DIPLOMARBEIT

Fallbasiertes Retrieval von Arbeitsplänen für rotationssymmetrische Drehteile

Peter Kammer

Februar 1994

Betreuer:
Dipl.-Inf. Stefan Weß

Universität Kaiserslautern
Fachbereich Informatik
AG Künstliche Intelligenz – Expertensysteme

Prof. Dr. Michael M. Richter

SFB 314, Künstliche Intelligenz und wissensbasierte Systeme

Kaiserslautern, Februar 1994

Erklärung

Hiermit erkläre ich, daß ich die vorliegende Diplomarbeit selbständig verfaßt und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Peter Kammer

Zusammenfassung

Bei der industriellen Fertigung von rotationssymmetrischen Drehteilen kommt es in hohem Maße auf die Effizienz der verwendeten Fertigungspläne an. Diese können durch eine ungeschickte Wahl der Abfolge der Fertigungsschritte zusätzliche Werkzeug- oder Aufspannungswechsel enthalten, welche die Bearbeitungszeit eines Werkstückes wesentlich verlängern und durch die so bedingte geringere Ausnutzung der Maschinenkapazitäten erhebliche Kosten verursachen.

Einfache Regeln lassen sich für diese Domäne kaum formulieren, denn schon durch relativ geringe Änderungen in der Problemspezifikation (wie z.B. Änderung der gewünschten Oberflächengüte an einem Ende des Werkstückes) kann die Erstellung eines vollständig neuen Fertigungsplanes notwendig werden.

Aus diesen Grund legen menschliche Experten die Abfolge der Fertigungsschritte aud der Basis der von ihnen bei anderen Werkstücken gemachten Erfahrungen fest, d.h. sie erinnern sich an frühere, ähnliche Aufgabenstellungen, und versuchen die dort verwendete Vorgehensweise auf die neue Situation zu übertragen.

Diese Vorgehensweise wird auch in fallbasierten Planungssystemen verwendet, indem man eine Menge von früheren Planungsaufgaben zusammen mit ihren Lösungen in einer Fallbasis speichert. Steht das System jetzt vor einer neuen Aufgabe, so versucht es in der Fallbasis ein früheres, ähnliches Problem zu finden und den zu diesem Problem gehörigen Plan so zu modifizieren, daß dieser auf das aktuelle Problem anwendbar ist.

Auf diese Weise ist es möglich, mit einer relativ kleinen Menge von Regeln auszukommen, die beschreiben, welche Schritte in einem bereits bestehenden Plan geändert, hinzugefügt oder gelöscht werden müssen, um ein bestimmtes Teilziel zu erreichen.

Diese Zielsetzung verfolgt das CAbPlan–System, daß innerhalb der Arbeitsplanung Abstraktion und fallbasierte Techniken zur Steuerung des Inferenzprozesses eines nichtlinearen, hiearchischen Planunssystems einsetzt und somit die Komplexität der Gesamtaufgabe reduziert.

In der vorliegenden Arbeit wird in diesem Rahmen eine Komponente zur Fallauswahl implementiert, deren Steuerung sowohl auf oberflächlicher als auch auf struktureller Ähnlichkeit zwischen zwei Werkstücken beruht. Zielsetzung dieser Zweiteilung ist eine Reduzierung der Anzahl der zu betrachtenden Fallbeispiele durch eine Indizierung der Fallbasis mit (schnellen) Algorithmen aufgrund von syntaktischen Übereinstimmungen bestimmter Attributwerte; anschließend werden diese (wenigen) Fälle mit Hilfe tiefergreifender Analyseschritte nach maximaler Übereinstimmung mit der Aufgabenstellung durchsucht. Des weiteren wird eine Komponente zur Verfügung gestellt, die hilft, frühere Planungsfehler durch Wahl geeigneter Fallbeispiele zu vermeiden.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Betrachtet man die aus der Literatur bekannten Planungsansätze, so sind zwei Grundströmungen zu beobachten. Einerseits existieren die klassischen suchraumorientierten Verfahren, die den Lösungsraum systematisch durchsuchen und dabei Verfahren zur gezielten Suchraumbeschränkung einsetzen, z.B. STRIPS [6], NONLIN [29] oder SIPE [38]. Andererseits existieren auch eine Reihe von wissensintensiven Ansätzen, wie etwa Skelettpläne in MOLGEN [7], Fallbeispiele in MEDIATOR [24] oder CHEF [8], die eine alternative, eher vorgehensorientierte Sichtweise verfolgen und sich daher mehr auf Steuerungsaspekte von Planungsprozessen konzentrieren, den Planungsablauf also nicht expliziet festlegen.

Für Aufgabenstellungen aus dem Expertensystembereich sind fallbasierte Techniken zur Repräsentation und Verarbeitung von Wissen ein neuer, vielversprechender Ansatz zur Realisierung von wissensbasierten Systemen. Ziel des fallbasierten Problemlösens ist es, ein aktuell vorliegendes Problem direkt auf der Basis von bereits bekannten Fallbeispielen zu lösen.

Im Rahmen des SFB–Projektes X6 [1] wurde in Kaiserslautern mit MOLTKE ein System geschaffen, das Wissen in Form von Fallbeispielen gleichberechtigt neben anderen Wissensarten zum diagnostischen Problemlösen verwendet. Das in MOLTKE integrierte fallvergleichende System PATDEX [36] kann Diagnosen durch den direkten Vergleich mit geeigneten Fallbeispielen ermitteln.

Nachdem im Laufe des MOLTKE–Projektes erste positive Erfahrungen mit fallbasierten Ansätzen im Bereich der technischen Diagnose gemacht wurden, wird in Zusammenarbeit mit dem CIM[1]–Centrum Kaiserslautern (CCK) in dem Nachfolgeprojekt X9 mit CAbPlan[2] [37] versucht, fallbasierte Techniken in einer Planungsdomäne aus dem Bereich des Maschinenbaus anzuwenden. Als Anwendungsdomäne wurde dabei in der Teilefertigung die Arbeitsplanerstellung für rotationssymmetrische Drehteile gewählt. Für synthetische Aufgabenstellungen wie Planung oder Design ist jedoch im Gegensatz zu PADTEX ein fallmodifizierender Ansatz zu wählen, da sich die zu lösenden Aufgabenstellungen und damit ihre Lösungen fast stets unterscheiden. Ein erster Prototyp [12] verfolgte den lösungsorientierten *Transformational Analogy* – Ansatz nach [2]. Hier zeigten sich vor allem die folgenden Defizite:

1. Trennung von Retrieval und Modifikation
   Eine strenge Trennung von Retrieval- und Modifikationswissen ist nicht möglich, da für das Retrieval eines geeigneten Fallbeispiels Wissen über die notwendigen Modifikationen erforderlich ist.

2. Informationsverlust
   Entscheidungsalternativen und Begründungen für die Wahl einer bestimmten Alternative und während der Planung aufgetretene Fehlschläge, die eine gezielte Modifikation der Lösung ermöglichen könnten, werden im Fallbeispiel nicht repräsentiert.

[1] Computer Integrated Manifacturing

[2] Der Name CAbPlan setzt sich aus der Abkürzung CAP = Computer Aided Planning und Case-based Planning in mechanical Engineering zusammen

Das Planungssystem CAbPlan soll nun aufgrund der initialen Aufgabenbeschreibung in Form eines zu fertigenden Werkstücks aus der Fallbasis geeignete Fälle auswählen und durch ein Replay der übertragbaren Teilschritte Fallwissen in die aktuelle Situation übernehmen.

Die voliegende Arbeit enthält in diesem Rahmen eine komplexe Retrievalkomponente, die zunächst das Werkstück analysiert und die zu lösenden Ziele festlegt. Anschließend wird mit Hilfe dieser Ziele ein vorläufiges Planskelett aufgestellt, wobei verschiedene „Spezialisten“ in Form voneinander unabhängiger Entscheidungsnetze eine Bearbeitungsstrategie festlegen. Jeder dieser Spezialisten trifft anhand der zu erfüllenden Ziele eine Entscheidung auf seinem Spezialgebiet, wobei diese Entscheidung als ein Beitrag zur Indizierung der Fallbasis herangezogen wird. Diese führt zu einer Teilmenge von Fällen, deren Bearbeitungsstrategien gleich oder ähnlich zu denen sind, die das Problem lösen. Aus dieser Menge wird nun durch *Tree–Mapping* derjenige Fall ausgewählt, der nach der Definition des verwendeten Algorithmus dem Problem am „ähnlichsten“ ist.

Um den oben erwähnten Nachteil des Prototyps, daß nämlich keine Information über eventuell auftretende Fehlschläge während der Planung bzw. daß keine Entscheidungsbegründungen aufgenommen werden, zu vermeiden, wird (ähnlich wie beim CHEF–System [8] von C. Hammond) ein *Anticipator* in die Entscheidungsnetze integriert, der bei Aufteten einer bestimmten Situation eine „Umleitung“ des vom jeweiligen Spezialisten an einer bestimmten Stelle im Entscheidungsnetz getroffenen Entscheidung vornimmt. Diese Komponente stellt neben der (mit der Anzahl der gelösten Aufgabenstellungen wachsenden) Fallbasis die eigentliche Lernkomponente des Planungssystems dar. In der hier vorliegenden Implementation des *Retrievers* und des *Anticipators* erfolgt die Aktivierung der einzelnen Anticipatoreinheiten im Dialog mit dem Benutzer, weil die eigentlichen Planungskomponenten, die im Planungsprozeß erkannte Fehlschläge analysieren und so Einfluß auf den *Anticipator* nehmen können[3], noch nicht implementiert sind.

Die vorliegende Arbeit gliedert sich in folgende Kapitel:

Kapitel ?? gibt einen Überblick über das fallbasierte Schließen mit dem Schwerpunkt auf fallbasierter Planung. In diesem Zusammenhang wird die prinzipielle Vorgehensweise eines fallbasierten Planungssystems am Beispiel von CHEF als bekanntestem Vertreter sowie die Zielsetzung des CAbPlan– Projektes vorgestellt. Es erfolgt eine formale Definition des Begriffs der „Ähnlichkeit“ zweier Objekte und eine Einführung in die Mapping–Algorithmen *Tai–Mapping* und *Structure Preserving Mapping*, die zwei Baumstrukturen ineinander überführen. Abschließend wird die Bedeutung von Expertensystem in CIM erläutert, wobei speziell auf ihre Rolle bei der Arbeitsplanung eingegangen wird.

In Kapitel ?? wird die Arbeitsdomäne Drehen als Teilbereich der Arbeitsplanung vorgestellt; dieses Kapitel ist als Einführung in diesen Bereich des Maschinenwesens gedacht und soll helfen, die bei der Arbeitsplanung auftretenden Begriffe und Fachausdrücke sowie die physikalischen Hintergründe besser zu verstehen.

Die Datenstrukturen zur Repräsentation der Werkstücke werden in Kapitel ?? beschrieben. Es erfolgt eine Darstellung der einzelnen Phasen der Fallauswahl (Analyse zur Erzeugung von Goals, Indizierung der Fallbasis über das semantische Netzwerk, Mapping von Baumstrukturen sowie die Anticipatorkomponente zur Vermeidung von bekannten Fehlern). Diese Komponenten werden dann im Kapitel ?? im einzelnen vorgestellt.

