den folgenden Zwecken:

- Die Assoziation mit einem Werkstück dient dazu, während des Konstruktionsprozesses den sich langsam entwickelnden Arbeitsplan zu speichern.
- Die Kopplung mit Werkstückelementen dient dazu, Standardpläne (Skelettpläne) für die Fertigung dieser Werkstückelemente abzulegen. Aus diesen Standardplänen können dann Arbeitspläne für komplexere Werkstücke und Werkstückelemente kombiniert und entwickelt werden.

Wesentliches Element eines Arbeitsplanes ist die Arbeitsvorgangsfolge, die aus einer Reihe von Arbeitsvorgängen besteht[32]. Zusätzlich enthält ein Arbeitsplan neben einem Identifier Verweise auf das Werkstück(bzw. das Werkstückelement), die Maschine, die verwendeten Werkzeuge und die benutzten Spannmittel.

Klassendefinition(Arbeitsplan):

```
(ist! (ein Arbeitsplan)
      (ein Arbeitsplanmodellteil
           (Name (ein Wort))
           (Werkstueck (:or (ein Werkstueck) (ein Werkstueckelement)))
           (Maschine (eine Werkzeugmaschine))
           (Einrichteblaetter (:set #[(ein Einrichteblatt) 1 2]))
           (Arbeitsvorgangsfolge (:sequence #[(ein Arbeitsvorgang) 0 inf]))))
```

Der Slot `Einrichteblaetter` enthält alle Informationen über die benötigten Spannmittel, Werkzeuge und Aufspannungen. Für jede Aufspannung wird ein Einrichteblatt gehalten, dessen Werte durch die Module Aufspannungsermittlung und Werkzeugbestimmung ermittelt werden. Ein `Einrichteblatt` enthält neben globalen Informationen wie `Werkstueck` und `Maschine` technische Daten zur Spannung wie `Einrichtemass`, `Spanndurchmesser`, `Hauptspannmittel` die in dieser Aufspannung benötigten Werkzeuge im Slot `Werkzeugbestueckung`.

[32] Da die Herstellbarkeit des Werkstücks auf einer Maschine vorausgesetzt wurde, entsprechen die Arbeitsvorgänge in et a den Teilarbeitsvorgängen aus Kapitel 2.

Klassendefinition(Einrichteblatt):

```
(ist! (ein Einrichteblatt)
      (ein Arbeitsplanmodellteil
           (Werkstueck(ein Werkstueck))
           (Aufspannung {1 2})
           (Einrichtemass [0mm 1000mm])
           (Rohmass [0mm 1000mm])
           (Spanndurchmesser [0mm 1000mm])
           (Spannlaenge [0mm 1000mm])
           (Spanndruckfutter [0N 10KN])
           (Spanndruck-Pinole [0N 10KN])
           (Hauptspannmittel (ein Hauptspannmittel))
           (Zentrierspitze (eine Zentrierspitze))
           (Lynette (eine Lynette))
           (Werkzeugbestueckung (:set #[(eine Werkzeugbestueckung) 0 inf]))))
```

Eine **Werkzeugbestückung** ist hier die Zuordnung eines Werkzeuges zu einer Position im Werkzeugspeicher der Maschine[33].

Bei den Arbeitsvorgängen wird ähnlich wie bei den Werkstückelementen vorgegangen; auch hier wird zwischen atomaren und zusammengesetzten Arbeitsvorgängen unterschieden. Der Grund hierfür liegt in der Verwendung in Standardarbeitsplänen. Während bei den anderen Slots eines Arbeitsplanes die Diskriminierung über eine Spezialisierung des Slotwertes ausreicht, soll für die Arbeitsvorgangsfolge eine größere Ausdrucksmächtigkeit erzielt werden, um etwa die folgenden Sachverhalte auszudrücken:

- Ein Schlichtschritt muß nur dann ausgeführt werden, wenn die geforderte Oberflächengüte dies verlangt.
- Wenn A zutrifft, dann führe die Schritte X,Y und Z aus

Diese Ziele werden durch die Kopplung einer Bedingung mit einem Arbeitsvorgang erreicht. Der erste Sachverhalt wird dann mit einem atomaren Arbeitsvorgang dargestellt, der zweite durch einen zusammengesetzten Arbeitsvorgang mit den 3 Teilvorgängen X,Y,Z. Als Bedingung können beliebige Pattern benutzt werden, d. h. Konjunktionen von Objekt-Slot-Wertbeschreibungen (wie sie auch bei der Definition von Konzeptuellen Constraints benutzt werden)[34]

Ein **Arbeitsvorgang** besteht dann aus einer Bedingung, einem Bearbeitungsverfahren und einem Bereich auf den das Bearbeitungsverfahren angewandt werden soll. Dieser Bereich kann durch ein Werkstückelement oder eine Polygonbeschreibung der Querschnittsfläche (bei rotationssymmetrischen Elementen) charakterisiert sein.

Klassendefinition(Atomarer Arbeitsvorgang):

```
(ist! (ein Atomarer-Arbeitsvorgang)
      (ein Arbeitsplanmodellteil
           (Bedingung (ein Pattern))
           (Bearbeitungsverfahren (ein Bearbeitungsverfahren))
           (Bereich (:or (ein Werkstueckelement) (eine Querschnittsflaeche)))))
```

[33] Die Begrenzung auf die Zahl der Plätze des Werkzeugspeichers erfolgt über ein Constraint.

[34] Es wäre auch eine Realisierung der Bedingungen als `Exist` bzw. `Not-Exist` Constraints mit den entsprechenden Pattern und Arbeitsvorgängen denkbar.

Die zweite Form der Bereichsangabe wird für die nicht Standardarbeitsplangestützte Form der Arbeitsvorgangsfolgenermittlung benötigt, bei der die einzelnen Arbeitsvorgänge durch Aufteilung der Differenzquerschnittsfläche zwischen Roh- und Fertigteil bestimmt werden. **Bearbeitungsverfahren** sind zum Beispiel Plandrehen, Längsdrehen etc.

4.6 Erzeugung der internen Werkstückrepräsentation

Die Erzeugung der internen Werkstückrepräsentation wird als eigenständiger Konstruktionsprozeß betrachtet, dessen Ausgabe die Eingabe des eigentlichen Arbeitsplanungsprozesses ist.

In **PLAKON** wird der Konstruktionsprozeß durch die Angabe des zu Konstruierenden Objektes (hier **Werkstueck** oder **sehr-einfache-Welle**) gestartet. Zusätzlich kann die Konstruktion durch Angabe weiterer Informationen wie Teilkonstruktionen, Slotwerte etc. genauer spezifiziert werden[35].

Dies erlaubt die Integration von CAD-Dateien ebenso wie die völlige Neukonstruktion:

- Für jedes zu lesende Dateiformat wird eine Konvertierungsroutine bereitgestellt (als Lisp-Funktion oder durch FRESKO-Objekte. Mit Hilfe dieser Konvertierungsroutine wird eine Initialbeschreibung des zu Konstruierenden Objekts generiert und der Konstruktionsprozeß damit gestartet.
- Der Fall der völligen Neubeschreibung des Werkstücks ist durch den Start des Konstruktionsprozesses mit der leeren Initialbeschreibung abgedeckt.
- Je nach Qualität der Konvertierungsroutinen ist die Initialbeschreibung mehr oder weniger gut. Es kann jedoch mit einer schlechteren Routine gestartet werden, die dann im Laufe der Zeit verbessert wird.
- Der Prozeß der Neubeschreibung kann auf verschiedene Art und Weise ablaufen. Einerseits kann er völlig benutzergesteuert erfolgen, indem in **PLAKON** die Strategie *interaktive Konstruktion* ausgewählt wird. Andererseits kann durch die Eingabe entsprechender zusammengesetzter Werkstückelement und Kontrollwissens ein technischer Modellierer wie TechMo realisiert werden.

4.7 Maschinenauswahl

Dadurch, daß die Problemstellung auf den Bereich der Klein- und Mittelserienfertigung und die Fertigbarkeit auf einer Maschine eingeschränkt wurde, reduziert sich die Maschinenauswahl auf die Selektion aus einer relativ kleinen Menge von Möglichkeiten.

Wichtig sind in diesem Zusammenhang nur die technischen Daten der Maschine, kaufmännische Aspekte werden ebenfalls vernachlässigt. Die Auswahl kann somit im wesentlichen durch eine Menge von Constraints realisiert werden, wie etwa:

```
(constrain ((#?w (ein Werkstueck)
           (#?m (eine Drehmaschine))
           (#?a (ein Arbeitsraum (part-of #?m)))))
    (< (#?w laenge) (#?a laenge)))
```

Ähnliche Constraints können für die erforderliche Leistung (etwa über das Gewicht des Werkstücks) etc. aufgestellt werden.

[35] Also durch Vorwegnahme von Konstruktionsschritten

Ein weiterer Gesichtspunkt für die Maschinenauswahl bildet die Ausführbarkeit bestimmter Bearbeitungsverfahren auf den unterschiedlichen Maschinen, die über **exist** bzw. **not-exist** Constraints gefordert werden können.

4.8 Aufspannungsermittlung

Folgende Schritte werden zur Aufspannungsermittlung vorgenommen:

- Ermittlung der Anzahl der notwendigen Aufspannungen über charakteristische Werkstüeckmerkmale (z. B. beidseitige Gewindeenden, hohe Oberflächengüten)
- Auswertung globaler Constraints zur Bestimmung besonderer Spannmittelanforderungen, wie etwa der Notwendigkeit einer Lynette aus dem Längen/Durchmesserverhältnis.
- Ermittlung der zur Spannung geeigneten Flächen über entsprechende Einträge an den Werkstückelementen (evtl. werden dort auch schon Anforderungen an die benötigten Spannmittel zur Spannung dieser Fläche gestellt).
- Auswahl der Spannflächen durch eine Grobabschätzung der ausführbaren Bearbeitungen in der jeweiligen Aufspannung (z. B wie in KAPLAN)
- Ermittlung der für diese Spannflächen geeigneten Spannmittel, unter Beachtung von Constraints wie etwa notwendige und maximal zulässige Spannkraft, Oberflächengüte etc.

4.9 Arbeitsvorgangsfolgenbestimmung

Die Arbeitsvorgangsfolgenbestimmung umfaßt die Bestimmung von Fertigungsbereichen die mit einem Werkzeug gefertigt werden können. Prizipiell sind hierzu zwei Vorgehensweisen möglich:

- Rekursive Zerlegung des gesamten zu zerspanenden Volumens in kleinere Bereiche, bis diese mit einem Werkzeug fertigbar sind (Top-Down Verfahren)
- Zusammenfassung elementarer Fertigungsbereiche (Bottom-Up Verfahren)

Beide Verfahren haben ihre Vorteile, in TUPPSY wird eine Integration beider angestrebt:

Die Fertigungsverfahren für Nebenelemente, wie etwa Paßfedernuten sind relativ eindeutig, hier können die dort abgelegten Standardarbeitspläne benutzt werden.

Besteht ein Werkstück aus einer relativ geringen Zahl von Hauptelementen und zusammengesetzten Werkstückelementen, so wird angenommen, daß eine Kombination der dort gespeicherten Standardarbeitspläne ein gutes Ergebnis liefert und die Arbeitsvorgangsfolge so generiert.

Sollte den Benutzer der so generierte Arbeitsplan nicht zufriedenstellen, so kann er ihn mit einem Editor verändern und evtl. neue zusammengesetzt Werkstückelemente mit ihm bzw. Teilen von Ihm assoziieren.

Bei einer großen Anzahl von Hauptelementen (zusammengesetzten Werkstückelementen)[36] wird folgendes Verfahren vorgeschlagen:

- Bestimmung des zu zerspanenden Volumens und Beschreibung seiner Topologie[37].

[36] oder parallel zur obigen Vorgehensweise

[37] Es werden hier nur rotationssymmetrische Elemente betrachtet, daher genügt eine zweidimenionale Topologiebeschreibung der Querschnittsfläche.

Allerdings sind mit der Methode von Güsgen nur orthogonal begrenzte Flächen beschreibbar, eine Erweiterung oder Verbesserung der Beschreibungsmethodik wäre hier sehr interessant

- Zerlegung dieses Top-Bereiches in Teilbereiche mit Hilfe von Regeln. Diese Regeln besitzen als Bedingungsteil eine Topologiebeschreibung des Ausgangselements und als Aktionsteil Topologiebeschreibungen der Teilbereiche.

  Das Matchen der Topologiebeschreibungen mit den Bedingungsteilen der Regeln erfolgt unter Übertragung des in [Nö90] vorgestellten Matchalgorithmus für Zeitsymptome.

4.10 Werkzeugauswahl

Die Werkzeugauswahl ist im wesentlichen ein Constraint-Satisfaction Problem. Ansätze hierfür werden im Rahmen von ARC-TEC verfolgt, daher soll an dieser Stelle auf eine Diskussion verzichtet werden.[38]

4.11 Schnittparameterbestimmung

Die Bestimmung der Schnittparameter erfolgt unter Berücksichtung der Werkstoff/Schneidstoffkombination unter Zuhilfenahme der Daten aus Informationszentren für Schnittwerte (wie etwa INFOS in Aachen). Stehen solche Daten nicht zur Verfügung muß auf eigene Erfahrungswerte, analytische Methoden stehen hierfür erst in geringem Maße zur Verfügung [SK83].

Die Optimierung kann unter verschiedenen Gesichtspunkten erfolgen, wie etwa:

- Maximierung der optimalen Schnittiefe
- Bestimmung des optimalen Vorschubs
- Minimierung der Zerspanungskosten

4.12 NC-Programmgenerierung

Für die in der Arbeitsvorgangsfolgenermittlung bestimmten Operationen erfolgt die NC-Programmgenerierung in den folgenden Schritten:

- Ermittlung der Schnittaufteilung unter Berücksichtigung der maximalen Schnittiefe.

  Möglich sind hier zum Beispiel Quer-, Längs oder Kegeldrehen, Konturdrehen oder eine Orientierung an der Rohteilkontur.

- Bestimmung der Schnittfolge.

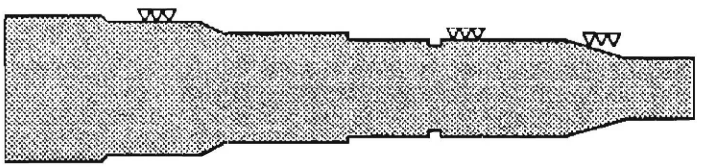
  Die Schnittfolge ist die Reihenfolge aller Einzelschnitte eines Werkzeugs

- Berechnung der Werkzeugwege.

  Werkzeugwege sind die geometrischen Orte aller Werkzeugbewegungen, die bei der Abarbeitung eines Segmentes einen unmittelbaren oder mittelbaren Arbeitsfortschritt erzielen [SK83]. Sie setzen sich zusammen aus dem Hauptweg und den Stellwegen, wie sie in DIN 6580 beschrieben sind.

- Erzeugung des Programms nach DIN 66215 aus den Werkzeugwegen und der Werkzeuginformation.

[38] Interessant ist in diesem Zusammenhang die in Round vorgenommene Kodierung in Toolvektoren



5 Integration bestehender Software

In Kapitel 3 wurde als einer der Mängel bestehender Arbeitsplanungssysteme die mangelnde Fähigkeit, bereits existierende Software zu integrieren, erkannt.

So werden oft Komponenten, z. B. zur Visualisierung der Werkstücke neu implementiert, statt ein CAD-System zu verwenden.

Dies hat außer einem zeitlichen Mehraufwand auch oft eine geringere Funktionalität zur Folge. Die mangelnde Fähigkeit, auf die Datenbestände existierender Datenbanken zugreifen zu können, führt zur Datenredundanz und damit zu einer erhöhten Fehleranfälligkeit.

Ziel ist es daher, ein System wie in Abb. 5.1 realisieren zu können:[1]

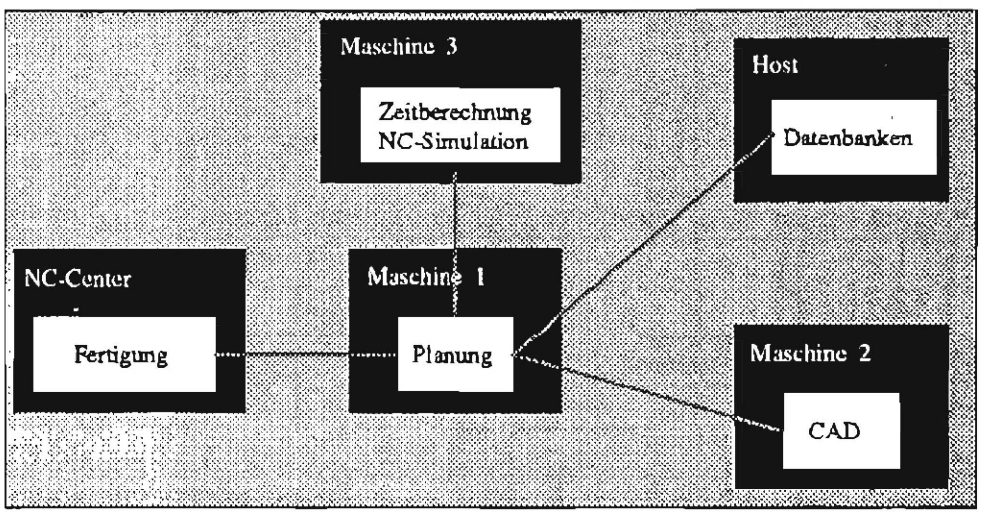


Abbildung 5.1: Ein verteiltes Arbeitsplanungssystem

[1] Das Problem andere Programme aus COMMONLISP heraus zu benutzen, tritt nicht nur in der hier besprochenen Aufgabenstellung auf, es ist vielmehr weit verbreitet.

So ist es einer der Hauptvorwürfe an die „Künstliche Intelligenz", daß sie ihre Entwicklungen in einem Elfenbeinturm spezieller Hard- und Software vollzieht und die daraus resultierenden Ergebnisse nur schwer in eine reale Umgebung zu integrieren sind.

- Das Planungssystem selbst ist in CommonLisp auf einer Workstation realisiert, um z. B. Werkzeugdaten zu erhalten, bedient es sich der zentralen Werkzeug- und Maschinendatenbank auf einem Großrechner.
- Zur Visualisierung wird ein CAD-System benutzt, das auf einer speziellen Graphikworkstation läuft und in C geschrieben ist.
- Die Zeitenberechnung erfolgt in `Fortran-77` auf einem weiteren Rechner, die Simulation und Ausführung des NC-Programms auf einem Leitrechner, an den ein NC-Bearbeitungszentrum angeschlossen ist.

Durch diese Vorgehensweise wird nicht nur die Erstellungszeit des Arbeitsplanungssystems reduziert, gleichzeitig werden dadurch die Investitionskosten gesenkt[2] und die Akzeptanz beim Benutzer erhöht[3].

In diesem Kapitel soll nun die Implementierung einer Integration externer Programme in CommonLisp-Systemen vorgestellt werden. Der erste Teilabschnitt beschreibt mögliche Architekturen und stellt den Client-Server Mechanismus und Sockets vor. Im zweiten Teilabschnitt wird dann der Basismechanismus mit seinem Nachrichtensende- und empfangszyklus und den elementaren Nachrichten diskutiert. Die folgenden Abschnitte sind verschiedenen Anwendungen gewidmet. An erster Stelle steht der Server, gefolgt von dem Anschluß eines 3-D Solid Modelers[4](`IRIT`) und verschiedener Datenbanken (`INGRES` und `POSTGRES`).

Voraussetzungen

Folgende Grundannahmen werden für das weitere Vorgehen gemacht:

1. Die einzelnen Rechner sind über ein schnelles Netz miteinander verbunden und fahren ein einheitliches Netzprotokoll (speziell: `Ethernet` und `TCP/IP`).
2. Die Rechner besitzen ein multitaskingfähiges Betriebssystem (hier: `UNIX`).
3. Die zentrale Instanz bleibt das Arbeitsplanungssystem, von ihm aus werden die externen Programme angesteuert.
4. Das Arbeitsplanungssystem ist in CommonLisp realisiert.

5.1 Grundsätzliches zum Anschluß externer Programme

Ausgehend von den Grundvoraussetzungen soll nun eine Schnittstelle realisiert werden, die über Programmiersprachen und Rechnertypen hinweg *einheitlich, einfach und effizient* ist.

Zuerst werden Möglichkeiten und Probleme der Einbindung anderer Programme in verschiedenen CommonLisp Dialekten skizziert, danach werden verschiedene Architekturmöglichkeiten vorgestellt und ihre Vor- und Nachteile diskutiert. Die genaue Implementierung ist Inhalt von Abschnitt 5.2.

5.1.1 Möglichkeiten und Probleme der Einbindung externer Programme in CommonLisp

Um den Anforderungen der Benutzer gerecht zu werden, bieten die meisten CommonLisp-Systeme heute Möglichkeiten externe Programme einzubinden. Art und Umfang der Einbindung sind bis jetzt aber nicht Teil des CommonLisp-Standards und wurden auch noch nicht in X3J13, dem Standardisierungsausschuß diskutiert.

[2] Indem z. B. für den Rechnertyp auf dem das Planungssystem läuft keine Datenbank angeschafft werden muß, oder der Farbbildschirm der CAD- Workstation mitbenutzt werden kann.

[3] Der Benutzer findet ihm vertraute Programme vor, deren Benutzung er kennt.

[4] Da kein CAD-Paket zur Verfügung stand, mußte auf ein Public Domain Programm zurückgegriffen werden, näheres dazu im entsprechenden Abschnitt.

So wurden sehr unterschiedliche Konzepte realisiert; der Anschluß externer Programme mit Hilfe eines `Foreign Function Interface` ist sehr systemabhängig. Im folgenden sollen kurz drei typische `Foreign Function Interface`'s vorgestellt werden:

1. COMMONLISP auf Lispmaschinen

   Für Lispmaschinen bildet COMMONLISP die Maschinensprache. Compiler für prozedurale Sprachen existieren zwar, werden aber selten mit erworben. Da auch das Betriebssystem in COMMONLISP realisiert ist, werden auch die entsprechenden Systemaufrufe (etwa zur Kommunikation) innerhalb der COMMONLISP-Umgebung bereitgestellt.

   Eine Einbindung externer Programme ist somit praktisch nicht möglich, stattdessen muß auf sie mit Hilfe von Kommunikationsmechanismen zugegriffen werden.

2. Kommerzielle COMMONLISP's auf Workstations

   Kommerzielle COMMONLISP's (z. B. `Allegro, Lucid`) besitzen ein definiertes `Foreign Function Interface` ähnlichen Umfangs, jedoch unterschiedlicher Syntax. Diese `Foreign Function Interface`'s erlauben es, Objektmodule anderer Sprachen einzubinden. Funktionen dieser Module die von COMMONLISP aus benutzt werden sollen, müssen mit Hilfe besonderer Konstrukte (z. B. `define-foreign-functio` in Lucid) in den COMMONLISP-Namensraum eingebunden werden.