Im Kapitel ?? wird eine Bewertung des vorliegenden Systems getroffen sowie eine mögliche Weiterentwicklung bzw. Integrität in bereits bestehende Komponenten des CAbPlan–Systems untersucht.

Die Objektklassen zur Darstellung der Werkstücke werden im Anhang aufgeführt. Es folgt eine Beschreibung der Benutzeroberfläche und der integrierten Tools. Für die in Kapitel ?? vorgestellten Werkmittel werden in Anhang ?? verschiedene Tabellen für DIN–Einteilungen angegeben, sowie in Anhang ?? einige allgemeine Tabellen. Abschließend folgt eine Tabelle mit englischsprachigen Entsprechungen von Begriffen der Arbeitsdomäne Drehen.

[3] Beim CHEF–System wird diese Aufgabe vom *Assigner* erfüllt (s. ??)

Kapitel 2

Fallbasiertes Schließen

Fallbasiertes Schließen (*Case Based Reasoning*) bedeutet, alte Erfahrungen anzuwenden, um neue Probleme zu verstehen bzw. zu lösen. Beim fallbasierten Schließen erinnert sich ein „Experte“ an eine frühere Situation, die ähnlich der aktuellen ist, und benutzt diese, um das neue Problem zu lösen. Wenn wir uns anschauen, wie Menschen Probleme lösen, stellen wir fest, daß fallbasiertes Schließen überall angewendet wird. Beispielsweise wird ein Arzt mit einem Patienten konfrontiert, dessen Krankheit eine ganz ungewöhnliche Kombination von Symptomen aufweist. Wenn er früher einen Patienten mit ähnlichen Symptomen behandelte, ist es wahrscheinlich, daß er sich an den alten Fall erinnert und die alte Diagnose als Basis zur Lösung des neuen Problems verwendet. Wenn man davon ausgeht, daß eine neue Diagnose zeitintensiv ist, kann man nun von einer großen Zeitersparnis sprechen, indem auf frühere Ergebnisse zurückgegriffen wird. Allerdings kann der Arzt nicht versichern, daß die alte Lösung direkt für den aktuellen Fall übernommen werden kann; er muß die Diagnose an den neuen Fall anpassen.

Allgemein kann man sagen, daß es einfacher ist, ein Problem auf der Basis eines Erfahrungsschatzes zu lösen, indem man sich an die frühere Lösung erinnert und sie in geeigneter Form nachvollzieht. Man ist beim zweitenmal auch deshalb kompetenter, weil man sich an frühere Fehler erinnert und sich bemüht, sie zu umgehen.

Die Qualität eines Fallbasierten Schließers hängt von vier Dingen ab:

- den Erfahrungen, die er gesammelt hat
- seinen Fähigkeiten, neue Situationen in Termen dieser alten Erfahrungen zu verstehen
- seiner Erfahrung in der Anpassung an die neue Situation (Modifikation)
- seiner Erfahrung in der Bewertung der Modifikation

Ein unerfahreneres „fallbasiertes System“[1] wird immer einen geringeren Erfahrungsschatz haben, auf den es zurückgreifen kann, als eines, das schon länger im Einsatz ist und somit mehr gelernt hat. Aber es zeigt sich, daß die Antworten, die ein unerfahrenes System liefert, nicht notwendigerweise schlechter sein müssen als die von einem System, das mehr Erfahrung besitzt, wenn es in seinem Verstehen und der Anpassungsfähigkeit kreativ genug ist. Jedes Programm, das wir zum automatischen Fallbasierten Schließen schreiben, muß im Kern eine Komponente zur Speicherung repräsentativer Erfahrung besitzen. Diese Erfahrungen (Fälle) sollten die Ziele umfassen, die beim Schließen auftreten, und sollten sowohl erfolgreiche als auch gescheiterte Versuche beinhalten, diese Ziele zu erreichen. Erfolgreiche Versuche werden benutzt, um Lösungen für neue Probleme zu finden, nachdem gescheiterte Versuche vor möglichen Fehlern gewarnt haben und die entsprechenden Fälle von der Wahl ausgeschlossen haben.

Der zweite Stichpunkt, der besagt, daß das neue Problem in Termen gespeicherter Erfahrung verstanden werden muß, besteht aus zwei Teilen: alte Erfahrungen aufzurufen (*recalling*), und die

[1] Hierbei kann ein *System* sowohl ein natürliches System (Mensch, Tier) als auch ein Expertensystem sein

neue Situation in Begriffen dieser aufgerufenen Erfahrungen zu interpretieren (*interpreting*). Der erste Teil wird auch das *indexing*–Problem genannt. Mit einfachen Worten ausgedrückt bedeutet es, im Speicher die Erfahrung zu suchen, die am nächsten an einer neuen Situation liegt. Entsprechende Fälle aufrufen ist der eigentliche Kern des Fallbasierten Schließens (Retrieval). Unter Interpretation verstehen wir den Prozeß, bei dem die neue Situation mit dem ausgewählten Fall verglichen wird. Wenn Problemsituationen interpretiert werden, werden sie alten Problemlösungen gegenübergestellt. Das Ergebnis ist eine Interpretation der neuen Situation mit zusätzlich gefolgertem Wissen oder eine Klassifikation der neuen Situation. Allgemein werden Interpretationsprozesse benötigt, wenn Probleme nicht klar erkannt werden oder eine Lösung kritisiert werden muß.

Der dritte Punkt, die Erfahrung in der Modifikation, ist der Prozeß der Anpassung einer alten Lösung an die neuen Anforderungen. Hierbei werden meist Teile der alten Lösung gestrichen bzw. ersetzt oder es werden zusätzliche Lösungsschritte eingefügt. Die Anwendung von Modifikationsstrategien führt direkt zu korrekten, aber oft nicht überraschenden Antworten, da prinzipiell die alte Lösungsstruktur übernommen wird.

Eine wesentliche Komponente eines Systems zum fallbasierten Schließen ist seine Fähigkeit, aus Erfahrung zu lernen. Um aus Erfahrung lernen zu können, benötigt das System eine Form von Feetback, um interpretieren zu können, was an seiner Lösung richtig oder falsch war. Ohne Feetback würde das System schneller bei der Problemlösung werden, aber es würde seine Fehler wiederholen und seine Fähigkeiten nicht erweitern. Deshalb sind Auswertung (*evaluation*) und konsequentes Reparieren (*repair*) wichtige Beiträge zur Erweiterung der Sachkenntnis des Fallbasierten Planers. Die Auswertung kann im Zusammenhang mit Ergebnissen anderer ähnlicher Fälle ausgeführt werden oder auf Simulation basieren.

2.1 Unterteilung von fallbasiertem Schließen

Es gibt zwei Arten von Fallbasiertem Schließen: das Lösen von Problemen und die Interpretation von Situationen.

Beim Problemlösen mit Fallbasiertem Schließen werden *Lösungen* zu neuen *Problemen* abgeleitet, indem alte Lösungen als Vorlage dienen. Alte Lösungen können annähernd korrekte Lösungen zu neuen Problemen sowie Warnungen vor möglichen Fehlern und Mißerfolgen liefern.

Bei der *Interpretation* werden neue *Situationen* im Zusammenhang mit alten Situationen ausgewertet. Ein Richter beispielsweise verwendet interpretatives Fallbasiertes Schließen an, wenn er eine Reihe alter Fälle benutzt, um einen Urteilsspruch in einem neuen Fall zu rechtfertigen. Dabei kann interpretatives Fallbasiertes Schließen auch beim Problemlösen benutzt werden.

Im folgenden wollen wir uns nur noch mit Fallbasiertem Schließen bei der Problemlösung beschäftigen. Es wird erfolgreich für eine Vielzahl von Anwendungsgebieten wie Planen, Diagnose oder Design eingesetzt.

2.2 Fallbasiertes Planen

Unter *Planen* versteht man den Prozeß, eine Folge von Aktionsschritten zu erzeugen, um einen bestimmten Zustand der „Welt" zu erreichen.

Es gibt eine Vielzahl von Problemen, die beim Planen beachtet werden müssen. Das wohl wichtigste ist das Auftreten von Interaktionen. Hierunter versteht man Wechselwirkungen zwischen Planungsschritten, die sich gegenseitig behindern bzw. die ihre Ergebnisse aufheben können (negative Interaktion), im Zusammenwirken aber auch nützliche Effekte haben können (positive Interaktion), wenn nämlich Planungsschritte im Zusammenwirken komplexere Ziele lösen. In guten Plänen sind die einzelnen Planschritte aufeinander abgestimmt, d.h. spätere Schritte zerstören nicht die Ergebnisse von früheren Schritten bzw. Vorbedingungen späterer Schritte im Plan werden nicht durch frühere Schritte verletzt.

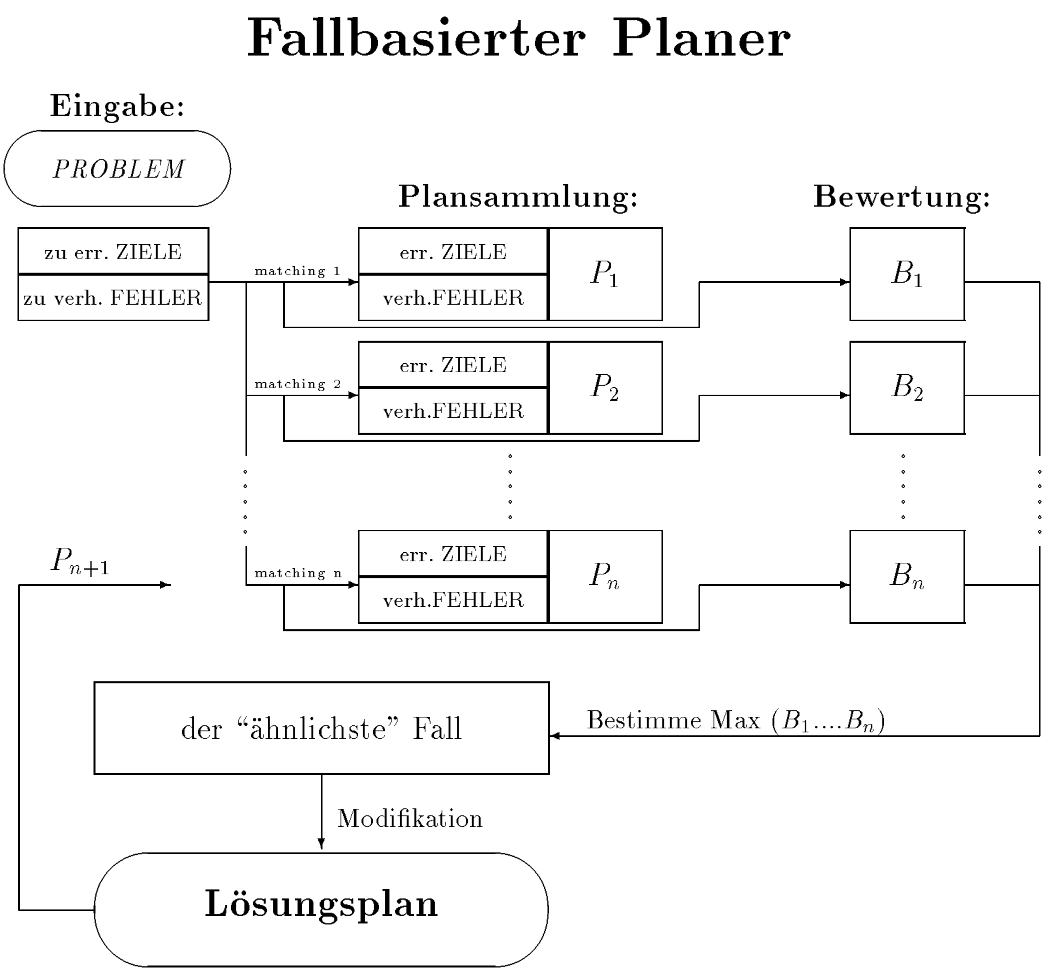


Abbildung 2.1: Fallbasiertes Planungssystem

Weiterhin gibt es das Problem der Vorbedingungen. Ein Planer muß garantieren können, daß die Vorbedingungen eines Planschrittes erfüllt sind, bevor er ausgeführt werden kann. Somit bringt das Planen Festlegungsalgorithmen mit sich, die neben der Festsetzung der Reihenfolge der eigentlichen Planungsschritte eine sinnvolle Reihenfolge der Vorbedingungen dieser Planungsschritte erkennt. Wenn diese beiden Probleme gleichzeitig von einem traditionellen Planungssystem gelöst werden sollen, muß ein beachtlicher Rechenaufwand geleistet werden [9].

Fallbasiertes Planen (*Case Based Planning*) umgeht diese Probleme, indem es Ergebnisse liefert, die auf bereits ausgeführten Plänen beruhen, bei denen diese Probleme bereits gelöst sind.

In [14] werden einige Vorteile eines fallbasierten Planers gegenüber einem traditionellen Planungssystem zusammengefaßt:

- Fallbasiertes Planen erlaubt es, Lösungen zu Problemen aus Bereichen zu finden, die nicht komplett verstanden werden können

  Es gibt Fälle, in dem einem Expertensystem nicht das nötige Wissen über seine Domäne mitgegeben werden kann, weil entweder viel von einem menschlichen subjektiven Eindruck abhängt (z.B. in der Ökonomie) oder weil sie noch nicht vollständig erforscht ist (z.B. der Einfluß eines Medikaments bei einer bestimmten Krankheit). Fallbasiertes Planen liefert auch in solchen Fällen eine Antwort, die darauf basiert, was in der Vergangenheit in einem ähnlichen Fall geschah.

- Fallbasiertes Planen liefert Lösungen, auch wenn keine algorithmischen Methoden zur Verfügung stehen

  Bei der Auswertung von Problemfällen mit vielen Unbekannten, die normalerweise eine Lösung unmöglich oder zumindest sehr schwer machen würden, ist das Benutzen von Fällen sehr hilfreich. Situationen werden im Zusammenhang mit früheren ähnlichen Situationen ausgewertet. Der Planer greift auf Lösungen zurück, die in der Vergangenheit schon einmal erfolgreich ausgeführt wurden.

- Durch das Erinnern an frühere Situationen kann leicht vor potentiellen Fehlern gewarnt werden

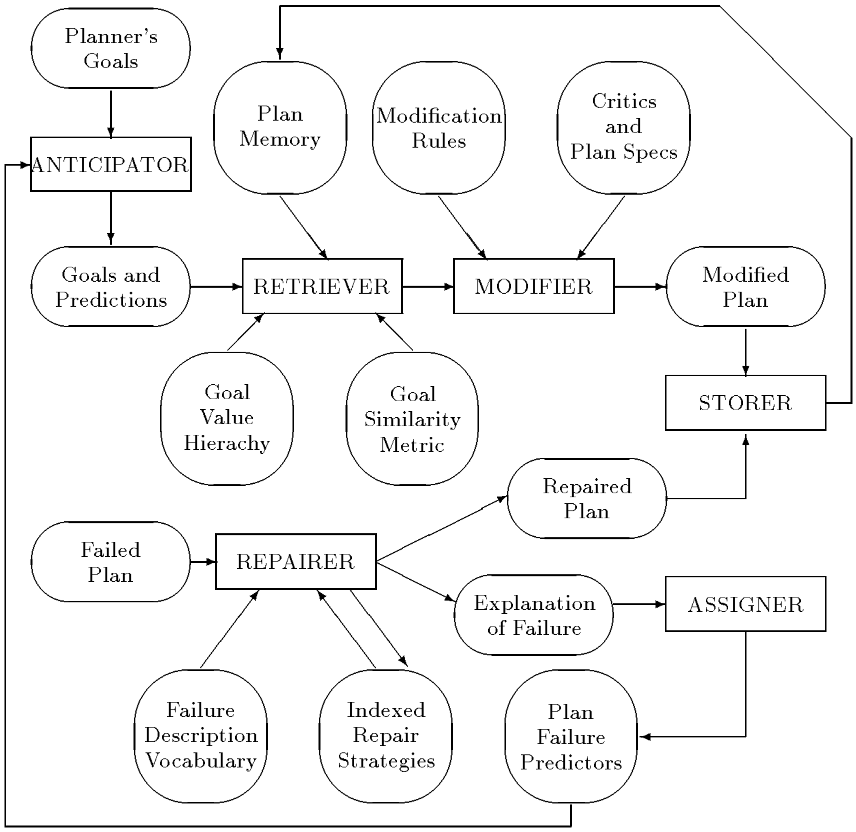
  Gespeicherte Erfahrung kann erfolgreiche Versuche darstellen, aber auch das Aufreten von Fehlern oder Eintreten von Situationen, die so nicht geplant waren. In diesem Fall wird der Planer gewarnt, diese bei einem neuen Problem zu vermeiden.

Im Folgenden werden die beiden Planungssysteme CHEF und CAbPlan vorgestellt. Anhand des CHEF-Systems lassen sich die einzelnen Komponenten eines fallbasierten Planers gut aufzeigen, die im Prinzip in allen Implementationen enthalten sind. Die voliegende Arbeit ist Teil des CAbPlan–Projektes, deshalb werden der Hintergrund und die Zielsetzung dieses Ansatzes zur fallbasierten Arbeitsplanung in ?? dargestellt.

2.3 CHEF

> *„The essential idea of CHEF is that planning is interaction with the world and that interaction can lead to learning.“*

CHEF ist ein fallbasierter Planer, der von Kristian J. Hammond am *Department of Computer Science* an der *University of Chicago* 1988 entwickelt wurde. Zielsetzung war, so oft wie möglich auf gespeichertes Wissen zuzugreifen. CHEF beginnt den Planungsablauf damit, Wissen über frühere Fehlschläge dazu zu benutzen, um vor Problemen zu warnen (*Anticipator*). Dann benutzt er seine Erinnerung an erfolgreiche Lösungen, um den besten Plan zu finden (*Retriever*), der passend auf die aktuellen Ziele und Probleme modifiziert wird (*Modifier*). Wenn eine Fehlplanung auftritt, behandelt CHEF sie als falsches Verständnis seiner Welt und versucht, den Fehler zu erklären. Damit kann er den Plan reparieren (*Repairer*) und gleichzeitig sein Wissen über die Welt verändern. Schließlich speichert er erfolgreiche Versuche in der Plansammlung (*Storer*), die sowohl



durch die Ziele, die erfüllt werden, als auch durch die Probleme, die vermieden werden, indiziert wird. Auf dieses Wissen kann bei einem ähnlichen Planungsproblem dann zugegriffen werden.

Die Planungsdomäne von CHEF ist das Kochen. Seine Aufgabe besteht darin, neue Rezepte auf der Basis von Speisewünschen des Benutzers mit speziellen Zutaten und Geschmacksrichtungen zu erstellen. Bedingt durch diese Domäne stellt sich dem CHEF–System die Aufgabe, einen einzigen Plan (Kochrezept) zu liefern, der eine ganze Reihe von Zielen erfüllt. Von daher läßt sich der Einsatzbereich von CHEF mit der Arbeitsplanung vergleichen, die gleichwohl viele untereinander interagierende Ziele gleichzeitig erfüllen muß.

2.3.1 Die einzelnen Planungskomponenten

Retriever

Der *Retriever* ist ein Modul, welches uns aus einer Menge von alten Plänen **einen**[2] Plan liefert, der den gestellten Anforderungen (*goals*) weitestgehend nahekommt; sollten allerdings nicht alle geforderten Ziele erfüllt werden, so wird zumindest ein Plan geliefert, der möglichst viele Ziele erfüllt (*best match*). Um den besten Plan zu bestimmen, müssen die alten Pläne im Speicher entsprechend indiziert werden, d.h. für jeden Plan muß eine vollständige Auflistung sowie eine mögliche Generalisierung der Ziele, die er erfüllt, vorhanden sein. Generalisierungen sind notwendig, um Ähnlichkeiten von Lösungsstrategien festzuhalten. Von Bedeutung sind entsprechende Rahmenbedingungen für die Durchführung eines Planes. Sind Ziele von solchen Bedingungen abhängig, so müssen diese auch mitverwaltet werden. Die Ziele an sich werden, da sie von unterschiedlicher Wichtigkeit zur Realisierung eines Gesamtzieles sind, in eine Bewertungshierarchie eingetragen. Für den *Retriever* sind zusammenfassend also folgende Beschreibungen wichtig:

[2] Es wird im CHEF–System tatsächlich genau 1 Plan als Ausgabe geliefert.

1. Eine Plansammlung, indiziert nach den Zielen, die die Pläne erfüllen.
2. Ein Ähnlichkeitsmaß für Ziele; Abstraktionshierarchie (Generalisierungen).
3. Eine Bewertungshierarchie für die Ziele (unterschiedliche Gewichtung).
4. Spezifikationen der entsprechenden Rahmenbedingungen.

Die Konsequenzen, die sich aus Interaktionen zwischen verschiedenen Zielen oder Rahmenbedingungen und Zielen ergeben, werden in dieser Konzeption noch nicht berücksichtigt.

Modifier

Da der *Retriever* in den meisten Fällen nicht alle erwünschten Ziele erfüllt, muß der gelieferte Plan verändert werden; diese Aufgabe übernimmt der *Modifier*.

Einem unvollständigen Plan müssen Arbeitsschritte hinzugefügt werden, um noch nicht erfüllte Teilziele zu erreichen. Es steht eine Sammlung von Modifikationsregeln, nach Zielsetzungen klassifiziert, zur Verfügung. Es handelt sich dabei allerdings nicht um komplette Lösungswege, sondern nur um Schritte, die dazu dienen, den existierenden Plan bezüglich eines Teilzieles zu verändern. Weiterhin benötigt der *Modifier* Informationen über spezifische Objekte seiner Domäne, um unter bestimmten Rahmenbedingungen der Pläne entsprechende geforderte Features dieser Objekte zu verwenden. Die generell zu erreichenden Ziele dürfen durch eine Modifikation auf keinen Fall in Frage gestellt werden (generelle Planspezifikation).

Retriever und *Modifier* bilden den Kern eines fallbasierten Planungssystems, der *Retriever* selektiert den für ihn bestmöglichen Plan und der *Modifier* verändert ihn bezüglich der noch nicht erreichten Ziele. Dabei werden folgende Operationen ausgeführt:

1. Hinzufügen von Arbeitsschritten.
2. Löschen von Arbeitsschritten.
3. Austauschen von Arbeitsschritten.

Storer

Das fallbasierte Planen beschränkt sich nicht nur auf das Selektieren und Modifizieren, denn es werden hierbei keine neuen Erkentnisse gespeichert, d.h. es findet noch kein Lernprozess statt. Dazu ist es notwendig, die Ergebnisse der bearbeiteten Fälle festzuhalten.