   Dies umfaßt meist neben einer Definition des Funktionsnamens noch die Angabe der Typen der Parameter und des Ergebnisses. Es steht eine Reihe von Standardtypen (meist `integer`, `float` und `string`) zur Verfügung. Komplexere Typen und Strukturen können in manchen Systemen definiert werden, der Umfang und die Art dieser Möglichkeiten sind jedoch unterschiedlich.

   Die `Foreign Function Interface`'s sind gut dokumentiert.

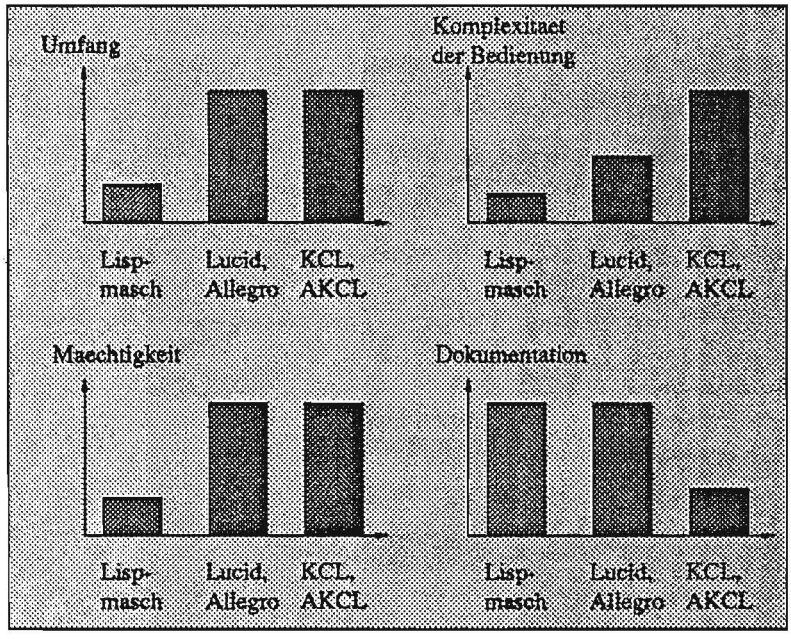


Abbildung 5.2: Die `Foreign Function Interface`'s verschiedener COMMONLISP's im Vergleich

3. Public Domain COMMONLISP's

   KCL[5] bzw. AKCL[6] sind Public Domain COMMONLISP-Systeme, die Lisp- Programme nach C compilieren. So ist ein Integration von C-Programmen leicht möglich. Mit Hilfe von `CLINES` werden die

[5] KCL – Kyoto Common Lisp ist ein an der Universität von Kyoto ursprünglich für ... unter ... entwickeltes COMMONLISP-System, daß mittlerweile auf praktisch allen UNIX-Rechnern verfügbar ist.

[6] AKCL – Austin Kyoto Common Lisp ist eine Weiterentwicklung von KCL die an der Universität von Texas in Austin entstand.

entsprechenden Zeilen C-Code vom COMMONLISP-Compiler unverändert übernommen.[7]

Das Konstrukt `def-c-function` ist in etwa äquivalent zu LUCID's `define-foreign-function`, besitzt jedoch eine geringere Anzahl von Standardtypen. Dafür liegt aber das gesamte COMMONLISP-System im C-Source-Code vor, so daß es innerhalb der C-Funktion leicht möglich ist, die den komplexeren Lisp-Objekten entsprechenden C-Strukturen zu benutzen.

Die Dokumentation des `Foreign Function Interface`'s ist dürftig.

Um nun eine möglichst breite Einsetzbarkeit der entwickelten Schnittstelle zu gewährleisten, muß also eine Schnittmenge an Funktionalität gefunden werden, die auf den verschiedenen `Foreign Function Interface`'s mit vertretbaren Aufwand implementierbar ist.

5.1.2 Architekturmodelle

Um die Verbindung zwischen dem Arbeitsplanungssystem und den externen Programmen herzustellen, sind verschiedene Architekturen denkbar. Hier sollen einige vorgestellt und bezüglich ihrer Vor- und Nachteile bewertet werden.

Dazulinken des externen Programms

Bei dieser Architektur wird das komplette externe Programm als Objektmodul zu den COMMONLISP-Systemen hinzugelinkt. Für die Toplevel-Routinen des externen Programms werden Lisp-Einstiegspunkte definiert.

Dem Vorteil der Schnelligkeit durch fehlende Prozeßwechselzeiten stehen bei dieser Architektur eine Reihe von Nachteilen gegenüber. So ist eine Verteilung auf mehrere Rechner oder eine Realisierung auf der Lispmaschine ist nicht möglich. Zudem entsteht ein riesiger Prozeß, da zu den erfahrungsgemäß großen Lispprozessen noch die externen Programme hinzukommen.

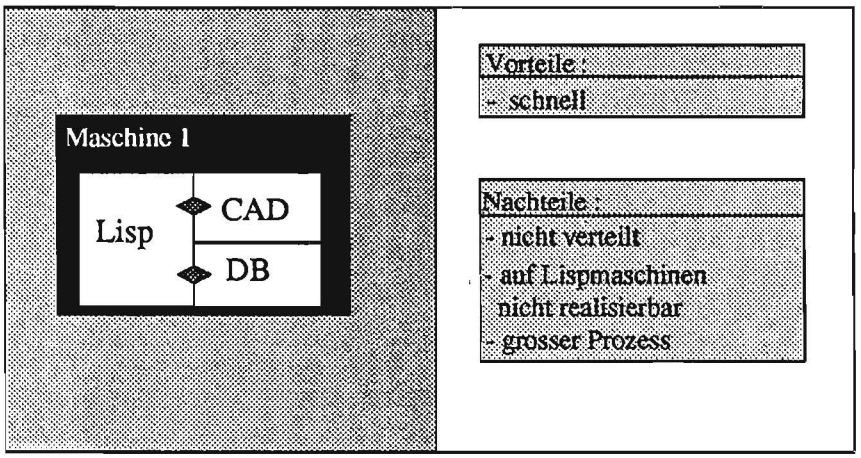


Abbildung 5.3: Dazulinken externer Programme

Getrennte Verbindungen

Bei diesem Architekturmodell sind das Arbeitsplanungssystem und die externen Programme über das Netz miteinander verbunden. Der Verbindungsaufbau erfolgt je nach Verbindung unterschiedlich und dezentral[8].

[7] Eine interpretierte Ausführung einer CLIPES enthaltenden Lispfunktion ist natürlich dann nicht möglich.

[8] Oft entsteht eine solche Architektur, wenn die Hinzunahme der externen Programme sukzessive erfolgt.

Jedes Programm wartet auf eingehende Daten an seinem Socket. In Abhängigkeit der Daten werden entsprechende Aktionen ausgelöst und die Ergebnisse an den Partnerprozeß übermittelt. Ein Protokoll ist oft nur implizit definiert, es variiert von Anwendung zu Anwendung ebenso wie die Schnittstelle.

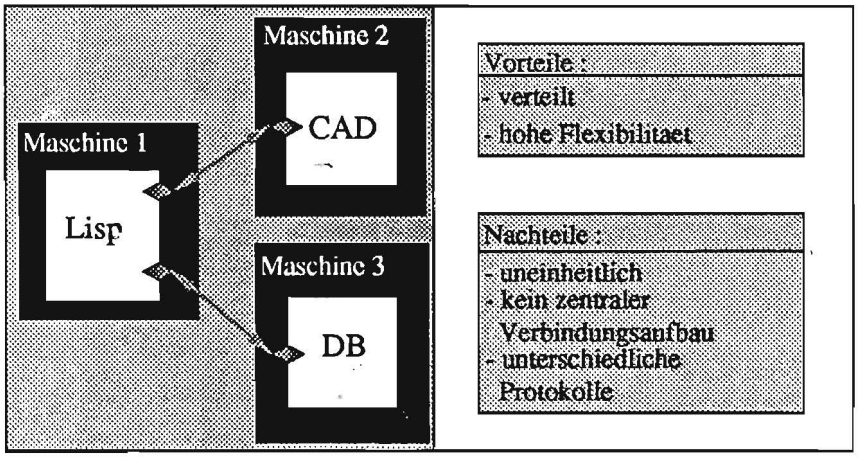


Abbildung 5.4: Getrennte Verbindungen

Zentraler Verbindungsserver

Hier existiert eine zentrale Instanz, über den die Verbindungen zwischen dem Arbeitsplanungssystem und den externen Programmen auf- und abgebaut werden. Besteht ein Verbindungswunsch, so wendet sich das entsprechende Programm an den Server, der diesen wiederum an das Partnerprogramm weitergibt.

Jedes Programm besitzt eine Socketverbindung zum Server, darüberhinaus können beliebig viele Kommunikationssockets offen sein. Der Server führt Buch über die momentan offenen Verbindungen, er kann sie wieder abbrechen.

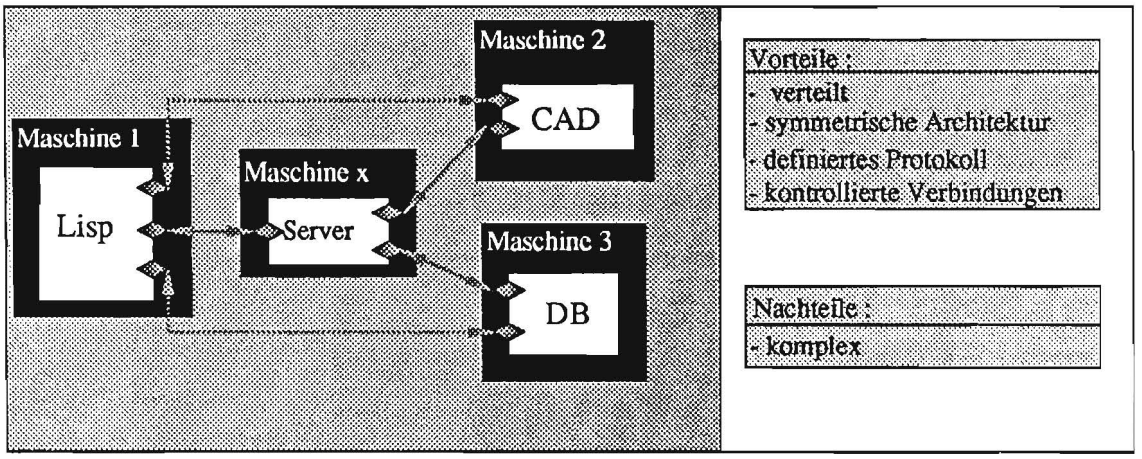


Abbildung 5.5: Zentraler Verbindungsserver

Verteilte Server

Diese Architektur unterscheidet sich von der vorhergehenden dadurch, daß sie nicht mehr symmetrisch ist, der Verbindungsaufbau erfolgt nur in einer Richtung. Darüberhinaus werden die externen Programme erst bei einem Verbindungswunsch des Arbeitsplanungssystems gestartet. Statt einem zentralen Server existiert

pro Rechner ein Server. Er weis, welche externen Programme auf diesem Rechner laufen und wie diese gestartet werden.

Das Arbeitsplanungssystem wendet sich an einen Server mit dem Wunsch das entsprechende externe Programm zu starten. Nachdem dies geschehen ist, kommunizieren die beiden Programme über einen getrennten Kommunikationskanal.

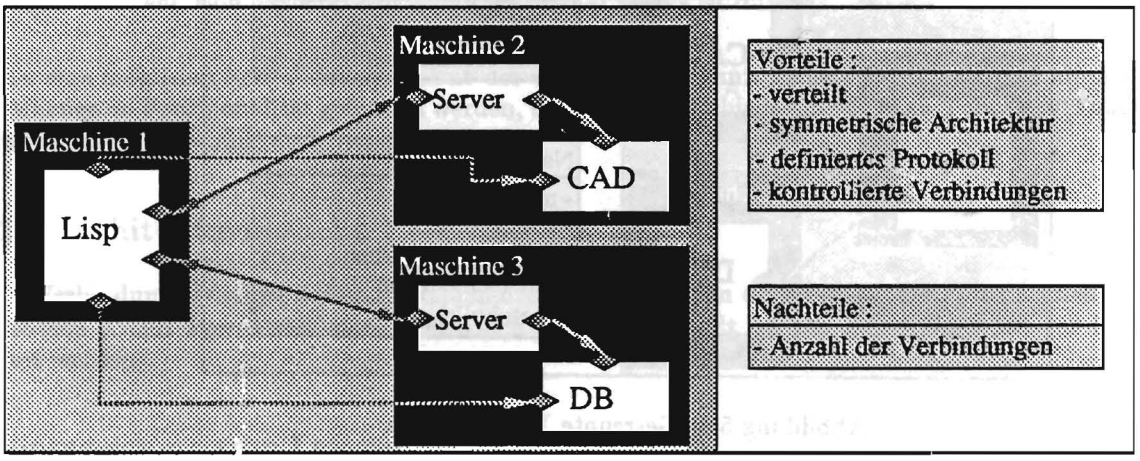


Abbildung 5.6: Verteilte Server

Wie schon in Abschnitt 5 gesagt, wird von der Grundvoraussetzung ausgegangen, daß die Kontrolle im Arbeitsplanungssystem verbleibt. Aus diesem Grunde und der geringeren Komplexität der Lösung wurde für das hier beschriebene System die Architekturvariante 5.1.2 der Variante 5.1.2 vorgezogen.

5.2 Der Server

Die Implemementierung des Servers erfolgt in 3 Teilen, die in den folgenden Teilabschnitten vorgestellt werden sollen:

- Gemeinsame Grundlage des zweiten und dritten Teils ist die Realisierung eines Protokolls und einer Menge von Basiskommandos. Es ist Inhalt von 5.2.1.
- 5.2.2 behandelt die Implementierung des Basismechanismus und des Servers in C.
- Die Integration des Basismechanismus in `Lucid Common Lisp` und `PLAKON` ist Inhalt von 5.2.5.

5.2.1 Basismechanismus und Protokoll

Wegen der in 5.1.1 geschilderten Probleme ist das Protokoll zwischen dem Arbeitsplanungssystem und dem externen Programmen so einfach wie möglich zu wählen. Abbildung 5.7 zeigt die vertikale Struktur des Protokolls.

Sockets Die Basis bilden in beiden Programmen `Sockets`. Dies sind Endpunkte der Kommunikation zwischen Prozessen sowohl auf dem lokalen, als auch auf den über das Netz erreichbaren Rechnern[9].

Sockets sind bidirektionale Datenpfade, der sichtbare Teil der Kommunikation besteht aus drei Teilen:

[9] Der Mechanismus der Sockets wurde in Berkeley-`UNIX` primär zur Kommunikation zwischen Prozessen über ein Rechnernetz eingeführt, ebenso wie der Mechanismus der Streams in `AT&T`-Systemen. Die Programme des inzwischen zur Kommunikation zwischen den Systemen unterschiedlicher Rechnerhersteller recht verbreiteten `TCP/IP`-Pakets stützen sich z. B. auf Sockets ab.

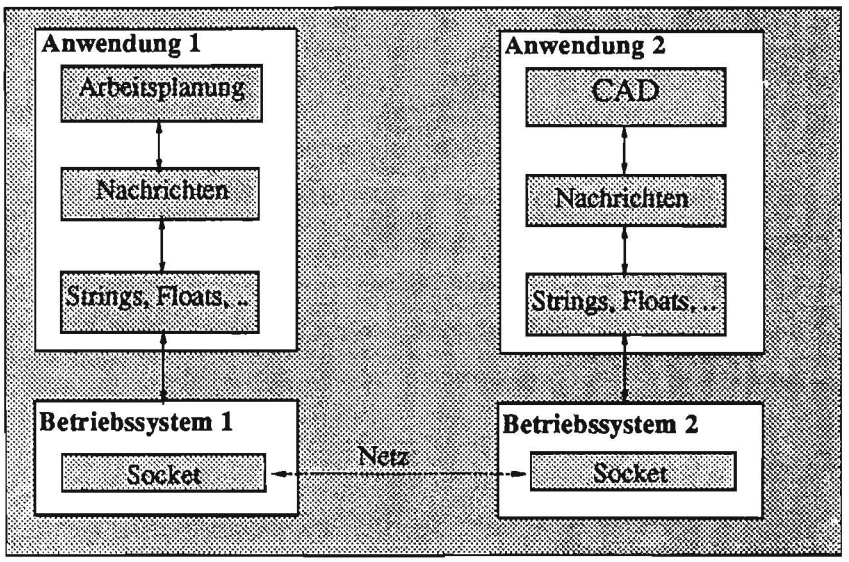


Abbildung 5.7: Das Kommandosprotokoll

- Socket-Kopf
- Protokollteil
- Gerätetreiber

Sockets mit gleichen Charakteristika werden in sogenannten **Domains** zusammengefaßt. Üblich sind die **Unix System Domain** zur Kommunikation zwischen Prozeßen auf der lokalen Maschine und die **Internet Domain** für die Kommunikation über ein Netzwerk unter der Verwendung des **DARPA**-Protokolls.

Sockets werden nochmals in unterschiedliche Typen untergliedert. Der sogenannte **Stream**-Typ stellt eine virtuelle, gesicherte, verbindungsorientierte Kommunikation zur Verfügung, der Typ **Datagram** eine Verbindung für Datagramme; d. h. es wird eine Nachricht an ein oder mehrere Adressaten geschickt, der Empfang ist jedoch nicht gesichert und die Reihenfolge der Kommandos nicht garantiert.

Kommandos und Argumente Über die Sockets werden Kommandos zwischen dem Arbeitsplanungsystem und den externen Programmen versandt. Art und Anzahl der Kommandos, sowie Anzahl und Typ der Kommandoargumente sind fest definiert.

Jedes Kommando besteht aus

- dem Kommandotyp
- der Anzahl der Argumente
- und den Argumenten.

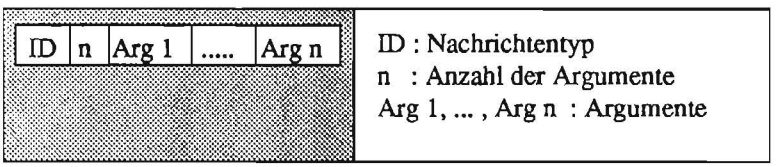


Abbildung 5.8: Das verwendete Kommandoformat

Jedes Argument selbst besteht aus drei Teilen:

- dem Argumenttyp
- der Argumentlänge
- und dem eigentlichen Inhalt.

Momentan werden vier verschiedene Argumenttypen unterstützt: **integer**, **float**, **symbol** und **string**. Eine Erweiterung der Argumenttypen ist jedoch leicht möglich wenn sie benötigt wird.

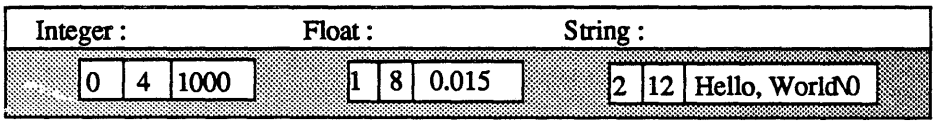


Abbildung 5.9: Typische Kommandoargumente

Zum Senden und Empfangen der Argumente und Kommandos müssen in den Partnerprogrammen entsprechende Funktionen definiert werden, dies ist der einzige Teil der mit Hilfe der `Foreign Function Interface`'s realisiert werden muß. Das kann durch Integration entsprechender C-Funktionen (in `KCL` und `LUCID`) oder Aufruf entsprechender Lispprimitiva (bei Lispmaschinen) geschehen.

Aufbauend auf dieser Schnittstelle kann dann die weitere Anwendung in COMMONLISP geschrieben werden.

Anwendungen Die Anwendungen bedienen sich einer Auswahl an Kommandos um die gewünschten Leistungen im jeweiligen Partnerprozeß abzurufen. Jeder Kommandotyp erhält einen eindeutigen Kommandoidentifier und ist mit einer Kommandobehandlungsfunktion verbunden. Diese wird bei Erhalt des Kommandos angestoßen. Wird die Behandlungsfunktion nur um ihrer Seiteneffekte willen ausgeführt, unterbleibt eine Zurücksendung des Funktionergebnisses. Ansonsten wird als Funktionsergebnis ein Kommando erwartet, das an den ursprünglichen Sender zurückgeschickt wird.

Auch der Server wird als Anwendung realisiert. Er unterscheidet sich von den anderen Anwendungen nur durch die Kommandos die er versteht.

5.2.2 Die Implementierung des Servers in C

Allgemeines zur Client-Server Kommunikation

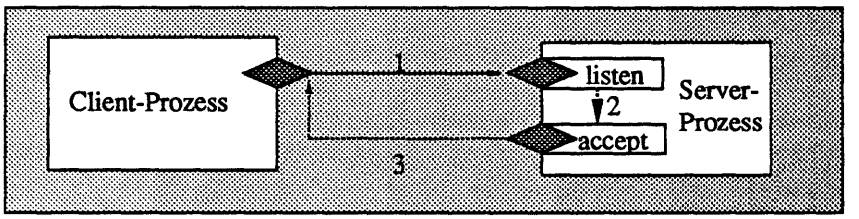


Abbildung 5.10: Client-Server Kommunikation

Wie schon in 5.2.1 erwähnt, basiert die gewählte Architektur auf dem Socket-Mechanismus. Eine Kommunikation läuft hierfolgendermaßen ab: Ein *Server*-Prozeß baut mittels des Aufrufs **socket** einen Kommunikationspunkt auf. Dabei können Typ und Bereich des Sockets angegeben werden. Mit **bind** kann nun ein Name an den Socket gebunden werden. Ein *Client*-Prozeß koppelt sich ebenfalls an einen (lokalen) Kommunikationsendpunkt und beantragt mit **connect** einen Verbindungsaufbau zu dem Socket des Servers. Der Serverprozeß macht mit **listen** dem System bekannt, daß er Verbindungen akzeptieren will und gibt die Länge einer Warteschlange an. Mit **accept** wartet er darauf, daß ein Clientprozeß eine Verbindung anfordert. Der **accept**-Aufruf liefert nach einem Verbindungsaufbau dem Server einen neuen Socket-Deskriptor

(analog zu einem Dateideskriptor) für einen anderen Socket zurück, über den nun die Kommunikation mit dem Client erfolgen kann. Der Austausch von Daten ist danach mit **send** und **receive** oder mittels **write** und **read** über diesen Socket möglich. Wie man in Abbildung 5.10 sieht, sind der Socket, an dem der Serverprozeß auf Verbindungen wartet, und der Socket, über den der Server nach einem Verbindungsaufbau mit dem Client kommuniziert, auf der Serverseite nicht identisch. Der Aufruf **shutdown** schließlich baut die Verbindung wieder ab.

Bei der hier gewählten Implementierung kommt hinzu, daß sowohl ein **Internet** als auch ein **UNIX-Domain** Socket aufgebaut werden. Das parallele Warten auf Verbindungswünsche an zwei Sockets wird mit Hilfe des **select**-Statements realisiert.