Diese Funktion übernimmt der *Storer*. Die erfolgreich modifizierten Pläne werden in die Plansammlung, auf die der *Retriever* zugreift, übernommen. Durch ständige Nutzung des Planungssystems erweitert sich die Wissensbasis des *Retrievers*, die Plansammlung wächst, was wiederum dem *Retriever* mehr Möglichkeiten bei der Auswahl eines Planes gibt und so flexiblere Vorauswahlen zuläßt.

Repairer

Unabhängig von der Größe oder Qualität eines Planungssystems kann es Kombinationen von Zielen geben, für die von *Retriever* und *Modifier* keine erfolgreiche Lösung angeboten werden kann, da die Domäne immer nur eine beschränkte Abbildung der Realität sein kann.

Der Mechanismus *Repairer* soll solche „Fehlplanungen“ aufarbeiten. Dabei sind als Eingaben der entsprechende Plan sowie eine Beschreibung der aufgetretenen Fehler notwendig. Die Fehlerbeschreibung kann auf folgende Art und Weise gewonnen werden:

1. Ausführen des Planes und Begutachtung des Ergebnisses.
2. Simulation und Ergebnis zur Diagnose verwenden.
3. Außenstehende Quellen befragen.

Die festgestellten Fehler werden im folgenden Schritt vom *Repairer* unter Verwendung eines speziellen Vokabulars[3] genauer beschrieben. Die Ursachen für das Eintreten der Fehler werden ebenfalls beschrieben, hierzu ist eine Analyse der möglichen Seiteneffekte und deren Folgen notwendig. Basierend auf diesem Vokabular verfügt der *Repairer* über korrespondierende Reperaturstrategien, die in einem weiteren Schritt zur Anwendung kommen. Der so „reparierte“ Plan wird vom *Storer* zuzüglich der Anmerkungen und Verbesserungsvorschläge zur Plansammlung hinzugefügt.

Damit werden die Pläne nicht nur durch die Ziele, die sie erreichen, sondern zusätzlich durch die Probleme, die dem Erreichen dieser Ziele entgegenwirken, indiziert.

Assigner

Für ein fallbasiertes Planungsystem ist es nicht ausreichend, nur die Erfüllung der Ziele im Auge zu behalten, sondern auch Fehlerquellen zu klassifizieren und beim Retrieval zu berücksichtigen beziehungsweise zu umgehen. Zur Klassifizierung werden die Rahmenbedingungen, unter denen ein Fehler aufgetreten ist, festgehalten. Diese Aufgabe übernimmt der *Assigner*, welcher auf den Beschreibungen des *Repairers* zurückgreift.

Sollte bei späteren Planungsvorgängen ähnliche Rahmenbedingungen gegeben sein, wie sie bei einem aufgetretenen Fehler bestanden haben, so läßt sich mit den Ausgaben des *Assigners*[4] ein vermeidbares Scheitern der Planung durch geeignete Maßnahmen verhindern.

Anticipator

Die vom *Assigner* gelieferten Informationen werden in einem letzten Schritt vom *Anticipator* dazu verwendet, die Liste der eingegebenen erwünschten Ziele um die zu vermeidenden Probleme, die sich aus den spezielen Rahmenbedingungen ergeben, zu erweitern und so der *Retriever* unter den Zwang gestellt wird, diese bei seinem Retrieval zu berücksichtigen, was dann unter Umständen zu neuen Problemen führen kann, aber verhindert, daß der gleiche Fehler noch einmal gemacht wird.

Der *Retriever* wird also nicht nur einen Plan unter dem Aspekt suchen, schon möglichst viele Ziele zu erfüllen, sondern auch möglichen Fehlerursachen weitestgehend auszuweichen.

2.3.2 Vorgehensweise von CHEF

Im folgenden wird die Vorgehensweise des CHEF–Systems anhand eines Beispiels von Hammond [8] dargestellt; hierbei soll der Planer ein Rezept erstellen, das gebratenes Rindfleisch mit Broccoli enthält (BEEF-WITH-BROCCOLI). Die Ausgaben des Planungssystems während den einzelnen Planungsphasen befinden sich in Anhang ??.

Eingabe ist eine Menge von Zielen, die erfüllt werden müssen; im Beispiel sind es die drei Ziele

1. Gericht wird gebraten
2. Gericht enthält Rindfleisch
3. Gericht enthält Broccoli

CHEF sucht nun nicht für jedes Ziel einzeln nach einem Plan in der Fallbasis, der es erfüllt, und mischt diese dann, sondern sucht nach dem Plan, der so viele Ziele wie möglich erfüllt.

Im Beispiel entscheidet sich CHEF für das Rezept BEEF-WITH-GREEN-BEANS, das die ersten beiden Ziele der Problemstellung erfüllt.

Wurde ein Plan in der Plansammlung gefunden, so wird er so modifiziert, daß er auch die Ziele erfüllt, die noch offen sind. Die Modifikation wird gesteuert sowohl von Regeln, die abhängig sind von der Art des gewünschten Gerichtes (gebraten), als auch von den Zielen, die noch offen sind (enthält Broccoli). CHEF ersetzt nun direkt das vom Plan erfüllte Ziel durch das im Problem

[3] Hiermit ist eine symbolische Sprache gefordert, die eine entsprechende Indizierung impliziert, so daß sowohl der Planer als auch der Anwender diese Beschreibung nachvollziehen kann.

[4] Plan Failures Predictors

gegebene, ersetzt also die Bohnen durch Broccoli; dabei greift er auf sog. *object critics* zu, die Strukturen zur Speicherung von Attributen über die Entities der Domäne darstellen, um so etwa die Kochzeit von Broccoli anstelle der von Bohnen einzusetzen.

Wurden diese Modifikationen durchgeführt, so erzeugte er prinzipiell einen Plan, der das gewünschte Gericht liefert. Dieser hat die gleiche Form wie der ursprüngliche Fall, der aus der Plansammlung ausgewählt wurde: eine Menge von Zutaten, eine Menge von Aktionen, die durchzuführen sind und eine Menge von Zielen, die der neue Plan nun erfüllen soll.

Pläne sind aber oftmals in dieser Phase der Planung noch nicht korrekt; das modifizierte Rezept BEEF-WITH-GREEN-BEANS stellt keine Lösung der Aufgabe BEEF-WITH-BROCCOLI dar, weil eine Übernahme der Zubearbeitungsart „koche das Gemüse zusammen mit dem Fleisch" dazu führt, daß der Broccoli verkocht; der Planer muß also erkennen können, daß in diesem Fall ein zusätzliches Ziel „Gemüse muß knusprig sein" in die initiale Aufgabenstellung aufgenommen werden muß. An dieser Stelle reicht eine einfache Ersetzung von Constraints (Bohnen durch Broccoli) nicht aus.

Scheitert ein Plan, so war das Verständnis des Planers über seine Domäne falsch; es wurde erwartet, daß der Plan ein Resultat erreicht, was er aber nicht tut. Eine nun ansetzende Fehlererklärung hat die Funktionen, den Teil des Plans zu identifizieren, der verändert werden muß, und die Stellen in der Wissensbasis des Planers zu erkennen, an denen es zu Fehlschlüssen kommt.

Dazu startet CHEF einen Simulator, der die einzelnen Schritte des erzeugten Plans nachvollzieht und der in jeder Phase der Planausführung die momentanen Zustände mit den vom *Modifier* erzeugten Zielen vergleicht. Auf diese Weise gelingt es dem System, den Fehler zu lokalisieren. Es muß nun in der Lage sein, das Auftreten des Fehlers zu erklären, um eine Planreparatur durchzuführen.

Eine solche Erklärung besteht aus

- dem Zustand, der vor dem Fehler gilt
- dem Schritt, der in dem Fehler resultiert
- den Bedingungen, die eigentlich erfüllt werden müßten

Die Erklärung stellt also eine Beschreibung dessen dar, was der Planer versuchte zu erfüllen, als der Fehler auftrat.

CHEF benutzt Inferenzregeln, die die Auswirkungen eines Planungsschrittes auf die einzelnen Entities der Domäne beschreiben (die die gleichen sind wie bei der Simulationsphase[5]); mit Hilfe dieser Regeln erkennt er die Art des Fehlers, warum er auftrat und welche Ziele von den Planungsschritten betroffen sind, die aus dem Fehler resultieren.

Vom lokalisierten Fehler aus bewegt sich CHEF nun rückwärts durch die Zusände und Schritte des Plans hindurch und baut somit eine kausale Kette auf, die zum Fehler hinführt. Gleichzeitig werden die initialen Ziele der Planungsaufgabe erkannt, die schließlich ein Scheitern zur Folge hatten.

Diese Fehlererklärung wird benutzt, um Strukturen in der Wissensbasis zu finden, die das beschriebene Problem auflösen. Diese Strukturen werden *Thematic Organization Packets* (TOPs) genannt[6]. Jedes TOP wird durch eine bestimmte Art von Planungsproblem indiziert und organisiert eine Reihe von Strategien, um diese zu beheben. Diese Strategien besitzen die Form allgemeiner Reperaturregeln wie „REORDER <*steps*>" oder „RECOVER from <*side-effects*>". Jede generelle Strategie wird ersetzt durch die Spezifikationen des aktuellen Problems, um so die Beschreibung einer Änderung im Plan zu erzeugen. In den TOPs gibt es eine Vielzahl von Planungsinformation über Interaktionen zwischen Plänen und Zielen. Diese Information kann nicht einem individuellen Ziel oder Planungsschritt zugeordnet werden, vielmehr bezieht sie sich auf Probleme, die aus ihrer Kombination entstehen. Hier liegt ein Hauptproblem beim Planen, nämlich in den Interaktionen zwischen mehreren Planungsschritten. TOPs bieten eine Möglichkeit, diese

[5] Aus diesem Grund kann CHEF Fehler immer erklären, wenn sie bei der Simulation erkannt wurden
[6] Diese entsprechen den *critics* in HACKER und NOAH

Informationen abzuspeichern. Jedes TOP bezieht sich auf ein Planungsproblem mit seinen Interaktionen zwischen den Schritten und den Zuständen.

Wurde dem Problem ein bestimmtes TOP zugeordnet, versucht CHEF nun, die hierin vorgeschlagenen Strategien anzuwenden, indem er für den aktuellen Plan Modifikationen sucht, die diese erfüllen.

Nachdem CHEF nun das unmittelbare Problem gelöst hat, ändert er auch sein Verständnis der Welt, so daß er ähnliche Probleme in Zukunft vermeiden kann. Hierzu muß CHEF herausfinden, was den Fehler ausgelöst hat. Er muß entscheiden, welche Teile der Ausgangssituation den Fehler nach sich zogen, so daß spätere Auftreten ähnlicher Ziele hiervon Gebrauch machen können. Hier greift CHEF wieder auf die kausale Erklärung zurück. Alle Entities, die für das Problem verantwortlich gemacht wurden, werden soweit generalisiert, wie es die Erklärungskette zuläßt. Im Beispiel kann „Rindfleisch“ durch „Fleisch“ generalisiert werden, weil CHEF das Scheitern seines Planes der Tatsache zuschreibt, daß beim Braten des Rindfleischs Saft austritt, was bei allen Fleischsorten der Fall ist. Eine höhere Stufe der Generalisierung wäre nicht erlaubt.