Aufbau des Internet und UNIX-Sockets

```
main(argc,argv)
int argc;
char **argv;
{
  struct sockaddr_un sa_un,  struct sockaddr_in sa_in;     /* the sockets: a unix and a internet */
  long  socket_mask, unix_socket_mask, inet_socket_mask;   /* the socket masks */
  struct timeval timeout;                                  /* the socket timeout */
  int num_desc, int ns, retries;

  /* Create one unix domain socket and one internet socket */
  /* the unix socket */
  if( (unix_socket = socket(AF_UNIX, SOCK_STREAM, 0)) < 0 )
    { perror("socket"); exit(1); }
  /* the internet socket */
  if( (inet_socket = socket(AF_INET, SOCK_STREAM, 0)) < 0 )
    { perror("socket"); close_sockets(); }
  /* Bind unix socket to a path in the directory hierarchie */
  sa_un.sun_family = AF_UNIX;
  strcpy(sa_un.sun_path, SOCKET_PATH);
  for( retries = 0; retries < 30; retries++ )
    {
      if( bind(unix_socket,&sa_un,strlen(SOCKET_PATH)+2) < 0 ) /* binding not successful */
        { .. }
      else
        { break; }
    }
  /* Bind the internet socket to an internet adress */
  bzero((char *)&sa_in, sizeof(sa_in));
  sa_in.sin_family = AF_INET;
  sa_in.sin_port = htons(SOCKET_PORT);
  sa_in.sin_addr.s_addr = htonl(INADDR_ANY);
  for( retries = 0; retries < 30; retries++ )
    {
      if( bind(inet_socket,(struct sockaddr *)&sa_in,sizeof(sa_in)) < 0 )
        /* binding not successful */
        { ... }
      else
        { break; }
    }
  /* Set up signal handlers to close down sockets properly */
  ...
  /* Initialize queues for sockets */
  if( listen(unix_socket,5) < 0 )
    { perror("listen"); close_sockets(); }
  if( listen(inet_socket,5) < 0 )
    { perror("listen"); close_sockets(); }
  /* Now wait for clients */
  /* initialize the socket masks */
  unix_socket_mask = 1 << unix_socket;
  inet_socket_mask = 1 << inet_socket;
  socket_mask = unix_socket_mask | inet_socket_mask;
  /* initialize the socket timeout */
  timeout.tv_sec = (long)10000;
  timeout.tv_usec = (long)0;
```

```
  /* wait for the clients */
  while (num_desc = select(32,&socket_mask,0,0,&timeout))
    {
      if (num_desc == -1)
        { printf("%d",errno); } /* nothing received */
      else
        {
          if (socket_mask & unix_socket_mask)
            { fork_server(unix_socket); }              /* client arrived at the unix socket */
          if (socket_mask & inet_socket_mask)
            { fork_server(inet_socket); }
        }
      /* restore the socket mask */
      socket_mask = unix_socket_mask | inet_socket_mask;
    }
}
```

Trifft ein Verbindungswunsch auf einem der beiden Sockets ein, wird ein Tochterprozeß erzeugt, der dann auf dem mit **accept** erzeugten Socket mit dem Client kommuniziert. Der Vater trägt den Sohnprozeß in die Liste seiner Söhne ein und wartet danach auf weitere Verbindungswünsche.

Erzeugen des Sohnprozesses

```
void fork_server(sock)
int sock;

{
  int pid;
  int desc;
  struct sockaddr sa;
  int sa_len = sizeof(struct sockaddr);
  int i;
  Command *cmd;

  /* establish the communication socket */
  if( (desc = accept(sock, &sa, &sa_len)) < 0 )
    /* not succesful */
    {
      if( errno == EINTR )
        return;
      perror("accept");
      close_sockets();
    }
  /* fork a son. The son does the communication, the father
   * continues
   */
  switch( pid = fork() )
    {
    case 0:                     /* child */
      /* close the sockets, leave the communication socket
       * and stdin, stdout, stderr open
       */
      for(i=getdtablesize()-1; i>2; i--)
        if( i != desc )
          close(i);
      /* Start the command handling loop */
      CommandLoop(desc);
      /* CommandLoop exited */
      close_sockets();
      /* NOTREACHED */
    case -1:                    /* fork() failed */
      perror("fork");
      return;
    default:                    /* parent */
```

```
    /* add the son in the child_list and close the
     * communication socket
     */
    add_child(pid,&sa,sa_len);
    close(desc);
    return;
  }
```

Die Kommunikation des Sohnes mit dem Clienten erfolgt in einer *Kommandoschleife*: Es wird auf ein *Kommando* gewartet, das Kommando wird ausgeführt, und falls notwendig, das Ergebnis dieses Kommandos anderen Klienten zurückgesandt[10].

Abbildung 5.11: Die Kommandoschleife des Servers

5.2.3 Kommandos

Die Kommandos selbst sind folgendermaßen aufgebaut: Einem Kopf bestehend aus Kommandoid und Argumentzahl folgen die einzelnen Argumente, die aus der Argumentlänge, dem Argumenttyp und dem jeweiligen Argument bestehen.

Kommando- und Argumentstruktur

```
/* the different argument types */

union CmdValue {
    int      int_value;
    char    *string_value;
    char    *symbol_value;
    float    float_value;
    /*caddr_t *binary_value;*/
};

/* the command argument structure */

typedef struct {
    int            arg_type;  /* Type of argument */
    union CmdValue v;
} CmdArg;

/* the command structure */

typedef struct {
    int     command;  /* Command to be executed */
    int     num_arg;  /* Number of arguments to be transferred */
    CmdArg *args;     /* List of arguments */
} Command;
```

Die eigentliche Kommandoübertragung erfogt gepuffert, wobei der Übertragungspuffer beiÜberlauf oder durch expliziten Aufruf geleert wird. Sowohl die C- als auch die Lispseite benutzen dazu die Funktionen aus der Datei „protocol.c“. Auf Lispseite muß daher diese Datei mit Hilfe des `Foreign Function Interface`'s eingebunden werden[11].

Eine Protokollfunktion

[10] Entsprechend der Read-Eval-Print Schleife von Lisp

[11] Näheres darüber im folgenden Abschnitt

```
int SendString(sock, string )
int   sock;
char *string;
{
  int length = (string ? (*string ? strlen(string) : 0) : 0);

  if( output_buffer.pos >= SERVER_BUFSIZ - 2*sizeof(int) - length)
    if( FlushBuffer(sock) == -1)
      return(-1);

  SEND_BYTES(&ArgIsString,sizeof(int));
  SEND_BYTES(&length,sizeof(int));
  if( length > 0)
    SEND_BYTES(string,length);
  return(0);
```

Es werden zwei Typen von Kommandos unterschieden:

- *Basiskommandos*
- *Anwendungsspezifische Kommandos*

Die *Basiskommandos* dienen im wesentlichen der Übertragung der primitiven Argumente und dem Abbruch der Verbindung zwischen Client und Server. Darüber hinaus wird noch ein Kommando zur Versendung von Warnungen und Fehlern bereitgestellt. Tabelle 5.1 zeigt die definierten Basiskommandos.

Nachricht	ID	Anz. Arg.	Argumente
SendInteger	0	1	integer
SendString	1	1	string
SendSymbol	2	1	symbol
SendFloat	3	1	float
QuitApplication	4	0	
QuitServer	5	0	

Tabelle 5.1: Die BasisKommandos des Kommunikationskonzeptes

Anwendungsspezifische Kommandos realisieren die Funktionalität der jeweiligen Anwendung. Im Falle des Servers besteht der Satz der anwendungsspezifischen Kommandos aus dem Kommando zum Starten einer Anwendung[12].

5.2.4 Die Kommandoschleife

Die Kommandoschleife testet nach Empfang eines Kommandos zuerst, ob es gültig ist und welchem Kommandotyp es angehört. Für die beiden Kommandotypen existieren Tabellen, in denen den Kommandos entsprechende Behandlungsfunktionen zugeordnet sind. Die dem Kommando zugeordnete Funktion wird ausgeführt. Soll ein Ergebnis an den Sender zurückübermittelt werden, so muß die Behandlungsfunktion dieses in Form eines Kommandos als Wert liefern. Erfolgte der Aufruf der Behandlungsfunktion nur wegen ihrer Seiteneffekte, so liefert sie `NULL` als Wert. Bevor die Behandlungsfunktion ihre eigentliche Aufgabe ausführt, testet sie die empfangenen Argumente. Sind diese nicht in Ordnung, wird eine Fehlermeldung zurückgeschickt.

[12] Die Kommandos der anderen Anwendungen werden jeweils dort vorgestellt.

Die Kommandoschleife

```
void CommandLoop(sock)
int sock;
{
  Command *cmd, *rc;
  current_sock = sock;
  while( ! closed ) {
    if( (cmd = ReceiveCommand(sock)) == NULL ) {
      fprintf(stderr,"CmdLoop: ReceiveCommand failed\n");
      fflush(stderr);
      abort();
    }
    if( cmd->command < 0 || cmd->command > cmdMaxID)
      if ((cmd->command <= BasicCmdMaxID && BasicFunctionTable[cmd->command]==NULL)
          ||(cmd->command > BasicCmdMaxID && FunctionTable[(cmd->command)-BasicCmdMaxID]==NULL))
        {
          fprintf(stderr, "main: illegal command ID: %d\n", cmd->command);
          fflush(stderr);
        }
    if (cmd->command <= BasicCmdMaxID)
      {
        rc = (*BasicFunctionTable[cmd->command])(cmd);
      }
    else
      {
        rc = (*FunctionTable[(cmd->command)-BasicCmdMaxID])(cmd);
      }
    FreeCommand(cmd);
    if( rc != NULL ) {
      if( SendCommand(sock, rc) == -1 ) {
        fprintf(stderr,"CmdLoop: SendCommand failed\n");
        fflush(stderr);
        abort();
      }
    }
    else {
      if( synchronous && ! confirmed ) {
        if( ConfirmCommand(sock) == -1 ) {
          fprintf(stderr,"CmdLoop: ConfirmCommand failed\n");
          abort();
        }
      }
      confirmed = FALSE;
    }
  }
}
```

Eine Behandlungsfunktion

```
Command *FhandleEvent(cmd)
Command *cmd;
{
  int eventType;

  if (check_command(cmd,EventCmd,EventNumArg) >= 0)
    {
      eventType = cmd->args[0].v.int_value;
      switch (eventType)
        {
        case 0:
          /* Error */
          printf("Error : %s\n",cmd->args[1].v.string_value);
```

```
            break;
          case 1:
            /* Warning */
            printf("Warning : %s\n",cmd->args[1].v.string_value);
            break;
          case 2:
            /* Confirmation */
            printf("Confirmation\n");
            break;
          default:
            printf("Wrong EventType\n");
          }
        return(NULL);
      }
}
```

Das Kommando StartApplication

Wegen seiner großen Bedeutung für die nachfolgenden Abschnitte soll hier auf das Kommando **StartApplication** etwas genauer eingegangen werden. Eine Anwendung wird durch einen Prozeßwechsel gestartet, indem durch den Aufruf der Funktion **execvp** der Speicherinhalt des bisherigen Prozesses mit dem der gewünschten Anwendung überschrieben wird und bestehende Kommunikationskanäle erhalten bleiben.

execvp erwartet zwei Argumente:

- Den Namen des auszuführenden Programms, das entweder eine ausführbare Objektdatei oder ein Shellscript sein kann
- Ein **NULL** terminiertes Feld von Argumenten für dieses Programm

Die auszuführende Datei muß sich bei Aufruf von **execvp** in einem Verzeichnis befinden, auf das in der Environment Variablen **PATH** verwiesen wird. Da bestehende Kommunikationsverbindungen bei **execvp** erhalten bleiben, kann auch in dem gestarteten Programm mit dem Client kommuniziert werden. Die Socketnummer wird als erstes Argument an die Anwendung übergeben. Wird eine Umlenkung von **stdin** oder **stdout** gewünscht, so wird dies wie die Weiterreichung des Kommunikationssockets durch den dritten Parameter von **StartApplication** gesteuert. Die Umlenkung selbst geschieht durch einen entsprechenden Aufruf von **freopen**. Das erste Argument von **StartApplication** ist der Name der zu startenden Anwendung, das zweite ein String, in dem Argumente an die Anwendung wie bei Aufrufen von Shell-Ebene aus angegeben werden können.

Das Kommando StartApplication

```
Command *FstartApplication(cmd)
Command *cmd;
{
  char *c;
  extern char **environ;
  Command *ret;
  int comm;

  if ((ret =check_command(cmd,StartApplicationCmd,StartApplicationNumArg)) != NULL)
    { return(ret); }
  else
    {
      comm = (cmd->args[2].v.int_value);
      if (comm & 1)
        /* with socket */
        {
          sprintf(socket_str,"%ld",current_sock);
          app_args[1]=malloc(sizeof(socket_str));
          strcpy(app_args[1],socket_str);
          str2args(cmd->args[1].v.string_value,&app_args[2]);
        }
      else
        { str2args(cmd->args[1].v.string_value,&app_args[1]); }
      if (comm & 2)
        /* stdin redirection */
        { freopen(cmd->args[3].v.string_value,"r",stdin); }
      if (comm & 4)
        /* stdout redirection */
        { freopen(cmd->args[4].v.string_value,"w",stdout); }
      app_args[0]=malloc(sizeof(cmd->args[0].v.string_value));
      strcpy(app_args[0],cmd->args[0].v.string_value);
      /* and start the application */
      execvp(cmd->args[0].v.string_value,app_args);
      return(NULL);
    }
}
```

5.2.5 Die Integration in Lisp

Bei der Integration des Protokolls und des Servers in Lisp sind mehrere Ebenen zu unterscheiden:

1. Anbindung des Protokollteils mit Hilfe des `Foreign Function Interface`'s
2. Aufbauend darauf eine Realisierung des Basismechanismus und der Anwendungen
3. Integration der Anwendungen in FRESKO

Die verschiedenen Ebenen sollen nun in den nächsten Abschnitten vorgestellt werden.

5.2.6 Der Protokollteil

Aufgabe des Protokollteils ist die Einbindung der in der Datei „protocol.c" definierten Funktionen zur Übertragung der Argumenttypen in den Lisp-Namensraum. Hinzu kommt noch die Einbindung einer Funktion zum Aufbau der Verbindung zum Server. Hier, wie bei Einbettung der Anwendung, wurde sich an der

Realisierung ähnlicher Problemstellungen in CLX[13] und CLM[14] orientiert[15]. Zur Einbettung von Funktionen anderer Sprachen in Lucid Common Lisp ist ein zweistufiges Vorgehen notwendig. Zuerst müssen die Objektdateien der fremden Funktionen geladen werden. Dies geschieht in unserem Falle durch den Befehl `(load-foreign-files (list " sun3/unixsock.o" " sun3/protocol.o" )`. Die Bindung der fremden Funktion mit entsprechenden Lispfunktionen erfolgt durch Aufrufe der Funktion `def-foreign-function`. Hier müssen zu einem Lispfunktionsnamen der Name der entsprechenden fremden Funktion, die Implementierungssprache und die Typen der Parameter und des Ergebnisses angegeben werden.

Einbindung fremder Funktionen mit `def-foreign-function`

```
#+lucid
(def-foreign-function (send-string (:name "_SendString")
                                   (:return-type :fixnum))
  (stream :fixnum)
  (value :simple-string))
```

Die Verbindung zum Server wird mit Hilfe der Funktion `connect-to-server` aufgebaut, die als Parameter den Hostnamen des Servers und die Socketnummer des Empfangsservers erhält und die Socketnummer des Kommunikationssocket zurückliefert. `connect-to-server` wiederum wird durch die Funktion `open-connection-stream` aufgerufen.

Anmerkung: Die Realisierung mittels eines 'Raw Sockets' erfolgte in Lucid Common Lisp aus reinen Effizienzgründen. Eine elegantere Lösung ist es, mit Hilfe von `make-tcp-stream` einen echten Stream als Ergebnis von `open-connection-stream` zu . Dieser kann dann als normaler Lisp-Stream genutzt werden, womit die Einbindung der Protokollfunktion wie `SendString` entfallen kann. Damit ist das `Foreign Function Interface` auf eine einzige Funktion, namentlich `connect-to-server` reduziert.[16].

5.2.7 Realisierung des Basismechanismus und der Anwendungen

Da ein möglichst geringer Anteil der Funktionalität mit Hilfe des `Foreign Function Interface`'s zur Verfügung gestellt werden sollte, muß der Basismechanismus und der Anschluß der Anwendungen auch in CommonLisp implementiert werden. Dies umfaßt in erster Hinsicht die Funktionen zum Senden und Empfangen von Kommandos.

Verbindungen zu Anwendungen werden wie folgt realisiert:

Eine Verbindung wird durch die Struktur `connection` repräsentiert, die neben der Kommunikationsverbindung noch Verweise auf Timeout- und Eventbehandlungsfunktionen sowie einer Schleifenfunktion (entsprechend der Kommandoschleife in C) enthält. Gestartet wird eine Anwendung durch Aufruf der Funktion `run-application` in der die Verbindung initialisiert und die Schleifenfunktion aufgerufen wird.

[13] CLX ist eine Kopplung von CommonLisp mit X (einem netzwerkbasiertem Fenstersystem) das am MIT entwickelt wurde.
[14] CLM ist eine Kopplung von CommonLisp an das Motif-Widget-Set der Open Software Foundation. Motif ist eine Reihe vordefinierter Fenstertypen (wie Texteditor etc.) das auf X aufbaut.
[15] Die weiteren Ausführungen beziehen sich auf Lucid Common Lisp, dem Code sind aber auch Ideen zur Realisierung in Symbolics Common Lisp zu entnehmen
[16] Entsprechend kann auf der Symbolics verfahren werden, indem die Funktion `open-tcp-stream` zum Einsatz kommt

Die Struktur connection

```
(defstruct connection
  (stream nil)
  (dispatcher-terminated nil)
  (sync nil)
  (command-handler-table *default-command-handler-table*)
  (main-loop-function nil)
  (main-loop-args nil)
  (event-handler 'default-event-handler)
  (event-handler-args nil)
  (timeout-handler 'default-timeout-handler)
  (timeout-handler-args nil)
  (timeout-detector 'default-timeout-detector)
  (timeout-detector-args nil)
  (last-tick 0)
  (running nil))
```

Soweit es durch das unterliegende COMMONLISP unterstützt wird, können Anwendungen in eigenen Lisp-internen Porzessen laufen[17]. Gesteuert wird dies durch den Schlüsselwertparameter `extra-process` von `run-application`. Damit können mehrere Anwendungen gleichzeitig vom Lisp aus genutzt werden.

5.2.8 Integration in FRESKO

Es wird eine Klasse `application` definiert, die als Instanzvariablen die Schlüsselparameter von `run-application` enthält. Der Aufruf von `run-application` wird durch die generische Funktion `run` ersetzt.

Die Klasse application

```
(create-class application
              :super frame-object
              :meta hashtable-class
              :instance-slots
              ((arguments "")
               (with-socket t)
               (stdin-redirection nil)
               (stdout-redirection nil)
               (init-function 'default-init-function)
               (init-arguments nil)
               (application-class "")
               (server-host nil)
               (extra-process t)
               (execute-main-loop t)
               (sync nil)
               (command-handler-table nil)
               (process-name "Application")
               (after-function nil)
               (after-arguments nil)
               (main-loop-function nil)
               (main-loop-args nil)
               (event-handler 'default-event-handler)
               (event-handler-args nil)
               (timeout-handler 'default-timeout-handler)
               (timeout-handler-args nil)
               (timeout-detector 'default-timeout-detector)
               (timeout-detector-args nil)
               (close-on-exit t)
               (connection nil)))
```

[17] Sogenannten 'Lightweight Processes', da sie keine Betriebssystemprozesse repräsentieren, sondern durch COMMONLISP innerhalb des Lisp-Namensraumes verwaltet werden

`Application` selbst ist eine abstrakte Klasse und dient als Basisklasse für die Definition echter Anwendungsklassen, wie wir im nächsten Abschnitt feststellen werden.

5.3 Die Anwendung `Irit`

`Irit` ist ein 3-D CSG Solid Modeler, der in Ermangelung eines professionellen CAD-Systems zur Werkstückvisualisierung eingesetzt werden soll. Er erhielt den Vorzug vor anderen Public Domain[18] Paketen, da er CSG basiert ist, geeignete Primitiva wie `Cylinder`, `Cone`, `Torus` bereithält, und die Unterstützung durch den Entwickler außerordentlich gut ist. Seine Anbindung soll hier als prototypische Anwendung des in den letzten Abschnitten vorgestellten Konzepts zum Anschluß externe Programme dienen. Dazu wird zuerst `Irit` und seine Fähigkeiten vorgestellt, bevor in 5.3.2 der Anschluß an Lisp und den Server und die Einbettung in FRESKO besprochen werden.

Die Anbindung verschiedener Hilfsprogramme zum Hidden-line-removal, Rendering und Bildwiedergabe, die im Rahmen von `Irit` mitgeliefert werden ist Inhalt von 5.3.3

5.3.1 `Irit` –eine Kurzbeschreibung

`Irit` ist ein batchorientierter polygonaler CSG Solic Modeler. CSG, Constructive Solid Geometry, ist ein Modellierungsverfahren, bei dem komplexere Objekte mit Hilfe von Mengenoperatoren aus einfacheren Objekten entstehen. Als Basisobjekte stellt `Irit` u. a. die Primitiva `Box`, `Cone`, `Cylinder`, `GBox`, `Sphere`, `Surfrev` und `Torus` zur Verfügung.

Primitiv	Beschreibung
Box	Achsenparaller Quader
Cone	Kegel
Cylinder	Zylinder
GBox	Allgemeiner Quader
Sphere	Kugel
Surfref	Rotationskörpe um die Z-Achse
Torus	Torus

Tabelle 5.2: Die Primitivkörper von `Irit`

Zur Kombination dieser Körper können die Operatoren `+`,`-` und `*` verwendet werden, die für Vereinigung, Differenz und Schnitt der jeweiligen Objekte stehen. `Irit` liest seine Kommandos entweder von `stdin` oder einer Datei. Nach seinem Start werden 2 Fenster geöffnet: Das View und das Input Fenster[19]

Die Darstellung der Objekte auf dem View-Fenster wird durch eine Transformationsmatrix `view_mat` bestimmt. Zur Beeinflussung dieser Matrix und der Objekte stehen Operatoren zur Skalierung, Verschiebung und Rotation zur Verfügung: `RotX`, `RotY`, `Rotz`, `Scale` und `Trans`. Als Zuweisungsoperator dient das `=`. Schleifen können mit dem `For` Statement gestaltet werden. Mit Hilfe der Befehle `View` und `Interact` werden ausgewählte Objekte unter Verwendung der momentanen `view_mat` auf dem View-Fenster dargestellt, wobei `Interact` eine Veränderung der Darstellung über ein Menü erlaubt.

[18] `Irit` ist via anonymen FTP von `wuarchive.wustl.edu (128.252.135.4)` im Verzeichnis `/mirrors/msdos/gif` erhältlich.
[19] `Irit` wurde ursprünglich für den IBM PC entwickelt und erst später auf UNIX Maschinen und X portiert.

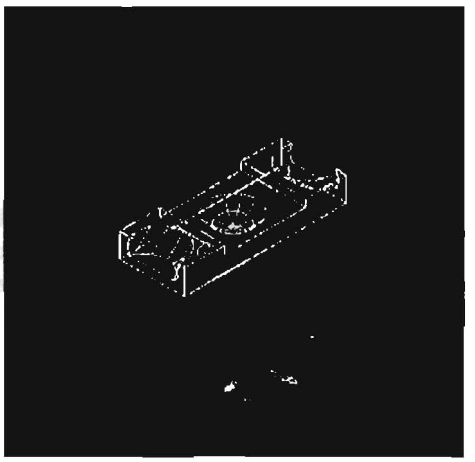


Abbildung 5.12: Ein mit **Irit** modelliertes Objekt

Die Befehle zur Erzeugung des Objektes aus Abb.5.12

```
save_mat = view_mat;
save_res = resolution;
resolution = 12;
B1 = box(vector( -0.5, -0.2, 0.0), 1.0, 0.4, 0.15);
B2 = box(vector(-0.25, -0.3, 0.1), 0.5, 0.6, 0.5);
M1 = B1 - B2;
free(B1);
free(B2);
C1 = sphere(vector( 0.0, 0.0, 0.2), 0.18);
M2 = M1 - C1;
free(M1);
free(C1);
C2 = plane(vector( 0.0, 0.0, 1.0), vector( 0 55, 0.0, 0.05), 0.15);
C2 = extrude(C2, vector(-0.2, 0 0, 0.2));
C3 = plane(vector( 0.0, 0.0, 1.0), vector(-0.55, 0.0, 0.05), 0.15);
C3 = extrude(C3, vector( 0.2, 0.0, 0.2));
M3 = M2 - C2 - C3;
free(M2);
free(C2);
free(C3);
final = convex(M3);
free(M3);
interact(list(final), false);
```

5.3.2 Die Anbindung von Irit

Zum Anschluß von **Irit** müssen sowohl auf der C als auch auf der Lisp-Seite entsprechende Erweiterungen vorgenommen werden. Auf der C-Seite umfaßt dies einerseits geringe Änderungen an **Irit** selbst, andererseits die Definition von Kommandos und Behandlungsfunktionen für die **Irit**-Befehle und einer Anwendung, die sich wiederum an **Irit** wendet. Auf der Lisp-Seite werden Funktionen zur Ansteuerung von **Irit** und zur Erzeugung neuer Objekte definiert, und **Irit** als spezielle **Application** in der FRESKO-Hierarchie verankert.

Anmerkung: Die Anbindung von **Irit** ist so gestaltet, daß an ihr exemplarisch die verschiedenen Möglichkeiten zur Erweiterung der Funktionalität einer Anwendung aufgezeigt werden können, an den jeweiligen Stellen wird darauf näher eingegangen.

Änderungen an Irit

Irit selbst wird so verändert, daß es statt von **stdin** von einer Pipe liest.

Irit-CON die Anwendung

Irit-CON ist die Anwendung, die von Lisp aus gestartet wird und die sich ihrerseits an Irit wendet. Vor Aufruf der Kommandoschleife zur Behandlung der ankommenden Kommandos eine Pipe erzeugt. Dann wird ein Sohnprozeß mit dem modifizierten Irit und dieser Pipe als Eingabe gestartet. Die Behandlungsfunktionen der Kommandos schreiben dann die entsprechenden Irit}-Befehle in die Schreibseite dieser Pipe und leiten sie so an Irit weiter.

Die Anwendung Irit-CON

```
main(argc, argv)
        int     argc;
        char    **argv;
{

  int pid;                      /* the process id returned by fork */
  int ret;                      /* the return value of pipe */

  cmdMaxID=CmdMaxID;            /* set the maximal command id */
  /* Get the file descriptor of the socket stream */
  sock = atoi(argv[1]);
  /* open the pipe */
  ret=pipe(pipe1);
  if (ret < 0)
    /* opening not succesful */
    { perror("creating pipe"); return; }
  /* copy the read pipe descriptor to a string */
  sprintf(pipe_str,"%id",pipe1[0]);
  /* fork a new process */
  switch (pid = fork())
    {
    case 0:
      /* the son : close the write pipe and execve
       * IRIT
       */
      close(pipe1[1]);
      execlp("myirit","myirit",pipe_str,NULL);
      return;
    case -1:
      /* fork not successful */
      perror("fork"); return;
    default:
      /*
       * the father: close the read pipe and start the command
       * loop
       */
      /* convert the write pipe to a file structure */
      file=fdopen(pipe1[1],"w");
      if (file == NULL)
        {
          perror("creating write file from filedescriptor\n");
        }
      /* close the read pipe */
      close(pipe1[0]);
      CommandLoop(sock);
      return;
    }                           /* end switch */
}
```

Irit-Kommandos

Die Kommandos, die Irit-CON versteht spalten sich in zwei Gruppen:

- Direkte Irit-Befehle
- Zusammengesetzte Kommandos

Direkte Irit-Befehle Für alle irit-Befehle existieren entsprechende Kommandos, so z.B. `IritCylinderCmd`, `IritRotXCmd` etc. Sie setzen ihre Argumente in Irit-Befehle.

Ein direkter Irit-Befehl

```
Command         *ConeFct(cmd)
Command         *cmd;
{
  Command *ret;

  if ((ret =check_command(cmd,ConeCmd,ConeNumArg)) != NULL)
    { return(ret); }
  else
    {
      /* construct the cone command */
      fprintf(file,"%s = cone(vector(0.0,0.0,%f),vector(0.0,0.0,%f),%f);\n",
              cmd->args[0].v.string_value,
              cmd->args[1].v.float_value,
              cmd->args[2].v.float_value,
              cmd->args[3].v.float_value);
      /* and send it */
      fflush(file);
      return (NULL);
    }
}
```

Ein besonderer direkter Befehl ist `IritSendMiscCmd`, der sein einziges Argument, einen String, unverändert an Irit weitergibt, und so beliebige Übertragungen an Irit ermöglicht.

Zusammengesetzte Kommandos Über die direkten Befehle hinaus existieren sogenannte zusammengesetzte Kommandos. Sie haben keine direkte Entsprechung in Irit, vielmehr setzen sie ihre Argumente in eine Reihe von Irit-Befehlen um. Damit kann die Funktionalität von Iritleicht erweitert werden.

Ein zusammengesetztes Kommando

```
Command *TrunconeFct(cmd) Command *cmd;
{
  float min,max,length,cone_length,offset;
  int dir;
  Command *ret;

  if ((ret =check_command(cmd,TrunconeCmd,TrunconeNumArg)) != NULL)
    { return(ret); }
  else
    {
      /* determine the min and max radii. If the right radius is greater, the direction must be
       * reversed, and the offset corrected */
      if (cmd->args[3].v.float_value > cmd->args[4].v.float_value)
        { /* the left radius is greater */
          min = cmd->args[4].v.float_value;
          max = cmd->args[3].v.float_value;
```

```
        offset = cmd->args[1].v.float_value;
        dir = 1;
      }
    else
      { /* the right one is bigger */
        min = cmd->args[3].v.float_value;
        max = cmd->args[4].v.float_value;
        offset = cmd->args[1].v.float_value+cmd->args[2].v.float_value;
        dir = -1;
      }
    /* the length of the cone is determined by the ratio of the radii and the length of the truncone */
    length = (cmd->args[2].v.float_value) *dir;
    cone_length = length * max /min;
    /* make the cone */
    fprintf(file,"%s = cone(vector(0.0,0.0,%f),vector(0.0,0.0,%f),%f);\n",
            cmd->args[0].v.string_value,
            offset,
            cone_length,
            max);
    if (dir > 0)
      /* due to a bug in Irit assure that the box has positive dimensions */
      { ...}
    else
      { fprintf(file,"H1= box(vector(%f,%f,%f),%f,%f,%f);\n",
                -(max+epsilon),
                -(max+epsilon),
                offset+cone_length + (dir *epsilon),
                2*(max+epsilon),
                2*(max+epsilon),
                dir * (cone_length-length+(dir*epsilon)));
      }
    /* subtract the box from the cylinder */
    fprintf(file,"%s = %s - H1;\n",
            cmd->args[0].v.string_value,
            cmd->args[0].v.string_value);
    /* free the box */
    fprintf(file,"free(H1);\n");
    /* and send it to Irit */
    fflush(file);
    return(NULL);
  }
}
```

Lisp-Kommandos

Auf Lisp-Seite können ebenfalls Erweiterungen der `Irit`-Funktionalität vorgenommen werden, indem Lisp-Funktionen entsprechende Folgen von Kommandoaufrufen absetzen.

Ein Lisp-Kommando

```
(defun make-compound-object (name operator subobjects &key destructive connection)
  (declare (special *connection*))
  (let ((*connection* (or connection *connection*)))
    (print name)
    (print operator)
    (print subobjects)
    (print *connection*)
    (when (notany #'null (list name operator subobjects *connection*))
          (print operator)
          (cond ((eq operator ':union)
                 (send-command 31 (list (format nil "~A = ~{ ~A~^ +~};"
                                                name subobjects))))
                ((eq operator ':difference)
                 (send-command 31 (list (format nil "~A = ~{ ~A~^ -~};"
                                                name subobjects))))
                ((eq operator ':cut)
                 (send-command 31 (list (format nil "~A = ~{ ~A~^ *~};"
                                                name subobjects)))))
          (when destructive
                (dolist (x subobjects)
                        (send-command 45 (list (format nil "~A" x))))))))
```

Außer Funktionen zur Erzeugung von Objekten werden auch Funktionen zur Manipulation der Viewing Matrix etc. bereit gestellt.

Integration in FRESKO

Die Integration in FRESKO zerfällt in drei Teile:

- Definition einer Unterklasse von `Application`
- Definition von Klassen zur Repräsentation von `Irit`-Objekten
- Bereitstellung einer interaktiven Benutzeroberfläche zur Erzeugung und Manipulation von `Irit`-Objekten

Die `Application`-Unterklasse `Irit-Application` `Irit-Application` schreibt fest, daß eine `Irit`-Anwendung immer in einem gesonderten Prozeß läuft. Außerdem definiert sie eine Methode für die generische Fuktion `run`.

`Irit`-Objekte als Instanzen von FRESKO-Klassen Die von `Irit` und der Lisp-Anbindung zur Verfügung gestellten Objekte bieten sich für eine Einbettung in FRESKO-Klassen an.

Insgesamt wurde die in Abb. 5.13 dargestellte Klassenhierarchie realisiert, sie kann aber leicht erweitert werden.

Abbildung 5.13: Die Klassenhierarchie der `Irit`-Objekte

Dabei ist die Abbildung der Objekttypen in FRESKO-Klassen kanonisch, einziges Problem ist die Konsistenzerhaltung der beiden Beschreibungen in Lisp und in `Irit`.

Dies ist ein allgemeineres Problem, das immer dann auftritt, wenn Daten redundant gespeichert sind. Zur Lösung dieses Problems bietet sich der Metaklassen-Mechanismus von FRESKO an. Es wurde eine Metaklasse `external-meta` geschaffen, die bei Slotzugriffen auf ein Objekt Methoden zum Schreiben bzw. Lesen des veränderten Objektes anstößt.

Subklassen dieser Metaklasse realisieren die konkreten Lese- bzw. Schreiboperationen. Bei der Metaklasse `irit-meta` kann die Lesemethode entfallen, bei Schreibenden Zugriff auf einen Slot wird das Objekt, falls seine Beschreibung vollständig ist, mit Hilfe der generischen Funktion `make-irit-object` übertragen.

Klassendefinition (Cone):

```
(create-class cone
              :super irit-object
              :instance-slots
              ((radius 0.0)
               (type nil)
               (length 0.0)
               (offset 0.0)))
```

Metaklassendefinition (external-meta):

```
(create-class external-meta
              :super function-class
              :meta function-meta
              :class-slots ((features '(:functional :external))
                            (requires ())))


(defmethod (slot-value :meta :around)
  ((frame external-meta) slot &optional facet)
  (when (hidden-slot-value frame '**modified**) (read-in frame))
  (run-super))

(defmethod (slot-value :meta :setf :around)
  ((frame external-meta) slot &optional facet) (value)
  (prog1
      (run-super)
    (write-out frame)))

(defmethod (write-out :meta)
  ((frame external-meta))
  (print-object frame))

(defmethod (read-in :meta)
  ((frame external-meta)))
```

Metaklassendefinition (irit-meta):

```
(create-class irit-meta
              :super external-meta)

(defmethod (write-out :meta)
  ((object irit-meta))
  (make-irit-object object))
```

Die generische Funktionmake-irit-object

```
(defmethod (make-irit-object :before) ((object cone))
  (with-object
   object
   (when (and (not (= #^radius 0.0)) (not (= #^length 0.0))
              (member #^type '(:ascending :descending)))
         (make-cone (name-of object)
                    (slot-value object 'offset)
                    (slot-value object 'length)
                    (slot-value object 'radius)
                    (slot-value object 'type)
                    :connection *irit-connection*)))
  object)
```

Eine interaktive Benutzeroberfläche Mit Hilfe des FRESKO-Paktetes WKS wird eine interaktive Benutzeroberfläche bereitgestellt, das es erlaubt Instanzen der `Irit`-Objekte zu erzeugen und zu modifizieren. Auch die Viewing Matrix und die Darstellung der Objekte kann so beeinflußt werden.

Abbildung 5.14: Die interaktive Benutzeroberfläche von `Irit`

5.3.3 Irit-Zusatzprogramme

Zusammen mit `Irit` werden eine Reihe von Zusatzprogrammen geliefert, die in erster Linie der Verbesserung der Darstellungsqualität dienen.

Es sind dies ein Programm zur interaktiven Darstellung von `Irit`-Objekten namens `poly3-d`, ein Programm zur Entfernung verdeckter Linien (`poly3d-h`) und ein Programm zur „realistischen" Darstellung des Objektes (`poly3d-h`).

poly3d `Irit` besitzt die Möglichkeit, erzeugte Objekte mit Hilfe der Befehle `GDump` und `MDump` abzuspeichern. Das Speicherformat ist eine polygonale Beschreibung des Objektes, d. h. eine Beschreibung der begrenzenden Polygone. Sie kann nur bedingt wieder in `Irit` eingeladen werden.

Zur Darstellung dieser polygonal beschriebenen Objekte dient das Programm `poly3d`. Auch zum Aufruf dieser Anwendung wurde eine Einbettung in Lisp und FRESKO realisiert, so daß `poly3d` von der interaktiven Oberfläche aus gestartet werden kann.

poly3d-h `poly3d-h` ist ein Batchprogramm zur Entfernung verdeckter Linien in polygonalen Beschreibungen. Es erhält als Eingabe eine solche Beschreibung und erzeugt als Ausgabe eine veränderte Beschreibung in der ganz verdeckte Linien entfernt und teilweise verdeckte Linien entsprechend gekürzt werden.

Bei der Einbettung in Lisp und FRESKO wurde zur Umlenkung der Ausgabe in eine Datei die Option `stdout-redirection` verwendet. Die veränderte Darstellung wird wiederum mit `poly3d` visualisiert.

poly3d-r `poly3d-r` ist ebenfalls ein Batchprogramm, das eine polygonale Beschreibung als Eingabe erhält. Es erzeugt eine „realistische" Darstellung des Objektes. Die Darstellung ist insoweit „realistisch", als daß nicht nur die begrenzenden Kanten des Objektes dargestellt werden[20], sondern das Objekt flachig und schattiert visualisiert wird. Die Darstellung kann über Angabe von Parametern wie Position und Anzahl

[20] wire-frame Darstellung

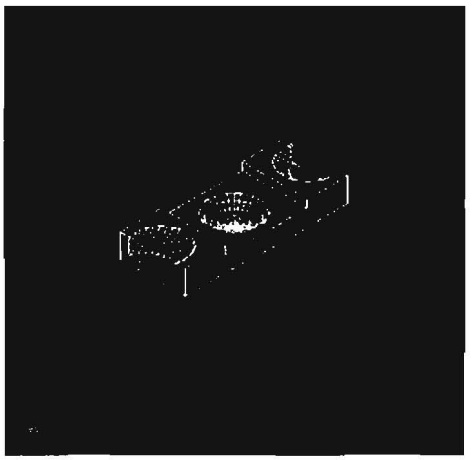


Abbildung 5.15: Ein Objekt nach Entfernung der verdeckten Linien

der Lichtquellen beeinflußt werden. Die Ausgabe erfolgt im GIF[21]-ormat, einem Standard zur Speicherung mehrfarbiger pixelbasierter Bilder[22].

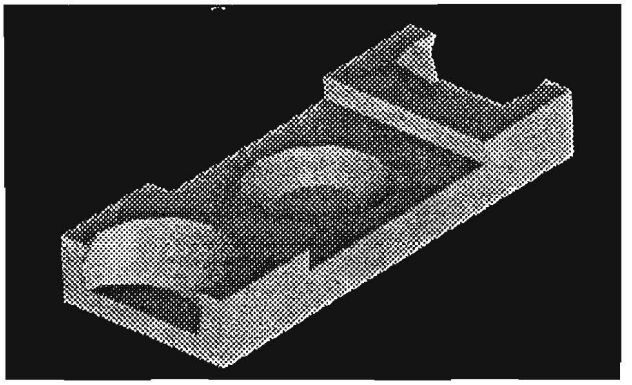


Abbildung 5.16: Ein schattiertes Objekt

xloadimage `xloadimage` ist ein X-Client der in der Lage ist, pixelbasierte Bilder unterschiedlichster Herkunft sowohl auf Monochrom als auch auf Farbbildschirmen darzustellen. Es wird dazu benutzt, die mit `poly3d-r` erzeugten Bilder anzuzeigen.

5.4 Integration von Datenbanken

In diesem Abschnitt soll kurz auf die Möglichkeit der Koplung von Datenbanken und Lispsystemen eingegangen werden. Betrachtet wird der Anschluß von `INGRES` und `POSTGRES`, zwei in Berkeley entwickelten Datenbanken, die daher kurz vorgestellt werden sollen.

`INGRES`

`INGRES` ist eine kommerziell vertriebene relationale Datenbank, die aus der in der in den siebziger Jahren an der Universität von Berkeley entwickelten Datenbank University Ingres hervorgegangen ist.

[21] Graphics Interchange Format

[22] Leider kann das GIF-Bild hier nicht in voller Qualitat wiedergegeben werden

Anfragen an INGRES werden in der Sprache QUEL oder SQL gestellt[23], als Ergebnis erhält man die qualifizierten Tupel.

POSTGRES

POSTGRES ist die Weiterentwicklung von INGRES an der UCB[24]. Ziel von POSTGRES war es relationale Datenbanken in ihrer Ausdrucksmächtigkeit zu erweitern. Als neue Konzepte kamen u. a. die Möglichkeit der Vererbung und Definition von Funktionen hinzu[25]. So wird POSTGRES von einigen Leuten als erweiterte relationale Datenbank von anderen schon als semantische oder objektorientierte Datenbank gesehen. Anfragen an POSTGRES werden in POSTQUEL einer Erweiterung von QUEL formuliert, so daß alle QUEL-Anfragen unverändert benutzt werden können.

Der Anschluß der Datenbanken kann wiederum auf unterschiedlichen Ebenen erfolgen, die im nachfolgenden skiziiert werden sollen.

5.4.1 String-Verbindung

Die wohl einfachste Art der Anbindung ist eine Kopplung auf der Basis der Stringübermittlung:

Die QUEL(SQL,POSTQUEL)-Anfrage wird als String an die Datenbank gesandt, das Ergebnis kommt auf dem gleichen Wege zurück. Dieses Verfahren ist problemlos mit Hilfe der entsprechenden Basisnachricht zu realisieren, auf C-Seite ist nur eine kleine Anwendung zu implementieren, die die Anfragen mit Hilfe der von der Datenbank definierten Schnittstellenfunktionen an die Datenbank weiterleitet.

Sowohl INGRES als auch POSTGRES sind aber auch schon von sich aus für den Netzwerkweiten Einsatz vorbereitet und es existieren Integrationen in Allegro und Lucid[26]. Eine direkte Verwendung dieser Integrationen resultiert allerdings in einer uneinheitlichen Architektur wie in 5.1.2, die ja gerade vermieden werden sollte.

5.4.2 Lose Integration in FRESKO

Diese Art der Anbindung bietet sich für Datenbestände an, die sich entweder sehr langsam ändern oder bei denen Änderungen unkritisch sind. Gekennzeichnet ist sie durch eine Art Polling, bei der das Arbeitsplanungssystem zu definierten Zeitpunkten (z. B. zum Systemstart) die Daten aus dem datenbanksystem abruft und in `fresko`-Objekten abspeichert. Dazu wird die Metaklasse `external-meta` verwandt, indem bei denFRESKO-Klassen entsprechende `read-from` Methoden definiert werden.

5.4.3 Enge Kopplung mit FRESKO

Die hier skizzierte Kopplung erfordert den höchsten Implementierungsaufwand, sie wurde in [Row] vorgestellt, aber nur ansatzweise realisiert[27] Schwierigkeiten macht im wesentlichen das Fehlen einer klaren Metaklassendefinition in CLOS, so daß eine Implementierung in FRESKO leichter fällt, und in Anlehnung an den vorhandenen Source-Code erfolgen kann[28].

Es wird eine „shared object hierarchy“ geschaffen, die gewährleistet, daß CLOS(FRESKO)-Objekte von mehreren Benutzern und Anwendungen geshared werden können. Das heißt, mehrere Benutzer können konkurrent

[23] Die Hintergründe dieser Zweisprachigkeit und eine Übersicht über aktuelle Datenbankentwicklungen werden in dem lesenswerten Artikel [Sto] erläutert

[24] POSTGRES ist via anonymen FTP von postgres.berkeley.edu () und in Europa von () erhältlich.

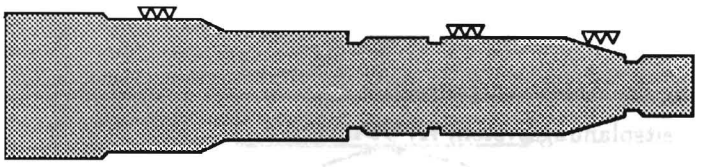
[25] Einen Überblick über den Umfang der Erweiterungen liefert [SR86], interessant für den Entwickler von KI-Programmen ist aber auch [SRH], in dem u. a Gründe für die Abkehr von der ursprünglich vorgesehenen Implementierungssprache COMMONLISP angegeben werden.

[26] Diese Integrationen (namens `libq` und `libpq`) sind Teil von PICASSO einem ebenfalls an der UCB entwickelten System (auch per anonymen FTP erhältlich).

[27] Mit PICASSO wird eine nicht funktionsfähige CLOS-POSTGRES Version geliefert, es soll aber auch eine funktionierende INGRES-Kopplung existieren, deren Source-Code aber nicht verfügbar ist.