CHEF speichert nun den Plan, indem er hierzu zwei Strukturarten benutzt. Die erste ist das Standartvokabular der vom Plan erfüllten Ziele, etwa welche Zutaten und welcher Geschmack darin vorkommen, aber auch um welche Zubereitungsart es sich handelt.

Eine zweite Speicherstruktur beinhaltet ein anderes Vokabular, daß die beim Erstellen des Plans aufgetretenen Fehler beinhaltet. Das bedeutet, daß der Plan durch mögliche Probleme zwischen Planungsschritten indiziert wird. Wird ein Plan auf diese Weise gespeichert, so ist es dem Planer möglich, die Erfahrung zu nutzen, die er beim Lösen eines Problems hinzugewonnen hat, und kann direkt einen Plan auswählen, der diese Probleme gelöst hat. Im Beispiel wird der Plan dadurch indiziert, daß er die Interaktion zwischen der Notwendigkeit einer trockenen Pfanne zum Kochen von knackigem Gemüse und dem Austreten von Bratensaft gelöst hat.

Bevor CHEF einen Plan sucht, der die gegebenen Ziele erfüllt, versucht er, Fehler vorherzusehen, die aus den Zielen resultieren. Auf diese Weise kann er schon an dieser Stelle einen Plan finden, der die auftretenden Fehler vermeidet. Wird CHEF etwa die Aufgabe gestellt, ein Rezept für Hähnchen mit Blumenkohl zu planen, so erkennt er anhand des *Anticipators*, daß er bei Verwendung des Plans BEEF-WITH-GREEN-BEANS am Zustand des Gemüses nach Ausführung des erzeugten Planes gescheitert ist und greift nicht auf diesen zu; des weiteren erkennt er, daß der Plan BEEF-WITH-BROCCOLI die hier oben aufgetretenen Fehler gelöst hat und wählt diesen zur Modifikation aus.

2.4 CAbPlan

Mit CAbPlan wird im Projekt X9 ein Planungssystem entwickelt, das aufgrund einer initialen Aufgabenstellung in Form des zu fertigenden Werkstücks aus einer Fallbasis einen geeigneten Fall auswählt, um geeignete Teilschritte zur Lösung der aktuellen Aufgabenstellung zu übernehmen. Weil ein Fall in der Regel nur einen Teil der Ziele des Problems löst, wurde die rein fallbasierte Planungskomponenete um eine generative Komponente erweitert.

Der Einsatz einer Fallbasis zur Steuerung eines Planungssystems erfordert bei der Fallauswahl weitreichende Analyseschritte, denen ein Modell der Problemklasse mit ihren Abhängigkeiten zugrunde liegen muß. Hierbei wird mit Hilfe von Wissen über die Geometrie des Werkstücks, die zu erfüllenden Qualitätsanforderungen (Oberflächengüten, Form- und Lagetoleranzen), die möglichen Spannflächen und die erforderlichen Bearbeitungsschritte ein mächtiges Hilfsmittel geschaffen, das eine Vorauswahl an Fällen trifft, die zur Problemlösung in Frage kommen. Auf diese Weise wird der Suchraum entscheidend eingeschränkt, was nötig ist, um ein effizientes Planen zu ermöglichen.

Dabei werden verschiedene Wissensquellen als voneinander unabhängige Spezialisten implementiert, wobei jeder dieser Spezialisten für einen bestimmten Aspekt der Problemlösung verantwortlich ist. Sie treten über einen nichtlinearen, hierarchischen Plan miteinander in Interaktion. Geplant sind ein Aufspannungsspezialist, der die Planung der Aufspannungen übernimmt (Anzahl der benötigten Aufspannungen, Spannflächen, Auswahl der Spannmittel), ein Geometriespezialist, der eine Abbildung der initialen Aufgabenbeschreibung in planungsrelevante Bereiche vornimmt, ein Technologiespezialist mit Wissen über Werkstoffe und phsikalische Größen sowie ein Zerspanspezialist. Der eigentliche Plan wird von einem klassischen Planungssystem verwaltet, das von den Spezialisten gesteuert wird. Es repräsentiert deren Kontrollentscheidungen expliziet (s. Abb. ??).

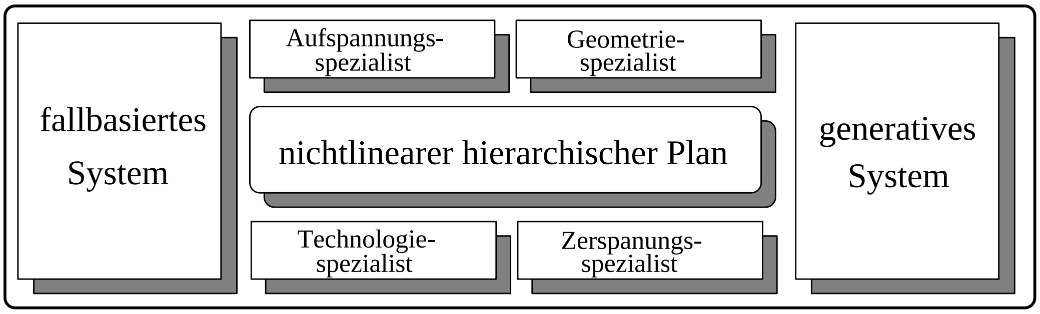


Abbildung 2.2: Archtitektur des CAbPlan–Systems

Dabei steuern die Spezialisten primär die generative Komponente des Planungssystems: sie bestimmen, welches Ziel als nächstes bearbeitet wird, durch welchen Operator dies erreicht wird, und sie erkennen Interaktionen zwischen Operatoren und Zuständen. Dabei wird die Menge der offenen Ziele vom Planungssystem nach jeder Entscheidung eines Spezialisten aktualisiert.

2.4.1 Dreiebenenarchitektur der generativen Komponente

Die meisten generativen Planungssysteme durchlaufen den Suchraum, um einen Plan zu erzeugen, der alle geforderten Ziele erfüllt. Kommt es dabei zu Konflikten durch Interaktionen, führen sie entweder ein Backtracking bis vor Auftreten des Konfliktes durch, oder sie wenden etwa *critics* [38] an, um zu einem konsistenten Zustand zu gelangen. Diese Methoden sind nur in rein generativen Systemen zur Neuplanung einsetzbar, sie unterstützen eine Modifikation bestehender Pläne nicht.

Diese zentrale Problematik kommt im Planmodifikationsproblem [11] zum Ausdruck; u.a. wird die Frage behandelt, wie ein Plan mit minimalen Änderungen an eine neue Problemstellung angepaßt werden kann. Gegen eine Verwendung von *critics* spricht, daß diese zu Sprüngen im Suchraum führen und eine Vollständigkeit des Lösungsraums nicht mehr gewährleisten; Abhilfe können hier *Truth-Maintenance-Systeme* (TMS) schaffen, welche die minimale Modifikation von Modellen im Falle von Inkonsistenz unterstützen.

In [20] wird für Planungsprobleme eine Struktur entwickelt, die es erlaubt, Gründe für die Optimalität und den Rückzug einer Entscheidung darzustellen und durch eine Erweiterung eines TMS zu verwalten. Es werden in diesem Modell weder positive noch negative Interaktionen berücksichtigt. Der zentrale Aspekt dieser Architektur liegt vielmehr in der Operatorauswahl- und Rücknahme durch *Dependency-Directed-Backtracking*. Da bei der Planmodifikation aber Optimierungsgesichtspunkte eine zentrale Rolle spielen, wurde das *Truth-Criterion* der Aktionsplanung [3] in diese Architektur aufgenommen [23], [34]. Hieraus resultiert eine Dreiebenenarchitektur des generativen Planungssystems:

- Ebene zur Behandlung der positiven und negativen Interaktion (*Truth-Criterion*)
  Neben Abhängigkeiten, die zu Inkonsistenzen führen und durch Backtracking behoben werden müssen, bestehen zwischen Operatoren positive und negative Interaktionen. Positive Interaktionen können zur Verbesserung des Plans ausgenutzt werden, wenn sie vom System erkannt werden. Negative Interaktionen können durch Einführen neuer Teilziele vermieden werden, wobei kein Backtracking nötig ist.

- Ebene zur Durchführung der Operatorauswahl und -rücknahme
  Ziele, die auf der ersten Ebene nicht erfüllt werden können, werden durch Anwendung von

Operatoren in Teilziele zerlegt. Die Rücknahme von Operatoren (d.h. die Planmodifikation) wird durch *Dependency-Directed-Backtracking* unterstützt (s. [19]).

- TMS–Ebene
  Die Abhängigkeiten, die bei der Zerlegung von Zielen in Teilziele oder bei der Behebung von Konflikten entstehen, werden in der TMS–Ebene verwaltet. Für jede Abhängigkeit auf der Planungsebene werden komplexe TMS–Teilnetze generiert und in das Abhängigkeitsnetzwerk eingefügt (s. [19]).

2.4.2 Hierarchische Planung

Sowohl bei der generativen Planung als auch bei der Fallauswahl kann eine Effizienzsteigerung erreicht werden, indem ein Problem und seine Lösung auf mehreren Abstraktionsstufen dargestellt wird. Hierbei werden Operatoren (und nicht die Zustände) abstrahiert, da sich in diesem Modell komplette Teilpläne durch abstrakte Operatoren beschreiben lassen. Die Untersuchung einer Problemstellung auf verschiedenen Abstaktionsstufen ist eine natürliche Vorgehensweise in vielen realen Domänen, so daß ein hierarchisches Modell direkt abgeleitet werden kann. Bei der hierarchischen Planung wird zunächst ein (vorläufiger) vollständiger Plan auf oberster Abstraktionsebene hergeleitet; danach werden die abstrakten Operatoren als eigenständige Teilprobleme der darunterliegenden Ebene(n) aufgefaßt. Falls für ein solches Teilproblem keine Lösung gefunden wird, wird auf der darüberliegenden Ebene ein alternativer nichtlinearer Plan bestimmt. werden mehrere Operatoren auf einer Ebene verschmolzen, so werden die resultierenden Teilpläne zusammengefaßt. Damit werden Interaktionen zwischen den Teilproblemen unmittelbar berücksichtigt, und Backtracking-Schritte auf darüberliegenden Ebenen können in vielen Fällen vermieden werden.

2.4.3 Die fallbasierte Planungskomponente

Im Zentrum des fallbasierten Ansatzes von CAbPlan steht nicht die konkrete Lösung eines Fallbeispiels, sondern der verfolgte Lösungsweg. Hierbei dient ein Fallbeispiel als Vorbild für eine Wiederholung des erfolgreichen Lösungswegs für die aktuelle Aufgabenstellung. Es übernimmt die Steuerung des Planungssystems, welche durch die TMS-basierende Systemarchitektur an beliebiger Stelle gestoppt und wieder aufgenommen werden kann.