[28] Die Realisierung bietet sich somit für eine Projekt- oder Diplomarbeit an.

auf die Objekte zugreifen und sie modifizieren, es bleibt eine Konsistente Sicht erhalten. Dazu müssen Änderungen an den Objekten bzw. das Löschen und Erzeugen von Objekten den anderen Benutzern mitgeteilt werden. Dies geschieht über Invalidierungsflags. Die Speicherung der Objekte unde der darauf definierten Methoden erfolgt im Datenbanksystem, das entsprechende Dictionaries bereithalten muß (siehe [Row]). Es garantiert auch, daß ein Objekt zu einem Zeitpunkt nur von einer Anwendung verändert werden kann. Zur Beschleunigung der Anfragen werden Objektpuffer benutzt, in die vorausschauend Objekte geladen werden.



A | Die Systeme im Überblick

A.1 Einleitung

In den nachfolgenden Abschnitten wird ein Überblick über wissensbasierte Systeme zur generativen Arbeitsplanung gegeben.

Bei den beschriebenen Systemen handelt es sich um Prototypen zur Fertigung mechanischer Teile. Dabei variieren die Systeme stark in ihrem Funktionsumfang. So reicht das Teilespektrum von Drehteilen bis zu prismatischen Teilen, die verwendeten Fertigungsverfahren beschränken sich im wesentlichen auf das Spanen mit geometrisch bestimmter oder unbestimmter Schneide. Ein ähnlich weites Spektrum trifft man bei den verwendeten KI-Techniken und Methoden an. Hier liegt jedoch ein Schwerpunkt bei der Verwendung von Produktionsregeln zur Repräsentation des Planungswissens.

Die einzelnen Systeme werden durch eine Zusammenfassung der jeweiligen Literatur kurz beschrieben. Eine Wertung unterbleibt, die verwendeten Begriffe werden übernommen. Durch das Fehlen einer einheitlichen Begriffsbildung und den unterschiedlichen Hintergrund der Autoren und Systeme kann es daher zu Begriffsduplizitäten und zu ungewohnten Verwendungen bekannter Begriffe kommen.

Um die Systeme leichter gegenüberstellen zu können erfolgt die Beschreibung stichpunktartig. Die Auswahl der Stichpunkte erfolgte in Anlehnung an [JF89], wurde aber leicht erweitert und verändert. Punkte die nach dem Studium der Literatur unklar blieben wurden mit ??? gekennzeichnet.

Die Vorstellung der einzelnen Systeme erfolgt in alphabetischer Reihenfolge und impliziert keinerlei Bewertung.

A.2 AVOGEN

Anwendungsgebiet
Generierung von Arbeitsvorgangsfolgen für rotationssymmetrische Werkstücke

Funktionsumfang

- Definition des Werkstücks
- Wahl der Aufspannung und Bestimmung der Arbeitsvorgangsfolge
- Kopplung mit dem konventionellen Arbeitsplanungssystem AVOPLAN

Teilebeschreibung
Die Definition des Teiles erfolgt mit Hilfe der Werkstückbeschreibungssprache DREBES. Das Teil wird als eine Menge von sogenannten Haupt- und Nebenformelementen beschrieben[1] . Die Übernahme von Informationen aus CAD-Daten ist möglich.

Werkstattbeschreibung
Die Werkstattbeschreibung ist in ein Operator- und ein Maschinenmodell zweigeteilt.

- Operatorenmodell
  Im hierarchischen Operatorenmodell werden weitgehend betriebsneutrale Fertigungselemente beschrieben. Ein Operator enthält Angaben über mit ihm fertigbare Formelemente und zu benutzende Fertigungsverfahren.
  Die Operatoren sind aufgrund der Struktur von AVOGEN invers definiert, d. h. sie tragen Material auf statt ab.
- Maschinenmodell
  Das Maschinenmodell enthält Daten über die betriebsspezifische Maschinenausstattung mit Verweisen auf die an einer Maschine benutzbaren Operatoren.

Repräsentation des Planungswissens
Repräsentation durch modularisierte Regelbasen. Diese dienen dem Fällen von Entscheidungen, wie z. B. :

- Bestimmung einer neuen Aufspannung
- Auswahl eines Fertigungsmerkmals und eines darauf anwendbaren Operators.

Vorgehensweise
Die Planung erfolgt vom Fertigteil zum Rohteil mit Hilfe des Situationskalküls. Dabei entsprechen den Situationen die Zustände des Werkstücks und den Übergängen die Anwendung eines Operators.
Die einzelnen Schritte sind:

1. Abarbeitung prismatischer Formelemente
2. Auflösung komplexer Toleranzbeziehungen zwischen Formelementen
   Dieser Schritt wird notwendig, da Toleranzbeziehungen einen wesentlichen Einfluß auf die Reihenfolge der Bearbeitung haben können.
3. Abarbeitung rotationssymmetrischer Formelemente

Verwendete KI-Methoden
Verwendung der Expertensystemshell Knowledge Craft. Dabei werden die zur Verfügung gestellten Mechanismen wie Frames, Regeln und chronologisches Backtracking genutzt.

Systemarchitektur
siehe Bild A.1 auf Seite 95

Weitere Vorhaben

- Erhöhung der Performanz

[1] Grob können Haupt- und Nebenformelementen dadurch unterschieden werden, daß Hauptelemente materialhaltig (z. B. Zylinder, Kegel), Nebenformelemente dahingegen materiallos sind (z. B. Nut, Fase).

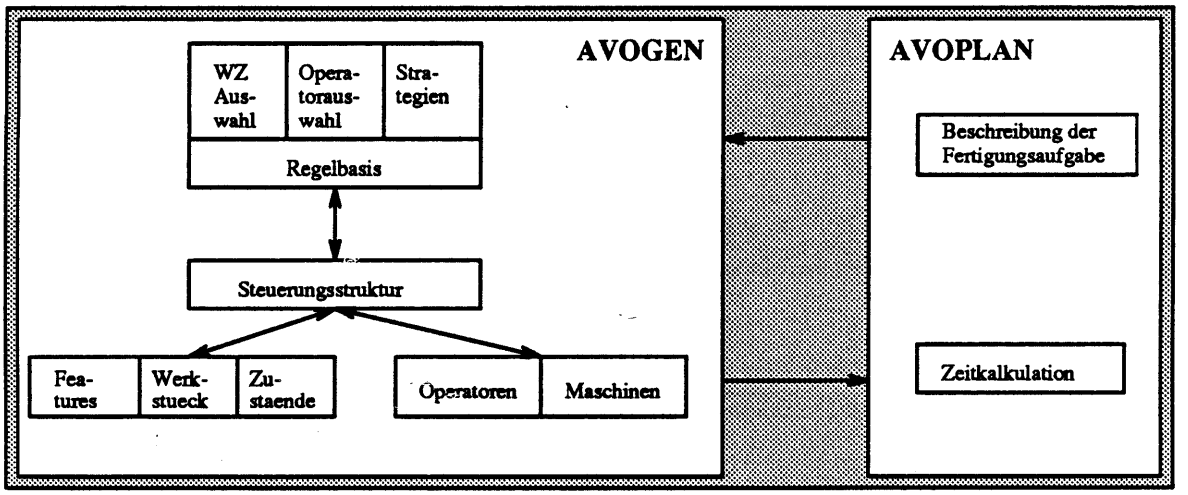


Abbildung A.1: Systemarchitektur AVOGEN

- Übergang zur Wartung der Wissensbasis durch den Arbeitsplaner

Implementierung
In Knowledge Craft auf einer μVax 3200

Umfang
???

Literatur
[ASP90]

A.3 CAMEX

Anwendungsgebiet
Arbeitsplanung prismatischer Teile, die in 2 1/2[2] Dimensionen beschreibbar sind.

Funktionsumfang

- Featureextraktion aus CAD-Daten
- Rekonstruktion der 3-D Struktur des Teils
- Generation des Arbeitsplanes

Teilebeschreibung
Das Teil wird als eine Liste von geometrischen Primitiven, die die zugrundeliegenden Strukturen, wie z. B. Tasche, Profil, Loch usw. widerspiegeln, beschrieben.

Es werden zwei Arten von Primitiven unterschieden:

1. Oberflächen
2. Subteile des Teils. Diese zerfallen wiederum in zwei Typen:
   - Materialhaltige Teile
   - Materiallose Teile (Löcher)

Zusätzlich können qualitative geometrische Relationen zwischen den Primitiven wie is-above, is-below definiert werden.

Werkstattbeschreibung
???

Repräsentation des Planungswissens
In einzelne Module gruppierte Produktionsregeln. Die Eingabe dieser Regeln erfolgt in „strukturiertem Englisch", das automatisch in LISP-Code transformiert wird.

Kontrollregeln dienen zur Steuerung des Ablaufes. Im wesentlichen werden dabei Modulwechsel durchgeführt.

Vorgehensweise
In 4 Schritten :

1. Identifizierung von Bereichen
   Bestimmung einer Aufteilung des zu zerspanenden Materials in Bereiche. Aus der Menge der möglichen Aufteilungen wird eine ausgewählt. Dazu wird primär eine Tiefensuche ausgeführt. Constraints realisierende Regeln schränken dabei den Suchraum ein.
2. Erzeugung eines Teilplanes für jeden Bereich.
3. Werkzeugoptimierung
   Wahlweise Optimierung in Bezug auf die Anzahl der benutzten Werkzeuge, der Fertigungszeit etc.
4. Festlegung einer Operationsreihenfolge durch zwei Arten von Constraints zwischen den Operationen:
   - must-be
   - should-be

Verwendete KI-Methoden
Produktionsregeln, Erklärung von How- und Why-Fragen

Systemarchitektur
???

[2] 2 1/2 Dimensionen bedeutet in diesem Zusammenhang, das das Teil eine konstante Dicke aufweist, und nur die Features unterschiedliche Tiefen aufweisen

Weitere Vorhaben
???

Implementierung
In FranzLisp auf Apollo unter Unix

Umfang
ca. 12000 Zeilen Lisp und ca. 3000 Zeilen C Code

Literatur
[EZBB87]

A.4 CHAMP

Anwendungsgebiet
Arbeitsplanung und NC-Programmierung einfacher Teile (Makrofolgegenerierung)

Funktionsumfang

- Maschinen- und Rohmaterialauswahl
- Ermittlung der Teilarbeitsvorgangsfolge
- Erstellen eines Makrofolgeplans für eine CNC-Werkzeugmaschine unter Verwendung einer Makrobibliothek

Teilebeschreibung
Als Wire-Frame Beschreibung die aus einer IGES[3]-Datei mit Hilfe eines Kopplungsbausteins gewonnen wird und zusätzliche technologische Informationen enthält.

Werkstattbeschreibung
???

Repräsentation des Planungswissens

- Produktionsregeln zur Makroauswahl
- Metaregeln zur Kontrollflußsteuerung

Vorgehensweise
In drei Schritten:

1. Zurückgewinnung eines Drahtmodells aus der IGES-Datei
2. Festlegung der Teileart
3. Generierung des Arbeitsplanes als Makrofolge mit Hilfe der Produktionsregeln.

Verwendete KI-Methoden

- Forward-chaining
- Forward-reasoning
- chronologisches Backtracking

Systemarchitektur
siehe Bild A.2 auf Seite 99

Weitere Vorhaben
Erweiterung der Schnittstelle zu CAD durch Übergang zu STEP[4]-Dateien

Implementierung
In C auf einer VAX unter VMS

Umfang
???

Literatur
[Mie88, JF89]

[3] **IGES** – International Graphics Exchange Standard
[4] **STEP** – Standard for the Exchange of Product Model Data

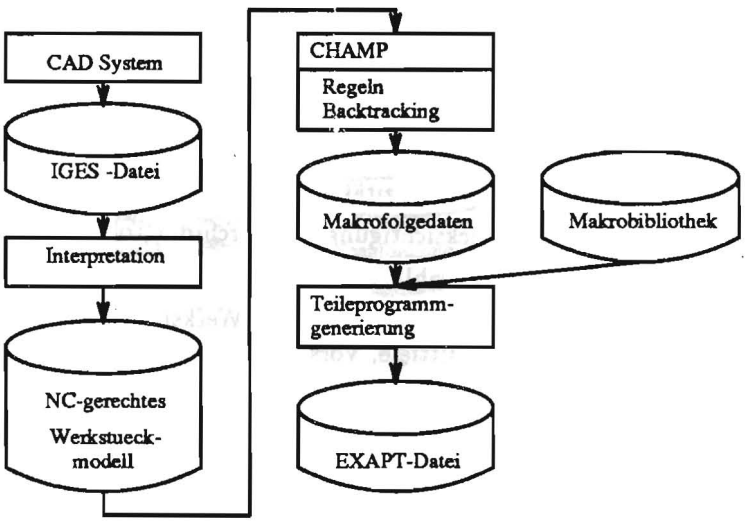


Abbildung A.2: Systemarchitektur CHAMP

A.5 Expertensystem zur spanenden Bearbeitung

Anwendungsgebiet
Außendrehen von Stahlwerkstoffen mit einseitig abfallender Kontur

Funktionsumfang
Durchgängiges System, von der Rohteil- und Fertigteileingabe bis zur Schnittaufteilung und Ermittlung der Werkzeugverfahrwege.

Teilebeschreibung
Das Teil wird durch interaktive Eingabe der Geometrie- und Technologiedaten erzeugt und in diesen Termini beschrieben. auf eine CAD-Kopplung wurde bewußt verzichtet, da dann nach Meinung der Autoren eine Festlegung auf ein bestimmtes CAD-System notwendig gewesen wäre.

Werkstattbeschreibung
Repräsentation von Maschinen, Spannmitteln, Werkzeugen und Wendeschneidplatten mit Hilfe von Frames.

Dabei erfolgte eine spezifische Ausrichtung auf das Werkzeugsystem (WIDAFLEX) des eigenen Hauses.

Repräsentation des Planungswissens

- Verschiedene Wissensbasen für die einzelnen Aufgabenteile.
- Ausnutzung der Mechanismen der benutzten Expertensystemshell.

Vorgehensweise
In 5 Schritten:

1. Maschinenauswahl nach benötigter Kapazität, Arbeitsraum und Leistung.
2. Wahl der Aufspannung unter Berücksichtigung der durch die Toleranzen erzeugten Constraints.
3. Schnittaufteilung und Werkzeugauswahl
   (a) Planung des Vorschruppens je nach Zustand der Werkstückoberfläche. Wahl des Werkzeugs und der Schnittparameter (Schnittiefe, Vorschub, Schnittgeschwindigkeit)
   (b) Planung der weiteren Bearbeitungszyklen (Schlichten, etc.) in ähnlicher Art und Weise
4. Graphische Simulation des Arbeitsplanes
5. Möglichkeit der Erklärung des Arbeitsplanes mit Hilfe von Regelbäumen.

Verwendete KI-Methoden
Benutzung der Expertensystemshell KEE und der dort bereitgestellten Mechanismen.

Systemarchitektur
siehe Bild A.3 auf Seite 101

Weitere Vorhaben
Mehrstufige Erweiterung des Systems auf den gesamten Bereich der spanenden Bearbeitung:

1. Erweiterung des Außendrehens zur Bearbeitung beliebiger Konturen mit Hilfe von Konturmakros, wie z.B. Fase, Einstich, Freistich, Gewinde.
2. Hinzunahme des Innendrehens unter Beachtung des geringeren Bewegungsspielraums.
3. Integration von Bohroperationen.
4. Berücksichtigung von Fräsoperationen mit Übergang zu dreidimensionalen Beschreibungen.

Implementierung
Mit Hilfe von KEE auf einer Symbolics–Lispmaschine

Umfang
???

Literatur
[PST88]

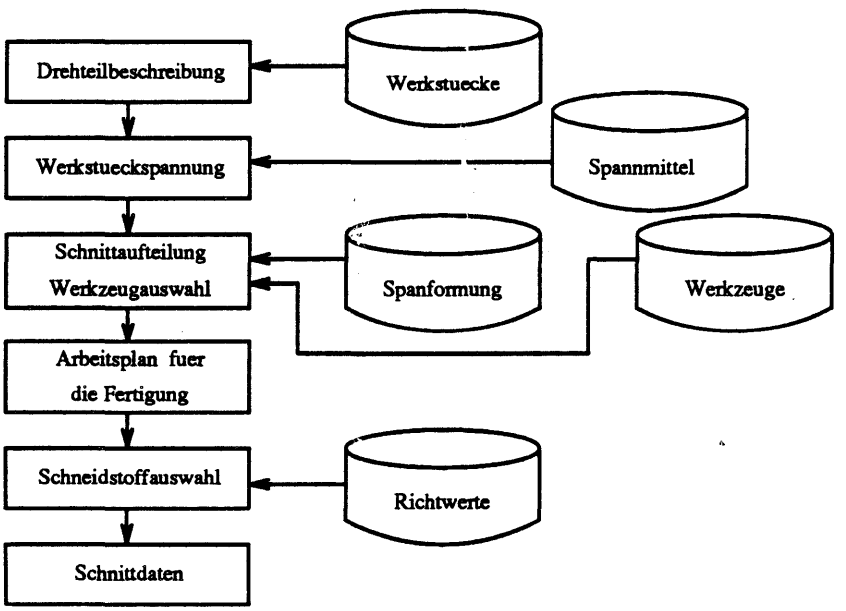


Abbildung A.3: Systemarchitektur des Expertensystems zur spanenden Bearbeitung

A.6 FEXCAPP

Anwendungsgebiet
Arbeitsplanung prismatischer Teile

Funktionsumfang

- Featurecognition aus CAD-Daten
- Bestimmung der Werkzeuge und Bearbeitungsfolge durch Baumsuche
- NC-Programmgenerierung
- NC-Bearbeitung

Teilebeschreibung
Featureorientierte Beschreibung, die durch eine Extraktion der Features aus der internen BREP[5]-Darstellung des geometrischen Modellierers gewonnen wird.
Ein Feature wird durch Anfangs- und Endfläche und die Bearbeitungsrichtung definiert.

Werkstattbeschreibung
Das System ist auf eine bestimmte Maschine zugeschnitten.

Repräsentation des Planungswissens
Mit OPS5-Produktionsregeln

Vorgehensweise

1. Featureerkennung
   (a) Extraktion der DEDS (Directed Edge Data Structure)
   Die DEDS unterscheidet sich insoweit von der internen Darstellung des Modellierers, indem sie einen festen Umlaufsinn auf den Begrenzungsflächen definiert.
   (b) Bestimmung des AAG's (Attribute Adjacency Graph) des Teiles
   Der AAG entsteht aus der DEDS unter Berücksichtigung der Nachbarschaftsbeziehungen zwischen den Flächen. Im AAG entsprechen die Knoten den Flächen und die Kanten den Begrenzungskanten der Flächen des Teiles. Jeder Kante wird ein Typ zugeordnet.
   (c) Ermittlung der im Teil enthaltenen Features
   Jedes erkennbare Feature besitzt einen charakteristischen AAG, so daß ein Feature erkannt ist, wenn ein Subgraph des AAG's des Teiles mit dem AAG des Features gematcht werden kann.
2. Bestimmung der Operationsreihenfolge
   (a) Grobunterscheidung in Fräs- und Schneideoperationen
   Die Fräsoperationen geben dem Teil sein charakteristische Form, während die Features mit den Schneideoperationen herausgearbeitet werden.
   In FEXCAPP werden prinzipiell zuerst die Fräsoperationen ausgeführt.
   (b) Bestimmung der Aufspannungen über die Bearbeitungsrichtungen der Features
   Gruppierung der Operationen die in einer Aufspannung gefertigt werden können.
   (c) Bestimmung der Reihenfolge der Operationen in einer Aufspannung durch Baumsuche.
   Es wird ein Baum aufgebaut der alle möglichen Operationsfolgen enthält. In einem ersten Schritt werden dann alle Knoten des Baumes anhand der in Tabelle A.1 wiedergegebenen Skala bewertet.
   Danach wird der optimale Pfad durch „Hillclimbing" bestimmt.
3. Bestimmung der Schnittdaten
   Entweder über eine Schnittdatendatenbank oder durch eine geschlossene mathematische Formel.

[5]BREP (Boundary Representation) bedeutet, daß der Modellierer die Daten intern als eine Menge von Flächen, Kanten und Punkten repräsentiert. Dabei werden die Flächen durch Zeiger auf die Begrenzungskanten und diese durch Zeiger auf ihre Endpunkte definiert. Eine weitere Art der Datenhaltung eines geometrischen Modellierers ist die sogenannte CSG-Darstellung (Constructive Solid Geometry). Dort wird das Modell aus einer Menge von Grundkörpern mit Hilfe Boole' scher Operationen aufgebaut.

Bewertung	Wird das gleiche Werkzeug wie beim Vorgänger verwandt	Ist das Feature mit dem Vorgänger benachbart	Erzeugt der Vorgänger die Anfangsfläche des Features
1	ja	ja	ja
2	ja	ja	nein
3	ja	nein	ja
4	ja	nein	nein
5	nein	ja	ja
6	nein	ja	nein
7	nein	nein	ja
8	nein	nein	nein

Tabelle A.1: Bewertungstabelle des FEXCAPP-Systems

4. Fertigung des Teiles

Verwendete KI-Methoden
Verwendung von OPS5

Systemarchitektur

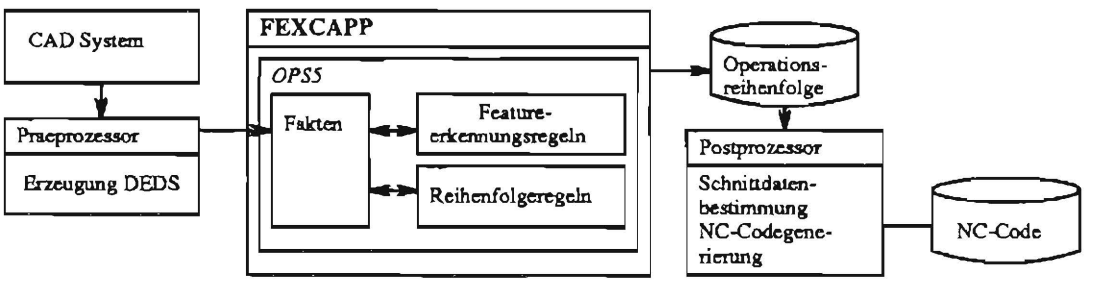


Abbildung A.4: Systemarchitektur FEXCAPP

Weitere Vorhaben

- Erweiterung des Spektrums der erkennbaren Features
- Verbesserung des heuristischen Wissens

Implementierung

Modellierer: ROMULUS auf einer VAX 11/750

Preprocessor: In Fortran77 (aus Kompatibilitätsgründen mit ROMULUS) auf einer VAX 11/750

FEXCAPP: In Sierra OPS5 auf einem IBM PC/AT

Postprocessor: In TurboPascal auf einem IBM PC/AT

Umfang

- 147 Regeln, davon 112 zur Featureerkennung und 35 zur Reihenfolgebestimmung.
- 11 Working-Element Klassen.

Literatur
[LLL89]

A.7 GARI

Anwendungsgebiet
Teilefertigung (Zerspanung) prismatischer Teile und Drehteile

Funktionsumfang

- Maschinenauswahl
- Planung der Arbeitsvorgangsfolge

Teilebeschreibung
Interaktiv erzeugte Beschreibung des Teiles als Basisvolumen plus einer Menge von Features

Werkstattbeschreibung
Beschreibung der vorhandenen Maschine über Eigenschaften wie Genauigkeit etc.

Repräsentation des Planungswissens
Produktionsregeln mit getypten Variablen zur schrittweisen Eingrenzung des Suchraums (wird in [JF89] mit constraint propagation bezeichnet).

Dabei haben Bedingungs- und Aktionsteil das folgende Aussehen:

Bedingungsteil: Der Bedingungsteil ist eine Konjunktion elementarer Constraints

Aktionsteil: Der Aktionsteil ist eine gewichtete Kombination sogenannter 'pieces of advice'.

Vorgehensweise

- Iterative Verfeinerung des Planes unter Beachtung von Reihenfolgebedingungen.
- Planung auf drei Hierarchiestufen: Phase, Operation, Vorgang.
- Feste Menge von Aktionen durch die Festlegung eines Schrupp- und eines Schlichtschrittes pro Feature.
- Konflikterkennung und -lösung durch selektives Backtracking.
- Bewertung der Lösungen als Summe der verwendeten 'pieces of advice'.

Verwendete KI-Methoden
Forward-chaining, gemischtes Forward- und Backward reasoning

Systemarchitektur
???

Weitere Vorhaben
???

Implementierung
Maclisp auf HB-68 unter Multics

Umfang
ca. 50 Regeln

Literatur
[DL81, JF89]

Die Syst

A.8 Generatives Arbeitsplanungssystem mit CAD-Kopplung

Anwendungsgebiet
Teilefertigung prismatischer Teile für den Flugzeugbau

Funktionsumfang

- Featureerkennung aus CAD-Daten und Konvertierung in eine technologieorientierte Teilebeschreibung
- Erzeugung vollständiger Arbeitspläne mit Reihenfolge der Operatoren, Maschinen- und Werkzeugauswahl

Teilebeschreibung Automatische Erzeugung einer featureorientierten Beschreibung durch Erkennung vordefinierter Featuretypen, wie z. B. Bohrungen, Durchsetzungen und Erhebungen. Dabei werden technologische Daten soweit möglich automatisch extrahiert, nicht vorhandene Daten werden interaktiv erfragt.

Werkstattbeschreibung
Beschreibung der Maschinen und Werkzeuge als Frames

Repräsentation des Planungswissens
Als Produktionsregeln und in 2 Arten von Prozeduren:

- als Aktivitäten für die Ebene der Problemlösung und Kontrolle
- als Aktionen auf der Ebene der Berechnungen und der Manipulation von Objekten und ihren Attributen.

Vorgehensweise

1. Grobanalyse des Teils mit Maschinenauswahl und Vorauswahl der Fertigungsprozsesse.
2. Bestimmung der Operationen und ihrer Reihenfolge für die Vorbereitung, Bearbeitung und Endbearbeitung des Teiles.
   (a) *Vorbereitung*
   Teilplanerzeugung für die Bereitstellung und Aufspannung der Rohteile
   (b) *Bearbeitung*
   Erzeugung eines Teilplanes für jedes Feature, bzw. jeder Gruppe identischer Features unter Beachtung von Constraints
   (c) *Endbearbeitung*
   Teilplanerzeugung zur Oberflächenbehandlung, Markierung und Qualitätskontrolle
3. Zusammenfügen der Teilpläne zum Gesamtplan

Verwendete KI-Methoden

- Einsatz eines spezifischen Werkzeugs, des Multi-Method-Planners.
- Bereitstellung von
  1. *Wissensrepräsentationsformalismen*
  Regeln, Frames, Constraints sind im MMP definiert.
  2. *Methoden*
  Vorwärts- und Rückwärtsregelinterpreter, Hierarchische- und Skelettplanung.
  3. *Kontrollstrategien*
  Durch Verwendung eines Kontrollspracheinterpreterund eines Agendamechanismus
- Im Hintergrund Verwendung eines ATMS zur Konsistenzerhaltung und Verwaltung von Alternativen

Systemarchitektur
siehe Bild A.5 auf Seite 106

Weitere Vorhaben
Erweiterung des Systems zur Bearbeitung anderer Teilespektren

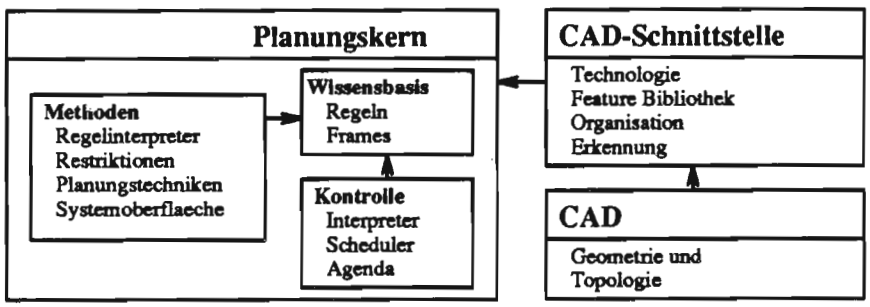


Abbildung A.5: Systemaufbau des generativen Arbeitsplanungssystems mit CAD-Kopplung

Implementierung
Auf einer VAX-Station unter VMS

Umfang
ca. 200 Frames, ca. 400 Regeln

Literatur
[IS89]

A.9 GENOA

Anwendungsgebiet
Arbeitsplanung im Maschinenbau

Funktionsumfang
Erweiterung eines entscheidungstabellenbasierten Systems um eine Komponente zur Optimierung der Arbeitsgangreihenfolge.

Teilebeschreibung
Generierung der Beschreibung innerhalb des CAD-Systems durch CAD-Makros. Dadurch wird über die geometrische Information hinaus auch funktionale und technologische Information enthältbereitgestellt.

Werkstattbeschreibung
Werkzeuge und Maschinen werden innerhalb des benutzten Entscheidungstabellensystems beschrieben.

Repräsentation des Planungswissens
Zweigeteilet in den einzelnen Komponenten des Systems:

- Entscheidungstabellen zur Bestimmung des Werkzeugs für jedes Feature, und zur Erkennung von Präzedenzen zwischen den Features.
- Bereitstellung einer Methodenbank zur Optimierung der Arbeitsgangreihenfolge.

Vorgehensweise Als dreistufiger Prozeß mit der folgenden Unterteilung:

1. Konstruktion des Teiles mit Hilfe von CAD-Makros, Bereitstellung der benötigten technologischen Informationen.
2. Operationsplanung mit einem Entscheidungstabellensystem, das sie folgenden Aufgaben löst:
   (a) Ermittlung der einzelnen Arbeitsgänge
   (b) Festlegung der Fertigungsverfahren, Werkzeuge, Betriebsmittel und Vorgabezeiten
3. Optimierung der Operationsreihenfolge mit Hilfe der Methodenbank
   In dieser werden Algorithmen zur Ermittlung der „optimalen" Operationsfolge relativ zu einer gegebenen Kostenfunktion bereitgestellt.
   Momentan sind in der Methodenbank Verfahren zur uniformierten und informierten Graphensuche enthalten.

Die einzelnen Komponenten werden über eine relationale Datenbank integriert.

Verwendete KI-Methoden
Innerhalb der Methodenbank Bereitstellung heuristischer Graphsuchverfahren (A*-Algorithmus etc.).

Systemarchitektur
siehe Bild A.6 auf Seite 108

Weitere Vorhaben
Bei Verfügbarkeit verbesserter CAD-Systeme verstärkter Einsatz wissensbasierter Systeme zur Operationsplanung.

Implementierung

Konstruktion: AUTOCAD auf PC unter MS-DOS

Arbeitsplanung: Auf Apollo unter UNIX mit folgenden Komponenten:

- Entscheidungstabellensystem ENGIN
- Methodenbank in C
- Benutzeroberfläche X,MOTIF
- Datenbank INGRES

Umfang
???

Literatur
[FHG90, gen90, JF89]

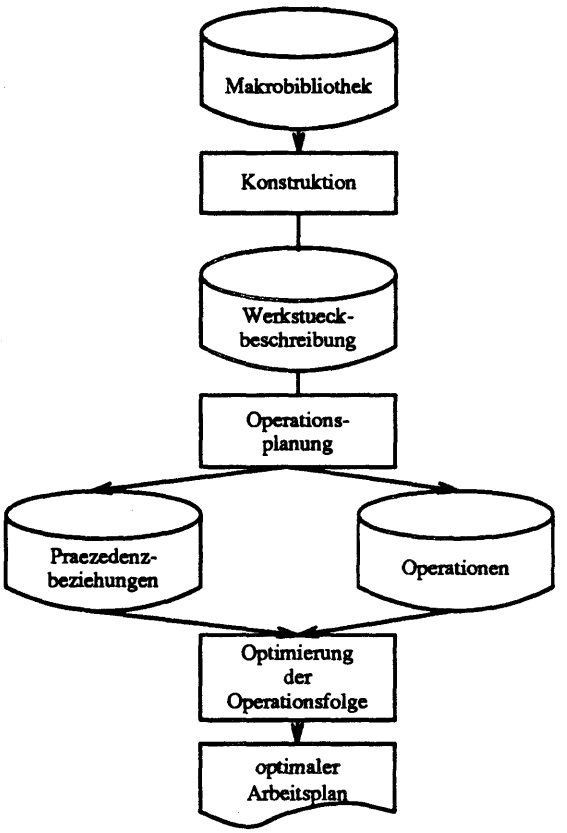


Abbildung A.6: Systemarchitektur GENOA

A.10 GPPS

Anwendungsgebiet
Arbeitsplanung für Drehteile

Funktionsumfang

- Erzeugung der Verfahrens- und Operationspläne für Drehteile
- Entwicklung einer Topologie der Herstellungsverfahren als Grundlage des Systems. In dieser Topologie sind sämtliche Parameter des Herstellungsprozesses beschrieben und strukturiert.[6]
- Integration in ein komplexes CAD/CAPP/CAM-System (siehe Bild A.7)

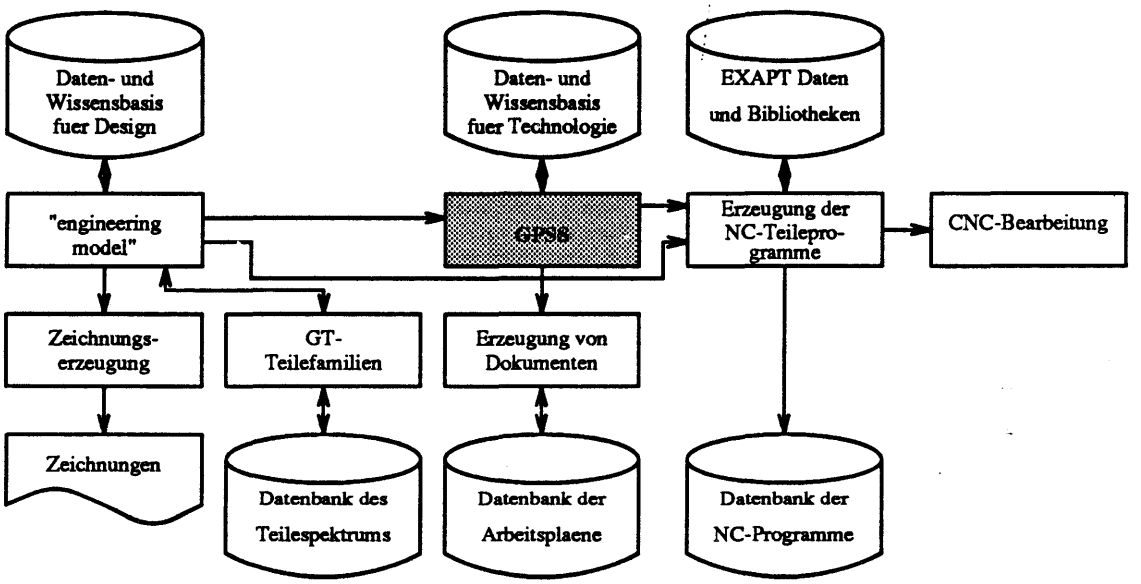
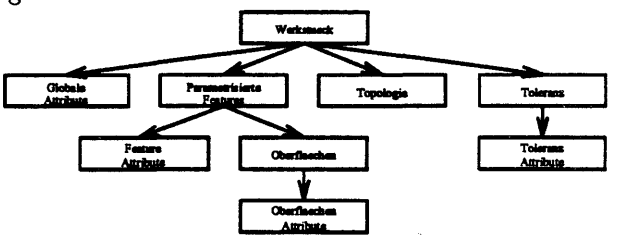


Abbildung A.7: Gesamtarchitektur der GPPS-Umgebung

Teilebeschreibung
Basis des GPPS bildet das sogenannte „engineering model" des Werkstücks. Dieses umfaßt:



Aufbau des „engineering models"

- alle Features
- die Toleranzbeziehungen zwischen Features
- die Oberflächeneigenschaften des Teiles
- allgemeine Information über das Teil

Im wesentlichen ist es also ein geometrisches Modell, dem zusätzliche topologische Informationen mitgegeben werden. Die Definition des Modells erfolgt mit Hilfe einer Matrix der geometrischen Features[7]

[6] Leider kann an dieser Stelle nicht näher auf diese Topologie eingegangen werden, da dies den Rahmen eines Überblicks sprengen würde. Für nähere Information sei auf [PS89] verwiesen.

[7] In GPSS wird der Begriff des Features in einem etwas weiteren Sinn als in den meisten anderen Systemen verwandt.

(MGF). Es wird das gewünschte Feature aus dieser Matrix ausgewählt und durch Wahl der Parameter instanziiert. Diese Art der Definition erlaubt es einerseits schnell Teile zu beschreiben, andererseits erleichtert die implizite Klassifikation der Features durch die Matrix die Klassifikation des Gesamtteiles.

Von besonderem Interesse für GPSS ist das topologische Modell des Werkstücks, da es die Grundlage für die Verfahrensplanung bildet. Die Topologie des Teiles wird hier mit zwei Relationen beschrieben:

- left-of(x,y)
- carries(x,y)

die die Beziehungen zwischen den Features des Teils darstellen.

Werkstattbeschreibung

Die Beschreibung erfolgt mit Hilfe von 4 Datenbasen:

- einer Verfahrensdatei
  Die Verfahrensdatei enthält die zulässigen Fertigungsverfahren eines geometrischen Features mit ihren kinematischen Charakteristika und Anforderungen an die benötigten Werkzeuge.
- einer Maschinendatei
  In der Maschinendatei sind die einzelnen Maschinen und ihre kinematischen Charakteristika abgespeichert. Damit können die Maschinen, die ein bestimmtes Fertigungsverfahren erlauben leicht bestimmt werden.
- einer Material- und Schnittbedingungsdatei
  Mit der Material- und Schnittbedingungsdatei werden anhand des Verfahrens und des Materials die Schnittbedingungen festgelegt.
- einer Werkzeugdatei
  Die Werkzeugdatei enthält alle verfügbaren Werkzeuge. Aus diesen erfolgt die Auswahl unter Berücksichtigung der verwendeten Maschinen, der bestimmten Schnittbedingungen, des verwendeten Fertigungsverfahrens und den durch das zu fertigende Feature vorgegebenen geometrischen Randbedingungen.

Repräsentation des Planungswissens

Es wird in GPSS zwischen zwei Planungsebenen unterschieden:

- Verfahrensplanung
- Operationsplanung

Bei der Verfahrensplanung werden die Reihenfolge der Featurebearbeitung und die anzuwendenden Fertigungsverfahren bestimmt, während bei der Operationsplanung die zur Bearbeitung eines Features notwendigen Operationen festgelegt werden.

Über die Repräsentation des Wissens zur Verfahrensplanung wird leider keine Aussage gemacht, das Wissen zur Operationsplanung wird durch Produktionsregeln dargestellt. Es werden 4 Arten von Regeln unterschieden:

- Regeln zur Ermittlung der technologischen Charakteristiken der zu fertigenden Features
- Regeln zur Bestimmung der Operationsreihenfolge
- Regeln zur Bestimmung von Eigenschaften von Operationen
- Regeln zur Veränderung des Werkstückzustandes

Um auf einfache Art und Weise Toleranzen und Unschärfe beim Planungsprozess darstellen zu können, stellt GPSS Intervalle mit darauf definierten logischen und arithmetischen Operationen zur Verfügung.

Vorgehensweise

In GPSS wird vom Fertigteil zum Rohteil geplant. In einem ersten Schritt erfolgt die Verfahrensplanung, an die sich die Operationsplanung anschließt. Im einzelnen werden die folgenden Schritte durchlaufen:

So werden z. B. die Hauptformelemente als 'externe' Features dargestellt, und Strukturen als ein Feature beschrieben, die anderenorts als eine Menge von Features angesehen werden.

1. Verfahrensplanung

   Die Verfahrensplanung bestimmt die Reihenfolge und Art der Bearbeitung der Features. Dazu genügt die Verwendung eines eingeschränkten Modells des Werkstücks, in dem im wesentlichen die Topologieinformation und einige grundlegende geometrische Daten enthalten sind. Dieses Modell wird durch einen Graphen realisiert, dessen Kanten mit den oben eingeführten Topologierelationen beschriftet sind, und dessen Knoten die noch zu bearbeitenden Features des Werkstücks darstellen.

   Die Planung geschieht in vier Phasen:

   (a) Auswahl des nächsten zu fertigenden Features

   (b) Bestimmung der für das Feature zulässigen Fertigungsverfahren

   (c) Auswahl aus den zulässigen Verfahren unter Berücksichtigung von Technologischen- und Kostenkriterien

   (d) Update des Topologiegraphen
   Die Anwendung eines Verfahrens auf ein Feature kann unterschiedliche Auswirkungen auf den Topologiegraphen haben:

      i. Änderung der Abmessungen eines Features ohne Änderung des Graphen

      ii. Transformation eines Features in ein anderes

      iii. Elimination eines Features

   Abschließend wird versucht die Verfahrungsfolge zu optimieren und Tasks, das sind Gruppen von Bearbeitungen auf der gleichen Maschine, zu bilden. Die endgültige Verfahrensfolge wird durch die Einhaltung von Reihenfolgeregeln und Beachtung genereller Prinzipien ermittelt.

   So erfolgt prinzipiell die Bearbeitung der rotationssymmetrischen Features zuerst. Daran schließt sich die Bearbeitung der nichtrotationssymmetrischen Features an, den Schluß bilden spezielle Oberflächenbehandlungen.

2. Operationsplanung

   Die Operationsplanung benötigt im Gegensatz zur Verfahrensplanung sämtliche technologische Informationen über das Werkstück. Daher baut sie ein technologisches Modell des Teiles (TMP) auf, das als dynamische Datenstruktur benutzt wird. Es wird aus den Daten des „engineering models“ gewonnen.

   Das TMP ist als Quadrupel workpiece(wd, eF, iFL, iFR) definiert. In diesem sind außer den globalen Werkstückdaten (wd) eine Liste der externen Features (eF) und je eine Liste der internen Features von links (iFL) und rechts (iFR) enthalten.

   Die Features selbst enthalten dann die Information über Toleranzen, Oberflächen und Dimensionen.

   Die Operationsplanung selbst erfolgt mit Hilfe eines regelbasierten Systems, dessen Grundstruktur sieht wie folgt aussieht:

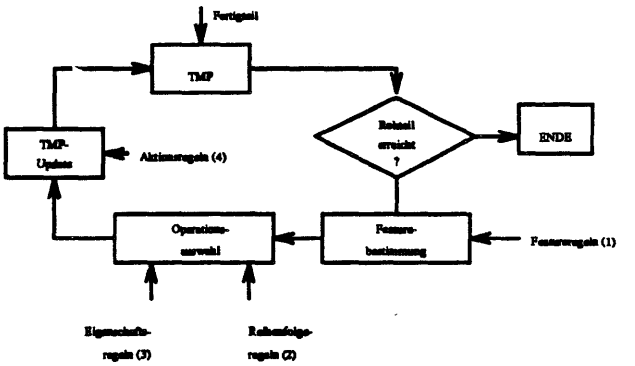
Solange das Rohteil nicht erreicht ist, führe folgende Schritte aus:

(a) Bestimme die charakteristische Oberfläche des ausgewählten Features mit Hilfe der Regeln des Typs 1.

(b) Teste die Operationen in der durch die Regeln vom Typ 2 festgelegten Reihenfolge auf ihre Zulässigkeit. Dies geschieht mit den Regeln vom Typ 3.

(c) Für jede zulässige Operation bestimme des neuen Zustand des TMP gemäß der Regeln des Typs 4.

(d) Bestimme die Operationsreihenfolge für das veränderte Feature



Der Operationsplanungszyklus

Verwendete KI-Methoden

Realisation der Operationsplanung mit Hilfe von Produktion regeln und Forward-Chaining.

Systemarchitektur

siehe Bild A.8 auf Seite 112

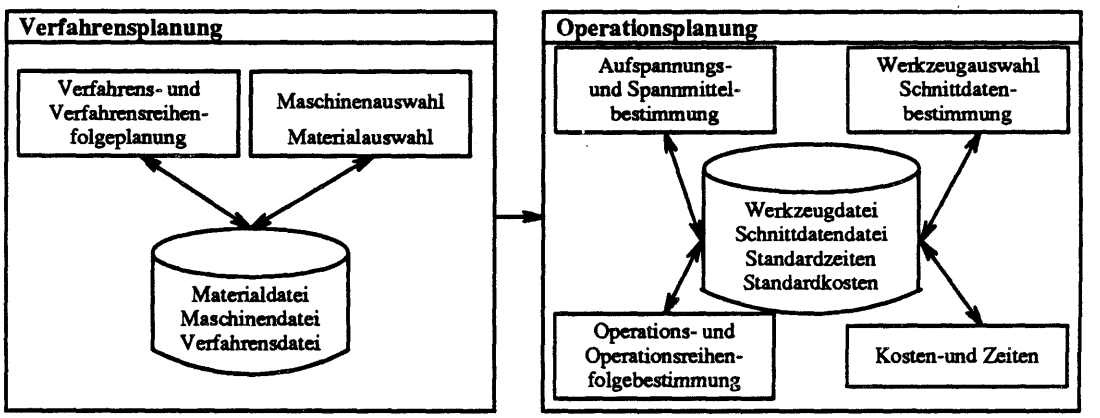


Abbildung A.8: Systemarchitektur GPSS

Weitere Vorhaben

Erweiterung der Topologie der Fertigungsverfahren auf die nichtspanende Bearbeitung und Anpassung des Produktspektrums.

Implementierung

In PROLOG

Umfang

???

Literatur

[PS89]

A.11 KAPLAN

Anwendungsgebiet

Vollautomatische Generation von Arbeitsplänen für Drehteile

Funktionsumfang

Teil eines komplexeren Planungssystems für eine flexible Fertigungszelle, die aus einer Drehbank einer Universaldrehmaschine und einem Roboter besteht (siehe Bild A.9).