Die Aufgabe der fallbasierten Komponente besteht nun in der Auswahl eines für die Lösung der aktuellen Problemstellung geeigneten Fallbeispiels; ein Fallbeispiel ist geeignet für ein Problem, wenn sich die gesuchte Lösung „wenig“ von der bekannten Lösung des Fallbeispiels unterscheidet. Somit ist der bei der Anpassung der gefundenen Lösung an die aktuelle Aufgabenstellung anfallende Modifikationsaufwand ein Maß für die Nützlichkeit eines Fallbeispiels. Da dieser Aufwand sich aber erst nach der Modifikationsphase des Planungssystems erfassen läßt, ist man bei der Auswahl eines geeigneten Fallbeispiels auf den Begriff der Ähnlichkeit zweier Werkstücke angewiesen; dieser Ansatz setzt voraus, daß ein geeignetes Ähnlichkeitsmaß den nötigen Modifikationsaufwand und damit eine Nützlichkeit des Fallbeispiels impliziert.

In der Kognitionspsychologie wird beim Ähnlichkeitsbegriff zwischen zwei Arten unterschieden:

- oberflächliche Ähnlichkeit (*surface similarity*)
- strukturelle Ähnlichkeit (*deep similarity*)

Diese Zweiteilung ist auch für wissensbasierte Systeme interessant: eine oberflächliche Überprüfung anhand von Attributen kann mit einfachen Methoden in verhältnismäßig kurzer Zeit durchgeführt werden, während eine strukturelle Ähnlichkeit oft aufwendige Verfahren nötig macht (z.B. Mapping–Algorithmen). Es wird bei der Ähnlichkeitsbestimmung in CAbPlan daher in einer ersten Stufe eine Menge von Fallbeispielen auf der Basis eines syntaktischen Ähnlichkeitsmaßes ausgewählt, die dann in einer zweiten Stufe einer strukturorientierten Prüfung unterzogen werden.

2.5 Ähnlichkeitsmaße

Der Begriff der Ähnlichkeit zweier Objekte nimmt eine zentrale Rolle ein bei der Untersuchung fallbasierter Methoden. Ohne ein geeignetes Ähnlichkeitsmaß kann ein Retriever nicht entscheiden, welches Fallbeispiel dem aktuellen Problem am ehesten entspricht. Prinzipiell läßt sich beim fallbasierten Schließen ein geeignetes Ähnlichkeitsmaß auf der Basis des Modifikationsaufwandes bei der Anpassung der alten Lösung an das Problem definieren. Es muß aber oftmals in Kauf genommen werden, daß zur Ähnlichkeitsfindung eher auf ad–hoc-Methoden zurückgegriffen wird, die sich aus der Aufgabenstellung ableiten lassen, weil sich ein ähnlichster Fall auf der Basis des Modifikationsaufwandes erst a posteriori korrekt ermitteln läßt, die Ähnlichkeitsbestimmung aber vorher stattfinden muß [37].

2.5.1 Definition der Ähnlichkeit

Die *Ähnlichkeit* zweier Objekte ist zunächst ein subjektiver Begriff. Ein Auto und ein Motorrad sind sich sehr ähnlich, wenn man als Maß nimmt, daß beides Fortbewegungsmittel auf Basis eines Verbrennungsmotors sind. Liegt das Interesse jedoch mehr am Fahrverhalten, so zeigen sich jedoch erhebliche Unterschiede. Um den Begriff *Ähnlichkeit* mathematisch in den Griff zu bekommen, benötigt man ein geeignetes Ähnlichkeitsmaß.

Um eine Einschränkung auf bestimmte Aspekte zu sichern, muß eine Abstraktion der zu vergleichenden Objekte durchgeführt werden. Die Betrachtung der interessierenden Objektklasse beschränkt sich also auf eine gewisse Anzahl von Parametern, wobei die exakten quantitativen Werte meist in qualitative umgewandelt werden, weil eine genaue numerische Übereinstimmung in den wenigsten Fällen gegeben, zur Problemlösung aber auch nicht notwendig ist.

Eine Formalisierung des Ähnlichkeitsbegriffs wird in [22] gegeben. Es werden hier Funktionen angegeben für die Beziehungen

- „x und y sind ähnlich“
- „x und y sind unähnlich“
- „x und y sind ähnlicher als x und z“

In den ersten beiden Fällen wird keine Relation definiert, weil nicht gesagt werden kann, daß eine Ähnlichkeit entweder besteht oder nicht besteht, sondern eine reellwertige Funktion, die ein Maß für die Ähnlichkeit von x und y darstellt; es wird also ausgedrückt, daß x und y „mehr oder weniger“ ähnlich zueinander sind.

Von der Ähnlichkeit wird Reflexivität und Symmetrie gefordert, nicht aber Transitivität; über längere Ketten hinweg kann der Grad der Ähnlichkeit abnehmen. Dies führt zur Einführung der Ähnlichkeitsfunktion μ:

Definition: Ein *Ähnlichkeitsmaß* auf einer Menge M ist eine reellwertige Funktion $\mu : M^2 \rightarrow [0,1]$ mit

$$\mu(x,x) = 1 \tag{2.1}$$
$$\mu(x,y) = \mu(y,x) \tag{2.2}$$

Um von der „Unähnlichkeit“ zwischen zwei Objekten sprechen zu können, wird eine Distanzfunktion d eingeführt:

Definition: Eine *Distanzfunktion* auf einer Menge M ist eine reellwertige Funktion $d : M^2 \rightarrow \Re$ mit

$$d(x,y) \geq 0 \tag{2.3}$$
$$d(x,x) = 0 \tag{2.4}$$
$$d(x,y) = d(y,x) \tag{2.5}$$

Im allgemeinen wird für $x \neq y$ nicht verlangt, daß $d(x,y) > 0$ ist, weil man oft nur auf wenige Merkmale der zu betrachtenden Objekte zugreifen kann, die zur einer vollständigen Unterscheidung nicht ausreichen; man erhält also keine Metrik.

Normalerweise wird zusätzlich noch die Dreiecksungleichung $d(x,y) + d(y,z) \geq d(x,z)$ gefordert, womit die Distanzfunktion zu einer sog. Quasimetrik wird.

Im dritten Fall, bei der Definition von „x ist ähnlicher zu y als x zu z", ergibt sich eine dreistellige Relation $R(x,y,z)$. Zunächst wird eine vierstellige Relation $\ddot{A}(x,y,u,v)$ definiert („x ist ähnlicher zu y als u zu v"):

$$\ddot{A}(x,x,u,v) \tag{2.6}$$
$$\ddot{A}(x,y,u,v) \leftrightarrow \ddot{A}(y,x,u,v) \leftrightarrow \ddot{A}(x,y,v,u) \tag{2.7}$$

Damit ergibt sich die gewünschte Relation R zu:

$$R(x,y,z) \quad :\leftrightarrow \quad \ddot{A}(x,y,x,z). \tag{2.8}$$

Für das analoge Schließen benötigt man schließlich eine Relation $\hat{R}$. Mit ihr kann man zu einem gegebenen x ein „ähnlichstes" y aus einer Menge M finden. Es gilt:

$$\hat{R}(x,y) \quad :\leftrightarrow \quad \forall z \in M : R(x,y,z) \tag{2.9}$$

Prinzipiell reicht es aus, numerische Werte von Ähnlichkeitsbeziehungen zu beachten; es ist in der Praxis aber von Vorteil, über Ähnlichkeitsmaß μ und Distanzfunktion d Relationen für das Ähnlichkeitsmaß zu erzeugen:

$$\ddot{A}_\mu(x,y,u,v) \quad \leftrightarrow \quad \mu(x,y) \geq \mu(u,v) \tag{2.10}$$
$$\ddot{A}_d(x,y,u,v) \quad \leftrightarrow \quad d(x,y) \leq d(u,v) \tag{2.11}$$

2.5.2 Matching von Baumstrukturen

Der Ursprung der Matching–Algorithmen liegt im Matchen zweier Zeichenketten (Wörter). Es wird eine Distanzfunktion definiert über die Editierkosten, die verursacht werden, um eine Zeichenkette in eine andere zu überführen. Hierzu werden drei Editieroperationen zugelassen:

- Ersetzen eines Zeichens durch ein anderes
- Löschen eines Zeichens
- Einfügen eines Zeichens

Die zugehörigen Algorithmen berechnen die Distanzfunktion mit Komplexität $O(m * n)$, wobei m und n die Längen der beiden Wörter sind.

Beispiel: (ohne nähere Erläuterungen)

Seien $x = ababcb$, $y = aabcbcc$. Die folgende Sequenz von Editieroperationen transformiert x nach y:

Lösche ein b;	Ergebnis: $aabcb$
Füge ein c ein;	Ergebnis: $aabcbc$
Füge ein c ein;	Ergebnis: $aabcbcc = y$

Verschiedene Ansätze versuchen, diese Methode auf mehrdimensionale Strukturen wie Bäume oder Graphen zu übertragen. Dabei sind Baumstrukturen wohl die meistbenutzten nichtlinearen Strukturen, die in Computerprogrammen auftreten.

Es soll nun ein Abstandsmaß für geordnete Bäume mit benannten Knoten (*labeled ordered trees*) definiert werden. Erlaubte Operationen auf Knoten sind hierbei:

- Einen Knoten durch einen anderen ersetzen
- Einen Knoten aus dem Baum löschen
- Einen Knoten in den Baum einfügen

Definitionen

Alle betrachteten Bäume haben eine Wurzel, sind geordnet und ihre Knoten sind benannt. Sei T ein Baum. $|T|$ gibt die Anzahl der Knoten von T an; $t[i]$ bezeichnet den Knoten an der i-ten Stelle im Baum T, wobei die Numerierung in Vorordnung erfolgt (d.h. Durchnumerierung in der Reihenfolge Wurzel – linker Teilbaum – rechter Teilbaum). In Abb. ??.a gilt: $t[2] = B$, $t[5] = D$. $T[i]$ ist ein Teilbaum von T mit der Wurzel $t[i]$. Es gilt: $T[1] = T$. Man bezeichnet das Blatt mit der größten Nummer in $T[i]$ als Endblatt (*end leaf*) von $T[i]$ und symbolisiert es mit $el(i)$. Im Beispiel ist das Endblatt von $T[4]$ gleich 7, es gilt also $el(4) = 7$. $Fa(i)$ ist der Vaterknoten von $t[i]$, und $Fa^2(i) = Fa(Fa(i))$. Die Menge der Vorgänger von $t[i]$ wird mit $An(i) = \bigcup_k Fa^k(i)$, der Pfad von $t[i]$ nach $t[j]$ wird mit $tr(i,j)$ bezeichnet; hierbei muß $t[i]$ ein Vorgänger von $t[j]$ sein.

V sei eine Menge von Knoten, E eine Menge von Kanten. Dann kann man den Baum T auch als Tupel (V, E) darstellen, entsprechend einen Teilbaum $T[i]$ von T als Tupel $(V[i], E[i])$.

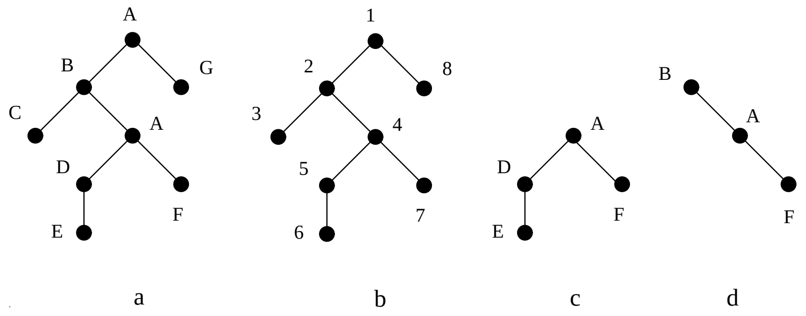


Abbildung 2.3: a Baum T, b Numerierung in Vorordnung, c Teilbaum $T[4]$, d Pfad $tr(2,7)$

Λ bezeichnet den leeren Knoten. Eine Editieroperation wird dargestellt als $b \to c$, wobei b und c entweder Knoten oder gleich Λ sind. $b \to c$ ist eine Ersetzungsoperation, falls $b \neq \Lambda$ und $c \neq \Lambda$, eine Löschoperation, falls $b \neq \Lambda = c$, und eine Einfügeoperation, falls $b = \Lambda \neq c$.

$S = s_1, s_2, ..., s_m$ sei eine Folge von Editieroperationen. S transformiert Baum T_α nach T_β, wenn es eine Folge von Bäumen $T_0, T_1, ..., T_m$ gibt mit $T_\alpha = T_0$, $T_\beta = T_m$ und $T_{i-1} \Rightarrow T_i$ durch s_i f.a. $1 \leq i \leq m$.

Sei r eine Kostenfunktion, die jeder Editieroperation $b \to c$ eine nichtnegative reelle Zahl $r(b \to c)$ zuordnet. Dann kann man r ausweiten auf eine Folge S von Editieroperationen:

$$r(S) = \sum_{i=1}^{m} r(s_i) \tag{2.12}$$

Es gilt:

1. $r(b \to b) = 0$
2. $r(b \to a) + r(a \to c) \geq r(b \to c)$

Die Distanzfunktion $d(T_\alpha, T_\beta)$ zwischen den Bäumen T_α und T_β wird definiert als das Minimum der Kosten aller möglichen Folgen von Editieroperationen, die T_α nach T_β transformieren:

$$\begin{aligned} d(T_\alpha, T_\beta) \quad = \quad & min\{r(S)|S \text{ ist eine Folge von Editieroperationen,} \\ & \text{die } T_\alpha \text{ nach } T_\beta \text{ transformieren}\} \end{aligned} \tag{2.13}$$

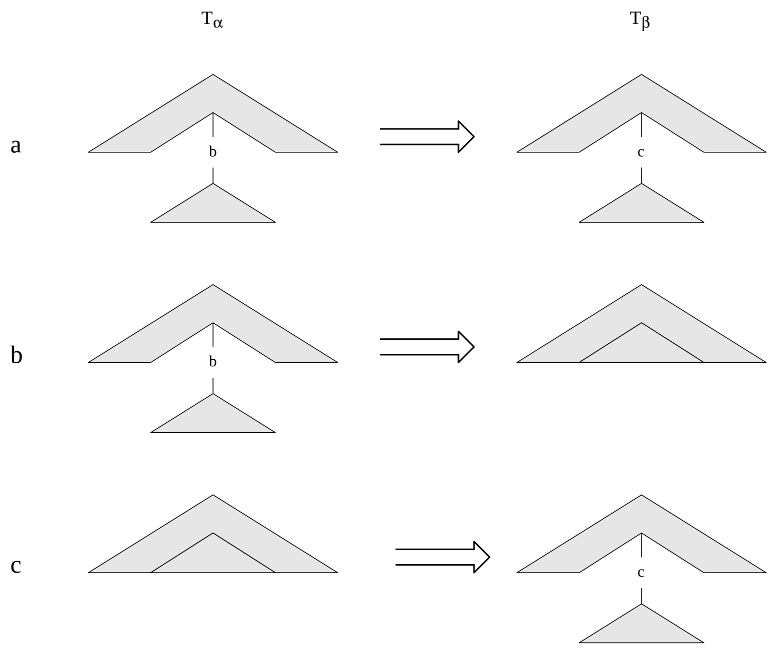


Abbildung 2.4: Editieroperationen auf Bäumen. a $b \rightarrow c$, b $b \rightarrow \Lambda$, c $\Lambda \rightarrow c$

Weil die Anzahl verschiedener Folgen von Editieroperationen, die T_α nach T_β transformieren, unendlich groß ist, kann man nicht alle möglichen Folgen berechnen und hieraus das Minimum ermitteln; aus diesem Grund werden Strukturen eingeführt, die $d(T_\alpha, T_\beta)$ in polynominaler Zeit berechnen: die sog. *Mappings*.

Tai Mapping

Unter Mapping versteht man einfach ausgedrückt eine Beschreibung von Editieroperationen, die einen Baum T_α in einen Baum T_β überführen, wobei die Reihenfolge dieser Operationen keine Rolle spielt.

Eine gestrichelte Linie von $t_\alpha[i]$ nach $t_\beta[j]$ in T_α bzw. T_β besagt, daß $t_\alpha[i]$ durch $t_\beta[j]$ ersetzt wird, wenn $t_\alpha[i] \neq t_\beta[j]$, und unverändert bleibt, wenn $t_\alpha[i] = t_\beta[j]$. Knoten in T_α, die von keiner gestrichelten Linie berührt werden, werden gelöscht, und Knoten in T_β, die nicht berührt werden, werden eingefügt. Ein solches Diagramm wird *Mapping* genannt, das eine Möglichkeit darstellt, Baum T_α in Baum T_β zu transformieren.

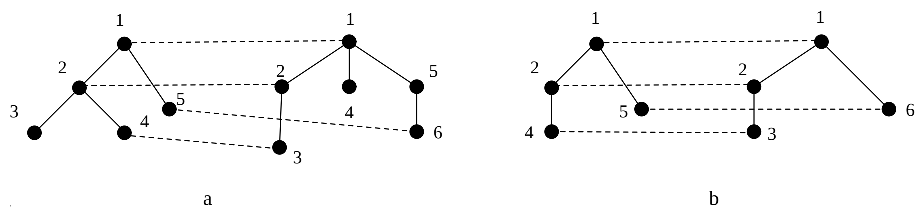


Abbildung 2.5: a Mapping zweier Bäume, b gleiche Strukturen nach Löschen der freien Knoten

Wir definieren zwei verschiedene Mappings von Baum T_α nach Baum T_β. Das erste wurde von K. C. Tai [28] entwickelt und nach ihm Tai Mapping genannt. Seien T_α und T_β zwei Bäume. Dann ist M_T eine Menge von Zahlenpaaren (i, j) mit $i \in V_\alpha$ und $j \in V_\beta$.

Definition: M_T heißt *Tai Mapping*, wenn

1. $1 \leq i \leq |T_\alpha|$ und $1 \leq j \leq |T_\beta|$ für (i, j) in M_T
2. Für alle Paare (i_1, j_1) und (i_2, j_2) in M_T gilt:
   (a) $i_1 = i_2 \Longleftrightarrow j_1 = j_2$
   (b) $i_1 < i_2 \Longleftrightarrow j_1 < j_2$
   (c) $t_\alpha[i_1]$ ist Vorgänger von $t_\alpha[i_2] \Longleftrightarrow t_\beta[j_1]$ ist Vorgänger von $t_\beta[j_2]$

Jedes Paar (i, j) in M_T wird als Linie interpretiert, die $t_\alpha[i]$ mit $t_\beta[j]$ verbindet. Bedingung (2.a) bedeutet, daß jeder Knoten in T_α und T_β von höchstens einer Linie berührt wird. Die Bedingungen (2.b) und (2.c) besagen, daß die Nachfolgerbeziehungen zwischen Knoten erhalten bleiben, und daß die bijektive Abbildung zwischen den Knoten der beiden Bäume genau durch die Elemente von M_T dargestellt wird.

Sei M_T ein Tai-Mapping von T_α nach T_β und seien I und J die Mengen der Knoten aus T_α bzw. T_β, die von keiner Linie berührt werden. Dann definieren wir die Kosten *cost* von M_T:

Definition:

$$cost(M_T) = \sum_{(i,j) \in M} r(T_\alpha[i] \to T_\beta[j]) + \sum_{i \in I} r(T_\alpha[i] \to \Lambda) + \sum_{j \in J} (\Lambda \to T_\beta[j]) \tag{2.14}$$

Dabei bezeichnet $T_\alpha[i] \to \Lambda$ eine Löschoperation in T_α, und $\Lambda \to T_\beta[j]$ eine Einfügeoperation in T_β.

Sei d_T definiert nach:

$$\begin{aligned} d_T(T_\alpha, T_\beta) \quad &= \quad min\{cost(S) | S \text{ ist eine Folge von Editieroperationen,} \\ &\qquad \text{die } T_\alpha \text{ nach } T_\beta \text{ transformieren}\} \end{aligned} \tag{2.15}$$

Dann gilt:

$$d_T(T_\alpha, T_\beta) \quad = \quad min\{cost(M_T) | M_T \text{ ist Tai-Mapping von } T_\alpha \text{ nach } T_\beta\} \tag{2.16}$$

Seien L_α und L_β die maximalen Tiefen von T_α bzw. T_β. Dann läßt sich $d_T(T_\alpha, T_\beta)$ mit der Komplexität $O(|T_\alpha| * |T_\beta| * L_\alpha^2 * L_\beta^2)$ berechnen.

Structure Preserving Mapping

Neben dem Tai-Mapping wurde von E. und K. Tanaka [27] ein weiterer Ansatz zum Mappen von Bäumen verfolgt: das *structure preserving mapping* M_S, das nicht in allen Fällen dem Tai-Mapping entspricht. Der Hauptunterschied zum Tai-Mapping besteht darin, daß versucht wird, Teilbäume in T_α auf strukturgleiche Teilbäume in T_β abzubilden.

Es seien $\psi_\alpha(a)$ und $\psi_\beta(b)$ wie folgt definiert:

$$\psi_\alpha(a) \quad = \quad \{i | (i, j) \in M_S, i \in V_\alpha[a])\} \tag{2.17}$$

$$\psi_\beta(b) \quad = \quad \{j | (i, j) \in M_S, j \in V_\beta[b])\} \tag{2.18}$$

mit $T_\alpha[a] = (V_\alpha[a], E_\alpha[a])$ und $T_\beta[b] = (V_\beta[b], E_\beta[b])$.

Es seien $\psi_\alpha(a) = \{e, f, ..., h\}$ und $(e, e'), (f, f'), ..., (h, h') \in M_S$. Außerdem sei $X(a) = \{e', f', ..., h'\}$.

Definition:

$$\hat{A}n(X(a)) \quad = \quad \cap\{An(x) \cup \{x\}\} \text{ f.a. } x \in X(a) \tag{2.19}$$

$$r(a) \quad = \quad max\{\hat{A}n(X(a))\} \tag{2.20}$$

$T_\beta[r(a)]$ ist der kleinste Teilbaum, der $X(a)$ enthält. Für gegebene M_S, T_α und T_β können wir den eindeutig bestimmten Teilbaum $T_\beta[r(a)]$ für $\psi_\alpha(a) \neq \emptyset$ als Bild von $T_\alpha[a]$ bestimmen.

Beispiel: T_α und T_β in Beispiel ??.