Weitere Komponenten dieses Systems sind:

1. Erweiterung eines CAD-Systems in Richtung produktionsorientierter Konstruktion
2. Werkzeugauswahl bei der Drehbearbeitung
3. Arbeitsplangenerierung für ein Bearbeitungszentrum (in Entwicklung)
4. Off-line Programmierung und Simulation von Roboterbewegungen
5. Graphische Simulation der Bearbeitung
6. Bildverarbeitung (Erkennen der Rohteile)
7. Kontrolle und Monitoring
8. Planung der Werkzeugmontage und Montage der Werkzeuge

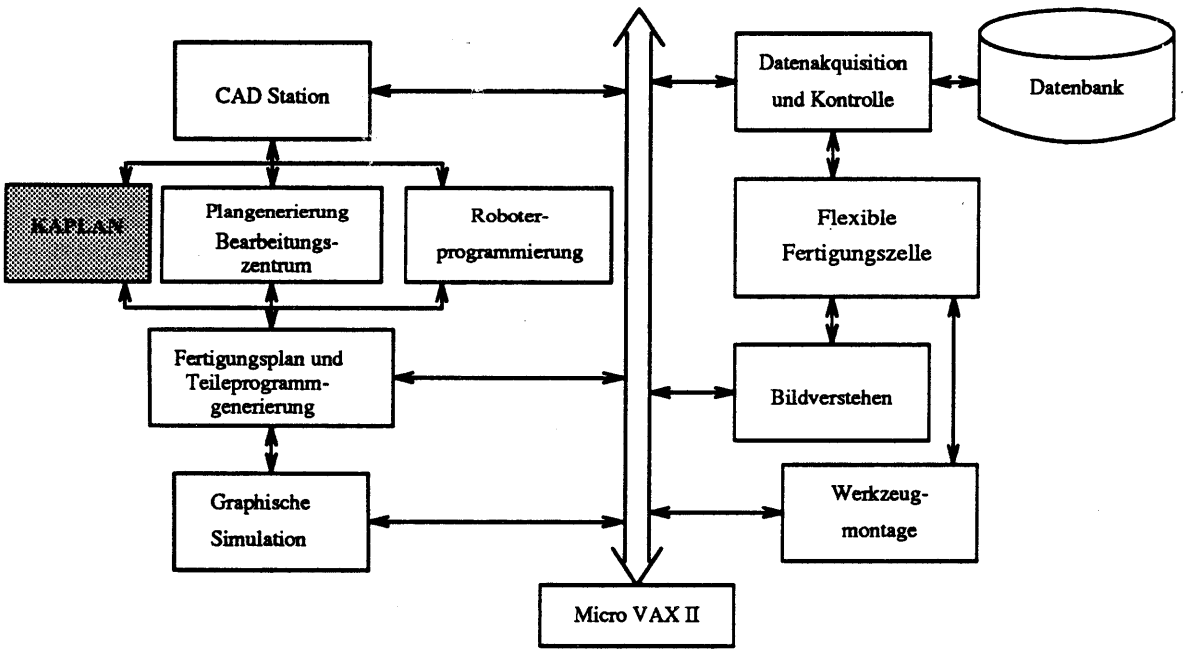


Abbildung A.9: Gesamtarchitektur der KAPLAN-Umgebung

KAPLAN arbeitet innerhalb dieser Umgebung mit den ersten beiden Modulen zusammen.

Teilebeschreibung

Definition des Teiles innerhalb des obengenannten CAD-Moduls. Dieses ist eine Erweiterung eines kommerziellen CAD-Systems in Richtung produktionsorientierter Konstruktion.

Das Teil wird als eine Folge von elementaren Oberflächen, wie z. B. Zylinder, Konus, Nut etc. beschrieben. Nach der Definition der Geometrie werden die technologischen Daten der Oberfläche erfragt.

Nicht rotationssymmetrische Elemente des Teiles werden von KAPLAN automatisch mit einer zylindrischen Hülle versehen, so daß das Teil soweit wie möglich auf einer Drehbank gefertigt werden kann.

Werkstattbeschreibung

???

Repräsentation des Planungswissens

4 Arten von Produktionsregeln:

1. Generelle Regeln
   Sie enthalten Tests ob das Teil prinzipiell gefertigt werden kann.
2. Einschränkungsregeln
   Sie dienen der Einschränkung des Suchraums und der Anpassung an die momentane Werkstattsituation.
3. Aufspannungsregeln
   Diese zerfallen wiederum in zwei Untertypen:
   (a) Regeln zur Wahl der Aufspannfläche
   (b) Regeln zur Wahl des Spannmittels
4. Zulässigkeitsregeln
   Mit ihnen wird geprüft, ob eine Fläche in einer bestimmten Aufspannung fertigbar ist.

Die Regeln enthalten zusätzlich zu Bedingung und Aktion noch ein Gewicht, das bei der Selektion des besten Planes benutzt wird.

Vorgehensweise

Im Gegensatz zu anderen Systemen werden alle möglichen Pläne generiert und der beste ausgewählt. Dazu werden für alle möglichen Aufspannungen eines Teiles die möglichen Operationsfogen ermittelt. Dieses Verfahren wird solange auf die dadurch entstandenen Teile angewandt, bis das Fertigteil erreicht ist.

Anschliessend werden die so erzeugten Pläne mit Hilfe einer Funktion bewertet und der beste ausgewählt.

Im einzelnen wird wie folgt vorgegangen:

1. Auswahl der Maschine (manuell)
2. Abarbeitung der generellen Regeln
3. Abarbeitung der einschränkenden Regeln
   Berücksichtigung der momentanen Werkstattsituation.
4. Analyse der Spannflächen und Auswahl des Spannmittels
   Ermittlung der zum Spannen geeigneten Flächen und Auswahl eines verwendbaren Spannmittels.
5. Analyse der Fertigbarkeit der Oberflächen in Bezug auf die gewählte Aufspannung
   Bestimmung der fertigbaren Flächen und Bewertung der Aufspannung mit Hilfe einer Zahl, des sogenannten Scores. Dieser ist eine Funktion der folgenden Parameter:
   - Position der Aufspannung
   - Geometrie des Spanmittels
   - Geometrie der zu fertigenden Oberflächen
   - Positionstoleranzen
6. Berechnung der neuen Teilegeometrie
7. Wiederholung der Schritte 4 – 6, bis das Fertigteil erreicht ist.
8. Auswahl des besten Planes
   Auswahl unter den Plänen mit den höchsten Scores. Eine endgültige Bestimmung des Planes ist aber erst möglich, wenn durch die nachfolgenden Module die Zeiten und Kosten ermittelt wurden.

Verwendete KI-Methoden

Produktionsregeln

Systemarchitektur

siehe Bild A.10 auf Seite 115

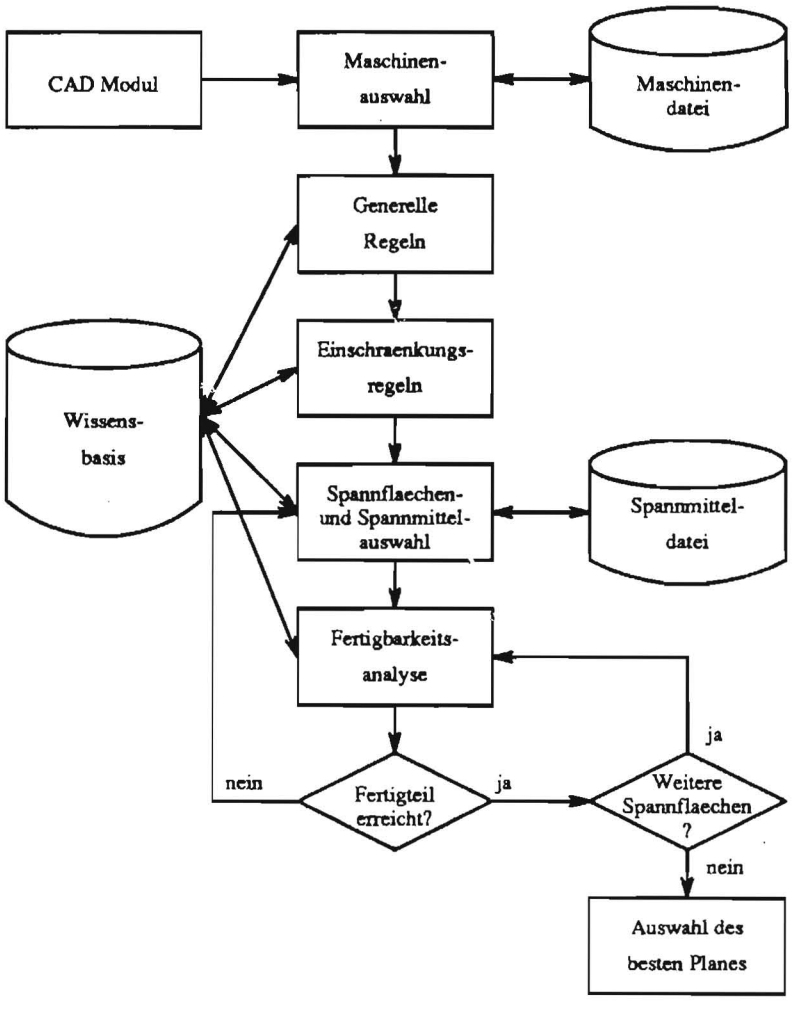


Abbildung A.10: Systemarchitektur KAPLAN

Weitere Vorhaben
Vervollständigung des Gesamtsystems

Implementierung
???

Umfang
???

Literatur
[GSD89]

A.12 ROUND

Anwendungsgebiet
Planung von Drehoperationen

Funktionsumfang
Komplettes System mit Planung und Informationsverarbeitung von der Teiledefinition bis zur NC-Bearbeitung.

Teilebeschreibung
Interaktive Beschreibung des Teiles im Modul RNDINP. Falls keine Rohteilbeschreibung erfolgt, wird automatisch ein passendes Halbzeug (Stange) ausgewählt (im Modul RNDFIX).

Werkstattbeschreibung
Maschinen, Materialien, Werkzeuge etc. sind in externen Dateien gespeichert.

Repräsentation des Planungswissens
???

Vorgehensweise
Das System macht die Grundannahme, das es nicht immer möglich ist, die optimale Lösung auf Anhieb zu finden, und eine Untersuchung aller Lösungen oder intensives Backtracking zu lange dauern würde.
Daher ist es notwendig, möglichst früh die Auswahl zu fokussieren, um dann durch Verfeinerung dieser Auswahl zu einer Lösung zu kommen. Aus diesem Grunde wurde in ROUND besonders viel Wert auf eine gute Wahl der Schätzfunktionen gelegt.
Die Vorgehensweise des Systems ist wie folgt:

1. Definition von Roh- und Fertigteil
2. Auswahl der Maschinen (manuell)
3. Wahl der Aufspannungen mit Wahl der Spannmittel
   Dazu werden die möglichen Spannflächen ermittelt und die passenden Spannbacken ausgewählt. Die Bewertung der einzelnen Möglichkeiten erfolgt mit einer sehr komplexen Schätzfunktion, die hier nicht näher erläutert werden soll.
4. Unterteilung des zu zerspanenden Volumens
   Die Unterteilung erfolgt momentan noch manuell, ein Modul zur Automatisierung ist in Entwicklung.
5. Auswahl der Schrupp- und Schlichtwerkzeuge
   Die Wahl des optimalen Werkzugs für jeden Bereich würde zu einer zu großen Menge von Werkzeugen führen. Daher muß nach einer optimalen Kombination von Werkzeugen gesucht werden. In ROUND wird diese Aufgabe in zwei Schritten gelöst:
   - Auswahl aller möglichen Werkzeuge für jeden Bereich
     Kann ein Bereich nicht mit einem Werkzeug gefertigt werden (z. B. Einschnitte), wird dieser rekursiv unterteilt.
     Um festzustellen, ob überhaupt ein geeignetes Werkzeug vorhanden ist, wird zunächst in einem kleinem Satz von Grundwerkzeugen gesucht, nur wenn dort keines gefunden wird, erfolgt eine Suche unter allen vorhandenen Werkzeugen.
     Die verwendbaren Werkzeuge werden wie folgt ermittelt:
     (a) Bestimmung der Art der Bearbeitungsoperation
     (b) Vergleich der Toolvektoren der Werkzeuge mit der Bereichsgeometrie
         Die Toolvektoren geben dabei den Winkelbereich an, den das Werkzeug abdeckt(siehe Bild A.11).
     (c) Kollisionsüberprüfung zwischen Werkzeughalter und Teil
   - Auswahl der optimalen Werkzeugmenge
     Die Auswahl ist realisiert als Lösung eines kürzesten Pfadproblems in einem hierarchischen Netzwerk. Dabei entsprechen die Knoten des Netzes den Bereichen, die Kanten den Kosten der Bearbeitung des jeweiligen Bereiches mit einem bestimmten Werkzeug. Die Bereiche sind

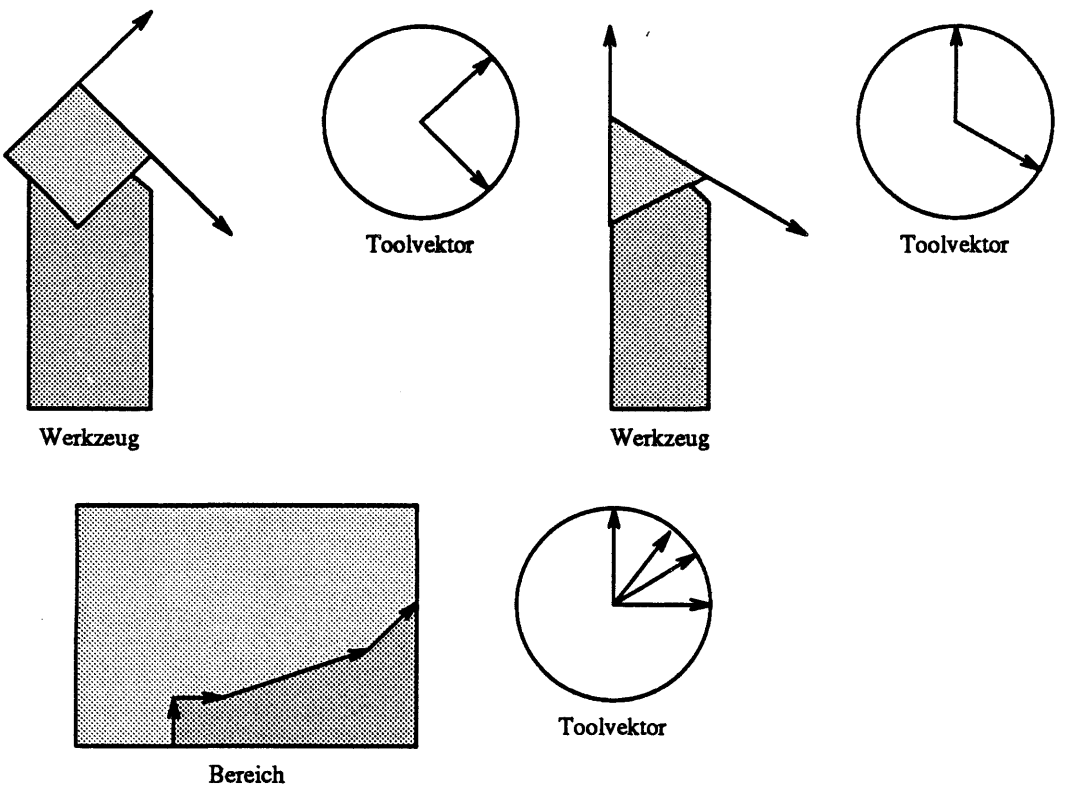


Abbildung A.11: Aufbau der Toolvektoren

nach ihrem Volumen aufsteigend sortiert, die Kantengewichte ergeben sich als eine Summe der Kosten für das Werkzeugmanagement, den Werkzeugwechsel, die Bearbeitung und des Werkzeuges.

Verwendete KI-Methoden
heuristische Suche

Systemarchitektur
siehe Bild A.12 auf Seite 118

Weitere Vorhaben
???

Implementierung
???

Umfang
???

Literatur
[vH86, vHK84]

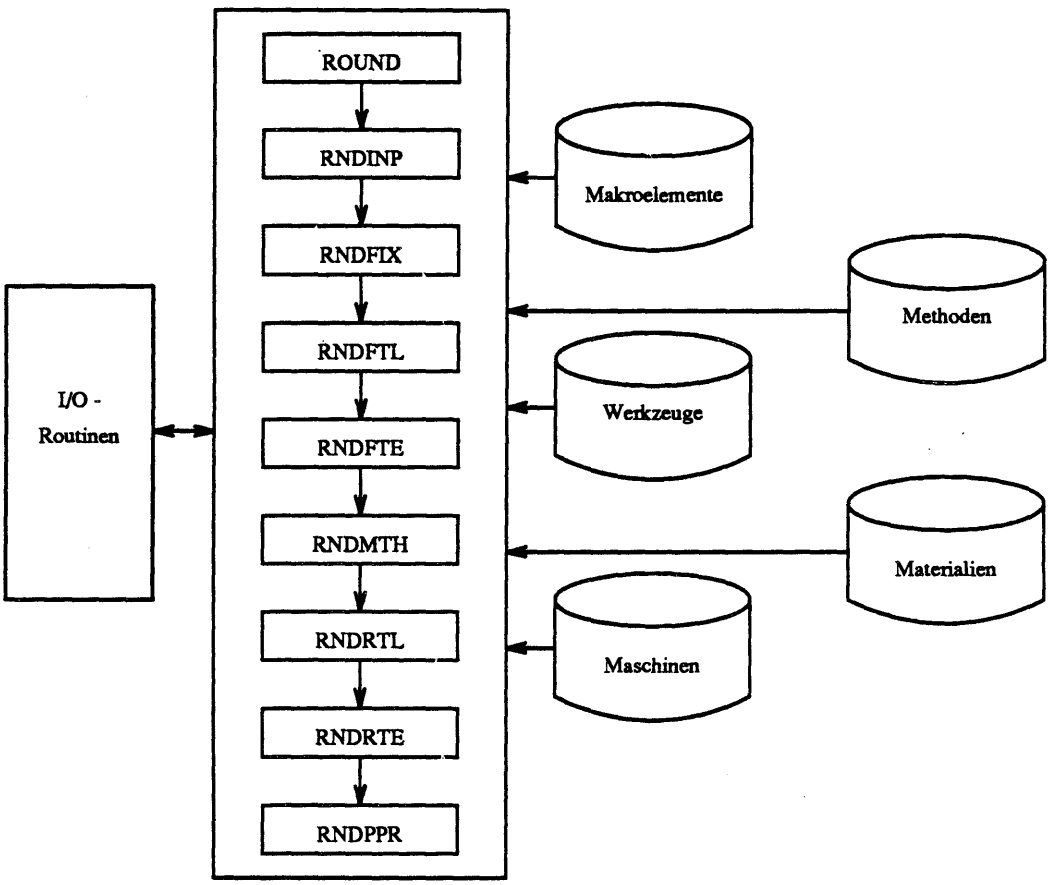


Abbildung A.12: Systemarchitektur ROUND

A.13 XPLANE

Anwendungsgebiet

Generative Arbeitsplanung für prismatische Teile

Funktionsumfang

- Feature-Extraktion aus einem Volumenmodell
- Auswahl von Maschinenoperationen
- Auswahl von Werkzeugen
- Reihenfolgebestimmung der Operationen
- Eingebettet in ein Gesamtsystem (siehe Bild A.13)

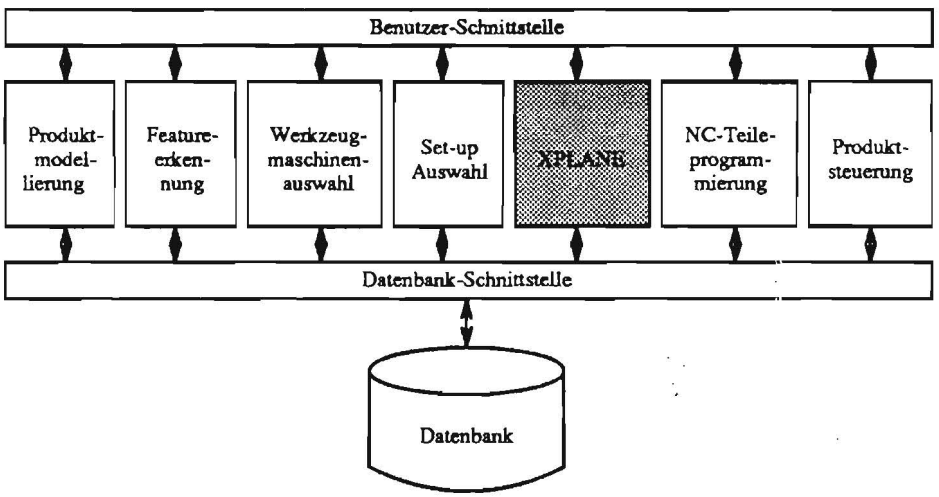


Abbildung A.13: Gesamtarchitektur der XPLANE-Umgebung

Teilebeschreibung

Ein Volumenmodell des Werkstücks mit technologischer Zusatzinformation wird mit Hilfe des geometrischen Modellieres G.P.M erzeugt.

Aus diesem Volumenmodell werden mit Hilfe eines Feature-Erkennungsmoduls die relevanten fertigungstechnischen Elemente erkannt.

Werkstattbeschreibung

Vorhandene Maschinen, Spannmittel, Werkzeuge und Materialien sind in einer Einbenutzerdatenbank (RIOT) abgelegt.

Repräsentation des Planungswissens

Produktionsregeln

Vorgehensweise

Es wird ein mehrstufiges Verfahren mit den folgenden Schritten angewandt:

1. Featureerkennung
   Die automatische Erkennung von Bohrlöchern, Schlitzen und Taschen ist möglich. Zur Erkennung weiterer Featuretypen und Korrektur von Fehlerkennungen wird ein interaktiver Editor verwandt.
2. Auswahl der Maschinenoperationen für jedes Feature eine Aufspannung
   Die Auswahl erfolgt unter Verwendung von heuristischer Graphensuche (A*-Algorithmus). Dabei stellen die Knoten des Graphen die erreichten (Zwischen-)Zustände des Werkstücks dar, während die Kanten die Ausführung der jeweiligen Maschinenoperation repräsentieren.
   Das Wissen über die anwendbaren Operationen etc. ist in Form von Produktionsregeln abgelegt. Gleichzeitig mit der Operationsauswahl werden die für die einzelnen Operationen zulässigen Werkzeuge bestimmt.

3. Minimierung der Anzahl der Werkzeuge in einer Aufspannung
   Aus den zulässigen Werkzeugen aller Features einer Aufspannung erfolgt die Auswahl der verwendeten Werkzeuge mit folgendem Algorithmus:
   (a) Erstellung einer Liste, die für alle Operationen die zulässigen Werkzeuge aufführt (LZW).
   (b) Bestimmung der Anzahl der Vorkommen jedes Werkzeugs in der Liste LZW
   (c) Entfernung aller Operationen die mit dem am häufigsten vorkommenden Werkzeug ausgeführt werden können aus der Liste LZW.
   Bei Gleichstand erfolgt eine Zufallswahl des Werkzeugs.
   (d) Ist die Liste LZW nicht leer, gehe nach b).
4. Festlegung der Operationsreihenfolge (geplant)
   Die Festlegung der Operationsreihenfolge erfolgt ebenfalls mit heuristischer Graphensuche, mit dem Ziel die Standzeit zu minimieren.
   Dabei finden Constraints, die Reihenfolge der Operationen innerhalb eines Features und die Reihenfolge der Featurebearbeitung betreffend, Beachtung.

Verwendete KI-Methoden

- heuristische Graphensuche
- Produktionsregeln

Systemarchitektur

siehe Bild A.14 auf Seite 120

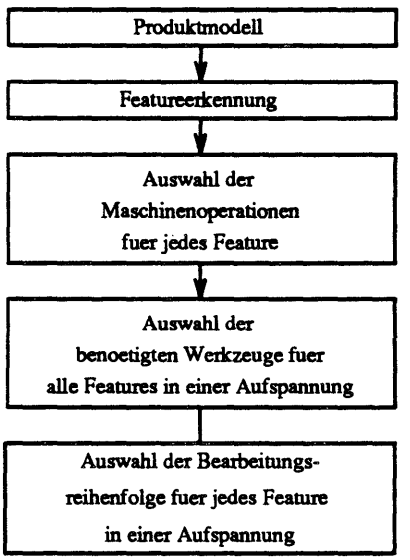


Abbildung A.14: Systemarchitektur XPLANE

Weitere Vorhaben

- Erweiterung des Spektrums der erkennbaren Features
- Realisierung der Featureerkennung als wissensbasiertes System

Implementierung

In FORTRAN-77 auf einer VAX aus zwei Gründen:

- Der Modellierer G.P.M. ist in FORTRAN implementiert
- Zu Beginn des Projektes standen keine geeigneten KI-Werkzeuge zur Verfügung. Daher werden in XPLANE die Produktionsregeln in einer selbst definierten Sprache eingegeben und in FORTRAN-Code compiliert.

Umfang
???

Literatur
[vEK86, vE88, JF89]

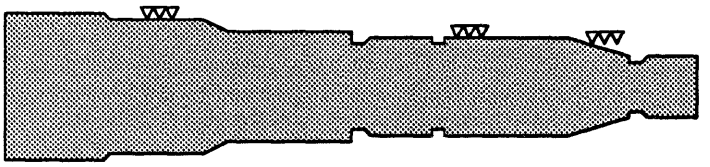
In der nachfolgenden Tabelle sind Systeme aufgeführt, die in dieser Arbeit nicht ausführlich behandelt wurden, da sie entweder schon an anderer Stelle behandelt wurden, oder das zur Verfügung stehende Informationsmaterial nicht ausführlich genug war. Sie gingen aber trotzdem in die Bewertung und den Vergleich des vorherigen Abschnitts ein.

System	Kurzbeschreibung	Literatur
EXPLAN	Arbeitsplanung prismatischer Teile Feature erkennung durch Interpretation von CAD-Daten Bereitstellung eines Maschinen- und Planungsmodells	[HPR90, HM]
FBPPS	Arbeitsplanung prismatischer Teile Erzeugung der Werkstückbeschreibung aus einem „intelligent CAD-System“ Hierarchisches Planen auf mehreren Abstraktionsebenen Darstellung des Planungswissens durch Regeln und Frames Blackboard-System in PROLOG implementiert	[JF89]
Hi-Mapp	Teilefertigung, Zerspanung prismatischer Teile Kopplung mit CAD-System möglich Hierarchisches, nichtlineares Planen unter Verwendung eines bereichsunabhängigen Planers. Verwendung von Produktionsregeln und backward-chaining	[JF89]
IXPRESS	Blechteilefertigung mittelgroßer Tiefziehteile Reihenfolgeplanung, Werkzeugbestimmung, Alternativplanung Eingabe der Werkstückbeschreibung im Dialog Implementierung in CommonLisp und C	[JF89]
XPS-E	Arbeitsplanungsshell für die Bereiche Zerspanung, Montage, Formerei und Schmieden Reihenfolgeplanung, Betriebsmittelzuordnung, Alternativplanung Featureorientierte Beschreibung des Werkstücks Teilplangenerierung für die einzelnen Features, dann Kombination der Teilpläne	[JF89]

Tabelle A.2: Weitere Systeme zur wissensbasierten generativen Arbeitsplanung

System	Kurzbeschreibung	Literatur
LOCAM AP	Für alle Bereiche der Fertigungsplanung verwendbar Kopplung mit Entscheidungstabellen Automatische Übernahme von Daten aus CAD und PPS-Systemen Alternativplanung	[Log89, Ltd90]
Kronos	Arbeitsplanung von Drehteilen Interaktive Definition des Werkstücks in einem Grafikeditor Manuelle Maschinen- und Werkzeugvorauswahl Automatische Bestimmung und Bewertung aller möglichen Aufspannungen, Spannmittel und Werkzeuge Interaktive Festlegung der Aufspannungen, Spannmittel und Werkzeuge Generierung des NC-Codes durch einen Postprozessor Backward-Reasoning mit Produktionsregeln Implementiert in C und PASCAL auf Apollo	[N.N90]
Autolathe	Arbeitsplanung von Drehteilen Kopplung mit dem CAD-System DOGS-2D zur Übernahme von Geometriedaten Erzeugung des CNC-Codes Implementierung in C auf Apollo	[Fri90]

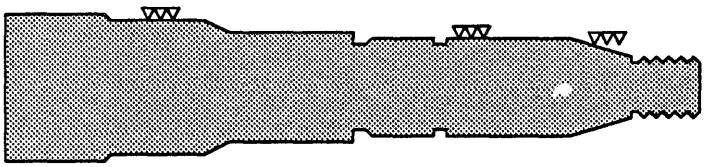
Tabelle A.3: Weitere Systeme zur wissensbasierten generativen Arbeitsplanung (Fortsetzung)



B | Verzeichnisse

Abbildungsverzeichnis

Tabellenverzeichnis



C | Literaturverzeichnis

[All83] J.F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843, 1983.

[ASP90] Nico Anders, Martin Schaele, und Karl-Wilhelm Prack. AVOGEN – wissensbasierte Generierung von Arbeitsvorgangsfolgen. *VDI-Z*, 132(4):49–52, April 1990.

[AWF69] AWF/REFA. *Arbeitsplanung*, Band 1 des *Handbuch der Arbeitsvorbereitung*. Beuth-Verlag, Berlin, Köln, Frankfurt, 1969.

[BKL90] A. Bernardi, C. Klauck, und R. Legleitner. Abschlussbericht des Arbeitspaketes PROD. Technical report, DFKI, Kaiserslautern, 1990.

[BKL90a] A. Bernardi, C. Klauck, und R. Legleitner. Formalismus zur Repräsentation von Geometrie- und Technologieinformationen als Teil eines Wissensbasierten Produktmodells. Technical report.

[Ble90] L. Blencke. STEP-IPIM als Integrationsmodell (Bewertung aus der Sicht der Datenmodellierung, Anforderungen an die Datenverwaltung). In Theo Härder und Christoph Hübel, Hrsg., *Daten- und Wissensverwaltung für integrierte Ingenieursysteme – Anforderungen und Realisierungskonzepte*, Nummer 1. ZRI - Zentrum Rechnergestützte Ingenieursysteme, 1990. TechReport.

[CGS91] R. Cunis, A. Günter, und H. Strecker, Hrsg.. *Das Plakon-Buch*, Band 266 der *Informatik-Fachberichte*. Springer, Berlin, Heidelberg, New York, 1991.

[DL81] Y. Descotte und J. C. Latombe. GARI: A problem solver that plans how to machine mechanical parts. In *Proc. of the 7 th IJCAI*, 766–772, Vancouver, Canada, 1981.

[EZBB87] O. Eliyahu, L. Zaidenberg, und M. Ben-Bassat. CAMEX: An expert system for process planning on CNC machines. In *Proc. of AAAI-87*, 794–798, Seattle, WA, 1987.

[FHG90] H. Feller, H.-J. Held, und G.Jüttner. Mit KI–Methoden Arbeitsvorgangsfolgen planen und optimieren. *Arbeitsvorbereitung*, 27(2):59–61, 1990.

[Fri90] J. Fricke. Autolathe, 1990.

[Gü87] H.-W. Güsgen. *CONSAT – A System for Constraint Satisfaction*. Dissertation, GMD, St. Augustin, 1987.

[gen90] GENOA - Generieren und Optimieren von Arbeitsplänen. Kurzbeschreibung, FAW Ulm, 1990.

[GSD89] F. Giusti, M. Santichi, und G. Dini. KAPLAN: a knowledge-based approach to process planning of rotational parts. *Annals of the CIRP*, 38(1):481–484, 1989.

[Gü89] H. W. Güsgen. Spatial reasoning based on allen's temporal logic. Technical Report TR-89-049, International Computer Science Institute, Berkeley, CA, 1989.

[Hä89] T. Härder. Die Rolle von Datenbanken in CIM. *CIM Management*, (6):4–10, 1989.

[HM] C. Mayer H. Muthsam. An expert system for process planning of prismatic workpieces. 211–220.

[HPR90] H. Muthsam H.-P. Roth, K.-P. Zeh. Einsatzmöglichkeiten rechnerunterstützter, wissensbasierter Planungsinstrumentarien. In *1st International Conference on Artificial Intelligence and expert systems in manufacturing*, 211–228, 3 1990.

[HPS89] C. Hübel, R. Paul, und B. Sutter. Technische Modellierung und DB-gestützte Datenhaltung – ein Ansatz für ein durchgängiges, integriertes Produktmodell. Technical Report 6, ZRI - Zentrum Rechnergestützte Ingenieursysteme, 1989.

[HPS90] C. Hübel, R. Paul, und B. Sutter. Datenbankgestützte technische Modellierung – ein Ansatz für die CAD/CAP-Integration. *CIM Management*, (2):48–54, 1990.

[Hum91] B. Humm. Ein System zur fallbasierten Arbeitsplanerstellung. Diplomarbeit, Universität Kaiserslautern, Kaiserslautern, 1991.

[IS89] N. Iudica und S.Ansaldi. Generatives Arbeitsplanungssystem mit CAD–Kopplung. *ZwF*, 84(11):630–634, 1989.

[JF89] G. Jüttner und H. Feller. *Entscheidungstabellen und wissensbasierte Systeme.* R. Oldenbourg Verlag, Müchen, Wien, 1989.

[Kra88] N. Kratz. Ein Ansatz zur Repräsentation von technischen und funktionalen Beziehungen bei der Konstruktion. In *2. Workshop Planen und Konfigurieren*, GMD Arbeitsbericht Nr. 310, 19–29. GMD, 1988.

[LLL89] K. I. Lee, J. W. Lee, und J. M. Lee. Pattern recognition and process planning prismatic workpieces by knowledge based approach. *Annals of the CIRP*, 38(1):485–488, 1989.

[Log89] F.A. Logan. Automatisierungsstufen der rechnerunterstützten Arbeitsplanung. *Zeitschrift für wirtschaftliche Fertigung*, 84(2):93–96, 1989.

[Ltd90] Pafec Ltd. LOCAM AP, 1990.

[Mie88] T. Mielke. CHAMP: Ein Expertensystem zur wissensbasierten Erstellung von Arbeitsplänen. In *WIMPEL' 88, 1.Konferenz über wissensbasierte Methoden für Produktion, Engineering und Logistik*, 86–97, 1988.

[N.N90] N.N. Kronos, der Weg zur automatischen Programmgenerierung, 1990.

[Nö90] K. Nökel. *Temporally Distributed Systems in Technical Diagnosis.* Dissertation, Universität Kaiserslautern, Kaiserslautern, 1990.

[PS89] J. Peklenik und A. Sluga. Contribution to development of a generative CAPP-system based on manufacturing process topology. *Annals of the CIRP*, 38:407–412, 1989.

[PST88] Jörg Pfeiffer, Thomas Siepmann, und Wolfgang Teichmann. Expertensystem zur spanenden Bearbeitung. *Technische Mitteilungen Krupp*, (2):113–124, Feb 1988.

[Pup88] F. Puppe. *Einführung in Expertensysteme.* Studienreihe Informatik. Springer, Berlin, Heidelberg, New York, 1988.

[REF74] REFA. *Planung*, Band 2 de *Methodenlehre der Planung und Steuerung.* Carl Hanser Verlag, München, 1974.

[Ric89] M. M. Richter. *Prinzipien der Künstlichen Intelligenz.* B. G. Teubner, Stuttgart, 1989.

[Row] L. A. Rowe. A shared object hierarchy. Technical report, University of California, Berkeley, ???

[SK83] G. Spur und M. Krause. *CAD-Technik.* 1983.

[Sp72] G. Spur. Automatisierung in der Arbeitsvorbereitung. *Industrial Engineering*, 2(1):59–68, 1972.

[Sp80] G. Spur. Rechnerunterstützte Zeichnungserstellung und Arbeitsplanung. *Zeitschrift für wirtschaftliche Fertigung*, 1980.

[SR86] M. Stonebraker und L. A. Rowe. The design of postgres. In *Proc. 1986 ACM-SIGMOD Conference on Management od Data*, 5 1986.

[SRH] M. Stonebraker, L. A. Rowe, und M. Hirohama. The implementation of postgres. Technical report, University of California, Berkeley, ???

[Sto] M. Stonebraker. Future trends in data base systems. Technical report, University of California, Berkeley, ???

[VDI74] VDI. *Elektronische Datenverarbeitung bei der Produktionsplanung und -steuerung. Automatische Arbeitsplanerstellung*, Band 61 der *VDI-Taschenbuch.* VDI-Verlag, Düsseldorf, 1974.

[vE88] A. H. van't Erve. *Generative computer aided process planning for part manufacturing - an expert system approach.* Dissertation, Universiteit Twente, 1988.

[vEK86] A. H. van't Erve und H. J. J. Kals. XPLANE, a generative computer aided process planning system for part manufacturing. *Annals of the CIRP*, 35(1):325–329, 1986.

[vH86] F.J.A.M. van Houten. Strategy in generative planning of turning processes. *Annals of the CIRP*, 35(1):331–335, 1986.

[vHK84] F.J.A.M. van Houten und H.J.J. Kals. ROUND - a flexible technology based process and operations planning system for NC-lathes. In *Proceedings of the 16th CIRP International Seminar on Manufacturing Systems*, 1984.

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

DFKI
-Bibliothek-
PF 2080
6750 Kaiserslautern
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen oder die aktuelle Liste von erhältlichen Publikationen können bezogen werden von der oben angegebenen Adresse.
Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of currently available publications can be ordered from the above address.
The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

RR-90-01
Franz Baader: Terminological Cycles in KL-ONE-based Knowledge Representation Languages
33 pages

RR-90-02
Hans-Jürgen Bürckert: A Resolution Principle for Clauses with Constraints
25 pages

RR-90-03
Andreas Dengel, Nelson M. Mattos: Integration of Document Representation, Processing and Management
18 pages

RR-90-04
Bernhard Hollunder, Werner Nutt: Subsumption Algorithms for Concept Languages
34 pages

RR-90-05
Franz Baader: A Formal Definition for the Expressive Power of Knowledge Representation Languages
22 pages

RR-90-06
Bernhard Hollunder: Hybrid Inferences in KL-ONE-based Knowledge Representation Systems
21 pages

RR-90-07
Elisabeth André, Thomas Rist: Wissensbasierte Informationspräsentation:
Zwei Beiträge zum Fachgespräch Graphik und KI:
1. Ein planbasierter Ansatz zur Synthese illustrierter Dokumente
2. Wissensbasierte Perspektivenwahl für die automatische Erzeugung von 3D-Objektdarstellungen

24 pages

RR-90-08
Andreas Dengel: A Step Towards Understanding Paper Documents
25 pages

RR-90-09
Susanne Biundo: Plan Generation Using a Method of Deductive Program Synthesis
17 pages

RR-90-10
Franz Baader, Hans-Jürgen Bürckert, Bernhard Hollunder, Werner Nutt, Jörg H. Siekmann: Concept Logics
26 pages

RR-90-11
Elisabeth André, Thomas Rist: Towards a Plan-Based Synthesis of Illustrated Documents
14 pages

RR-90-12
Harold Boley: Declarative Operations on Nets
43 pages

RR-90-13
Franz Baader: Augmenting Concept Languages by Transitive Closure of Roles: An Alternative to Terminological Cycles
40 pages

RR-90-14
Franz Schmalhofer, Otto Kühn, Gabriele Schmidt: Integrated Knowledge Acquisition from Text, Previously Solved Cases, and Expert Memories
20 pages

RR-90-15
Harald Trost: The Application of Two-level Morphology to Non-concatenative German Morphology
13 pages

RR-90-16
Franz Baader, Werner Nutt: Adding Homomorphisms to Commutative/Monoidal Theories, or: How Algebra Can Help in Equational Unification
25 pages

RR-90-17
Stephan Busemann
Generalisierte Phasenstrukturgrammatiken und ihre Verwendung zur maschinellen Sprachverarbeitung
114 Seiten

RR-91-01
Franz Baader, Hans-Jürgen Bürckert, Bernhard Nebel, Werner Nutt, and Gert Smolka :
On the Expressivity of Feature Logics with Negation, Functional Uncertainty, and Sort Equations
20 pages

RR-91-02
Francesco Donini, Bernhard Hollunder, Maurizio Lenzerini, Alberto Marchetti Spaccamela, Daniele Nardi, Werner Nutt:
The Complexity of Existential Quantification in Concept Languages
22 pages

RR-91-03
B.Hollunder, Franz Baader: Qualifying Number Restrictions in Concept Languages
34 pages

RR-91-04
Harald Trost
X2MORF: A Morphological Component Based on Augmented Two-Level Morphology
19 pages

RR-91-05
Wolfgang Wahlster, Elisabeth André, Winfried Graf, Thomas Rist: Designing Illustrated Texts: How Language Production is Influenced by Graphics Generation.
17 pages

RR-91-06
Elisabeth André, Thomas Rist: Synthesizing Illustrated Documents
A Plan-Based Approach
11 pages

RR-91-07
Günter Neumann, Wolfgang Finkler: A Head-Driven Approach to Incremental and Parallel Generation of Syntactic Structures
13 pages

RR-91-08
Wolfgang Wahlster, Elisabeth André, Som Bandyopadhyay, Winfried Graf, Thomas Rist
WIP: The Coordinated Generation of Multimodal Presentations from a Common Representation
23 pages

RR-91-09
Hans-Jürgen Bürckert, Jürgen Müller, Achim Schupeta
RATMAN and its Relation to Other Multi-Agent Testbeds
31 pages

RR-91-10
Franz Baader, Philipp Hanschke
A Scheme for Integrating Concrete Domains into Concept Languages
31 pages

RR-91-11
Bernhard Nebel
Belief Revision and Default Reasoning: Syntax-Based Approaches
37 pages

RR-91-13
Gert Smolka
Residuation and Guarded Rules for Constraint Logic Programming
17 pages

RR-91-15
Bernhard Nebel, Gert Smolka
Attributive Description Formalisms ... and the Rest of the World
20 pages

RR-91-16
Stephan Busemann
Using Pattern-Action Rules for the Generation of GPSG Structures from Separate Semantic Representations
18 pages

DFKI Technical Memos

TM-89-01
Susan Holbach-Weber: Connectionist Models and Figurative Speech
27 pages

TM-90-01
Som Bandyopadhyay: Towards an Understanding of Coherence in Multimodal Discourse
18 pages

TM-90-02
Jay C. Weber: The Myth of Domain-Independent Persistence
18 pages

TM-90-03
Franz Baader, Bernhard Hollunder: KRIS: Knowledge Representation and Inference System -System Description-
15 pages

TM-90-04
Franz Baader, Hans-Jürgen Bürckert, Jochen Heinsohn, Bernhard Hollunder, Jürgen Müller, Bernhard Nebel, Werner Nutt, Hans-Jürgen Profitlich: Terminological Knowledge Representation: A Proposal for a Terminological Logic
7 pages

TM-91-01
Jana Köhler
Approaches to the Reuse of Plan Schemata in Planning Formalisms
52 pages

TM-91-02
Knut Hinkelmann
Bidirectional Reasoning of Horn Clause Programs: Transformation and Compilation
20 pages

TM-91-03
Otto Kühn, Marc Linster, Gabriele Schmidt
Clamping, COKAM, KADS, and OMOS: The Construction and Operationalization of a KADS Conceptual Model
20 pages

TM-91-04
Harold Boley
A sampler of Relational/Functional Definitions
12 pages

TM-91-05
Jay C. Weber, Andreas Dengel and Rainer Bleisinger
Theoretical Consideration of Goal Recognition Aspects for Understanding Information in Business Letters
10 pages

DFKI Documents

D-89-01
Michael H. Malburg, Rainer Bleisinger: HYPERBIS: ein betriebliches Hypermedia-Informationssystem
43 Seiten

D-90-01
DFKI Wissenschaftlich-Technischer Jahresbericht 1989
45 pages

D-90-02
Georg Seul: Logisches Programmieren mit Feature -Typen
107 Seiten

D-90-03
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner: Abschlußbericht des Arbeitspaketes PROD
36 Seiten

D-90-04
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner: STEP: Überblick über eine zukünftige Schnittstelle zum Produktdatenaustausch
69 Seiten

D-90-05
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner: Formalismus zur Repräsentation von Geo-metrie- und Technologieinformationen als Teil eines Wissensbasierten Produktmodells
66 Seiten

D-90-06
Andreas Becker: The Window Tool Kit
66 Seiten

D-91-01
Werner Stein , Michael Sintek
Relfun/X - An Experimental Prolog Implementation of Relfun
48 pages

D-91-03
Harold Boley, Klaus Elsbernd, Hans-Günther Hein, Thomas Krause
RFM Manual: Compiling RELFUN into the Relational/Functional Machine
43 pages

D-91-04
DFKI Wissenschaftlich-Technischer Jahresbericht 1990
93 Seiten

D-91-06
Gerd Kamp
Entwurf, vergleichende Beschreibung und Integration eines Arbeitsplanerstellungssystems für Drehteile
130 Seiten

D-91-07
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner
TEC-REP: Repräsentation von Geometrie- und Technologieinformationen
70 Seiten

D-91-08
Thomas Krause
Globale Datenflußanalyse und horizontale Compilation der relational-funktionalen Sprache RELFUN
137 pages

D-91-09
David Powers and Lary Reeker (Eds)
Proceedings MLNLO'91 - Machine Learning of Natural Language and Ontology
211 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

Entwurf, vergleichende Bewertung und Integration eines Arbeitsplanerstellungssystems für Drehteile

Gerd Kamp

D-91-06
Document