Betrachte $M_S = \{(5,7'),(7,9')\}$. Dann ist $X(4) = \{7',9'\}$ und $r(4) = 4'$. Sowohl $T_\beta[1'], T_\beta[2']$ als auch $T_\beta[4']$ enthalten $X(4)$. $T_\beta[4']$ ist der kleinste Teilbaum, der $X(4)$ enthält.

Definition: M_S heißt *Structure Preserving Mapping*, wenn

1. $1 \leq i \leq |T_\alpha|$ und $1 \leq j \leq |T_\beta|$ für (i,j) in M_S
2. Für alle Paare (i_1, j_1) und (i_2, j_2) in M_S gilt:
   (a) $i_1 = i_2 \Longleftrightarrow j_1 = j_2$
   (b) $i_1 < i_2 \Longleftrightarrow j_1 < j_2$
   (c) Wenn $r(i_1)$ und $r(i_2)$ für i_1 und i_2 in V_α existieren, gilt:
   $el(i_1) < i_2 \Longleftrightarrow el(r(i_1)) < r(i_2)$

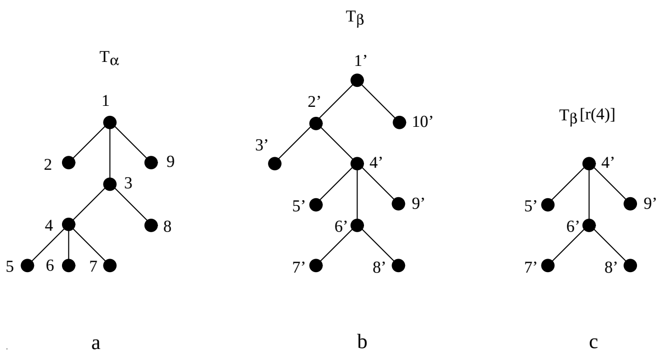


Abbildung 2.6: a Baum T_α, b Baum T_β, c Teilbaum $T_\beta[r(a)]$

Beim *Structure Preserving Mapping* werden seperate Teilbäume $T_\alpha[a]$ und $T_\alpha[b]$ auf separate Teilbäume $T_\beta[r(a)]$ und $T_\beta[r(b)]$ gemappt (Abb. ??.a). Deshalb heißt dieses Mapping „Strukturerhaltendes Mapping". Demgegenüber können beim Tai-Mapping getrennte Teilbäume $T_\alpha[a]$ und $T_\alpha[b]$ auf $F_\beta[r(a)]$ und $T_\beta[r(b)]$ gemappt werden, wobei $F_\beta[r(a)]$ kein Baum ist (Abb. ??.b).

2.6 Expertensysteme in CIM

Nach der Einführung in fallbasierte Planungstechniken und der formalen Definition von Ähnlichkeit sowie der Vorstellung von zwei Mapping–Verfahren als Anwendungsbeispiel zur Ähnlichkeitsbestimmung von Baumstrukturen wird im nachfolgenden Abschnitt ein Überblick gegeben über die Einsatzmöglichkeiten von Expertensystemen im Bereich von CIM, insbesondere auch in Hinblick auf den Nutzen fallbasierter Systeme.

Flexible Fertigungsautomatisierung und bedarfsgerechte Produktion wurden erst durch die Rechnertechnik und durch Informationssysteme ermöglicht. CIM[7] als Unternehmensstrategie zielt auf eine funktionale Vernetzung und systemtechnische Kopplung aller produktionstechnischen Prozesse und Bereiche. In durchgängigen Prozeßketten werden rechnergestützte Systeminseln, z.B. Konstruktion, Prozeßplanung und Fertigung, so miteinander verknüpft und vernetzt, daß einmal erzeugte Daten jederzeit und an beliebigem Ort verfügbar sind.

[7] Computer Integrated Manufacturing = rechnergeführte und -integrierte Produktion

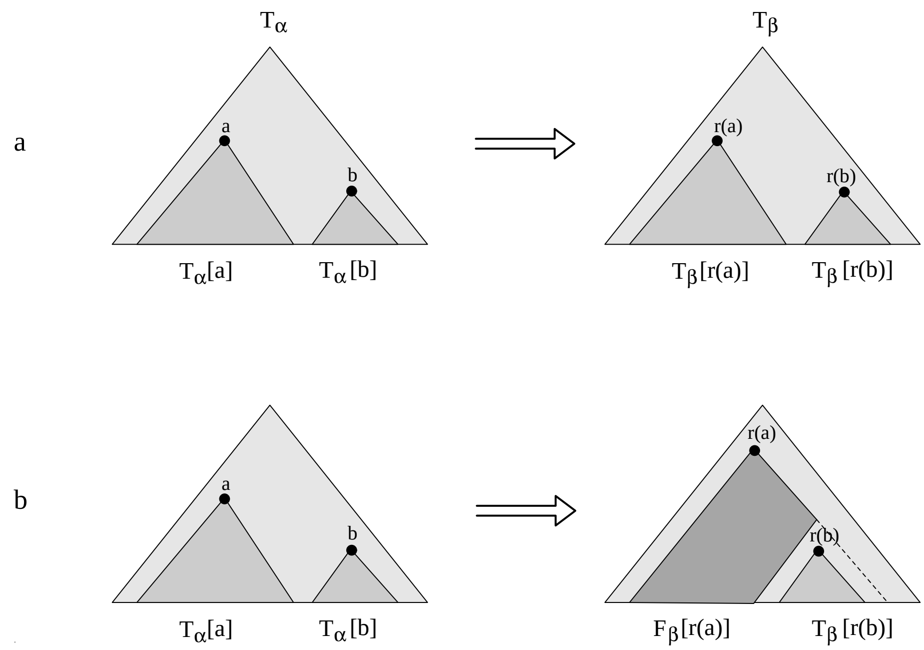


Abbildung 2.7: a Structure Preserving Mapping, b Tai-Mapping

CIM als Drei–Säulen–Modell umfaßt den betrieblich-organisatorischen Bereich des Auftragswesens, den technisch-funktionalen Bereich der Produktentstehung sowie den strategisch-interagierenden Bereich der Qualitätssicherung, die durch gemeinsame Datenbasen in einem Netz von Hard- und Softwaresystemen sowie durch gemeinsame Kommunikationsstrukturen miteinander verbunden sind (Abb. ??).

Die industrielle Produktionstechnik umfaßt die Verknüpfung von Materialprozessen, Organisationsprozessen und Informationsprozessen, deren rechnergestützte Integration neue Konzepte erfordert, so daß durch Prozeßkopplung zwischen den Bereiche Technologieplanung, Logistik sowie Produktionsplanung und Werkstattsteuerung CIM mit unterschiedlichen Integrationsgraden erreicht werden kann (Abb. ??).

Produktionsprozesse verhalten sich probabilistisch und werden stochastisch beeinflußt. Sie sind dynamischen Schwankungen unterworfen und ihre Daten sind in hohem Maße heterogen. Damit lassen sich einzelne Prozesse, Prozeßketten und Prozeßstrukturen mit der „konventionellen" Datenverarbeitung nur eingeschränkt koppeln. In Teilbereichen (Automatisierung von Abläufen mit prozeßtechnischer Koppelung) wurden Lösungen erreicht, wie die Beispiele CAD[8]/CAP[9]/CAM[10] oder adaptive Regelkonzepte für Fertigungsprozesse (AC) immer wieder belegen.

Informationsprozesse in der Produktionstechnik sind mehr als nur datentechnische Operationen mit mehr oder weniger großen Datenbeständen. Sie erfordern intelligente Fähigkeiten wie Abstraktion, Interpretation, Schlußfolgern und Entscheiden in Verbindung mit umfangreichem Expertenwissen aus verschiedenartigen Fachgebieten. Die Wissensverarbeitung in Form von Expertensystemen kann als Programmierung auf höherem Abstraktionsniveau bezeichnet werden und unterstützt so „intelligente" Prozesse, wie sie bei der Konstruktion und Planung oder bei der Überwachung und Diagnose komplexer Fertigungssysteme und Produktionsanlagen gefordert sind.

---

[8] Computer Aided Design
[9] Computer Aided Planning
[10] Computer Aided Manifacturing

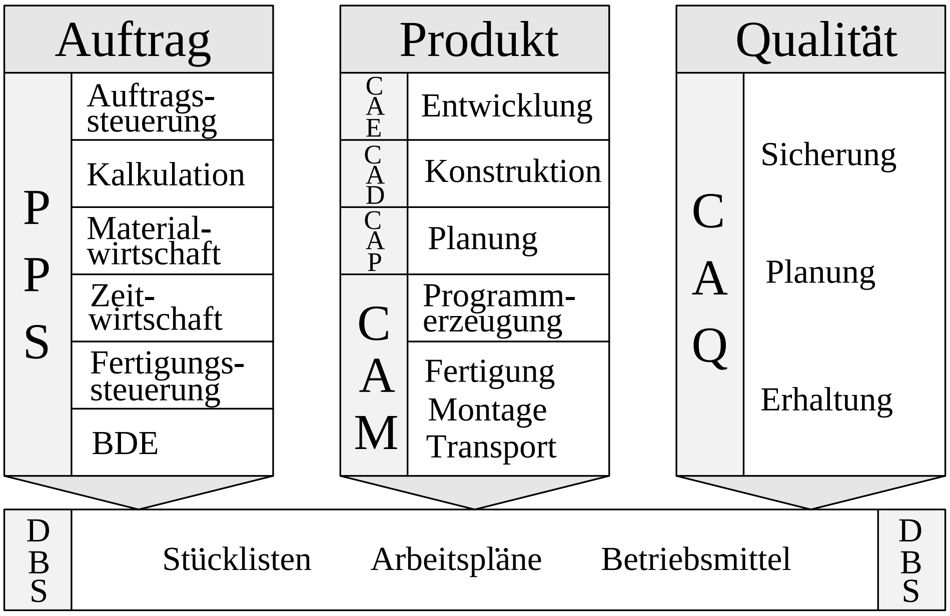


Abbildung 2.8: CIM als Drei–Säulen–Modell

2.6.1 Expertensysteme in der Produktionstechnik

Einsatzmöglichkeiten von Expertensystemen in CIM

Konventionelle Datenverarbeitung und die damit verbundene enge Verknüpfung von Daten und Programmablauf eignen sich für Problemstellungen, die algorithmisch gelöst werden können und bei denen die zu verarbeiteten Daten in numerischer Form vorliegen. Demgegenüber grenzt sich ein integriertes Produktionsdatensystem durch komplexe, symbolische Zusammenhänge, diffuses Wissen und Heuristiken ab, deren Verarbeitung mit Hilfe wissensbasierter Systeme erleichtert wird. Ihre Struktur sieht eine klare Trennung zwischen Daten und Ablaufstruktur vor.

Es ergeben sich Einsatzmöglichkeiten für Expertensysteme als firmenspezifische Kopplungsbausteine zwischen vorhandenen Insellösungen (z.B. CAD/CAM), aber auch ganz allgemein als Hilfsmittel zur Unterstützung des Menschen bei der Lösung von:

- Planungsproblemen
  - Reihenfolgeplanungen
  - Konfigurationsaufgaben (Konstruktion)
  - Prozeßgestaltung
  - Auswahl von
    - Fertigungsverfahren
    - Fertigungsmitteln
    - Bearbeitungsbedingungen
- Diagnoseaufgaben
  - Überwachung von Prozessen und Maschinen
  - Meßdatenanalyse
  - Qualitätssicherung

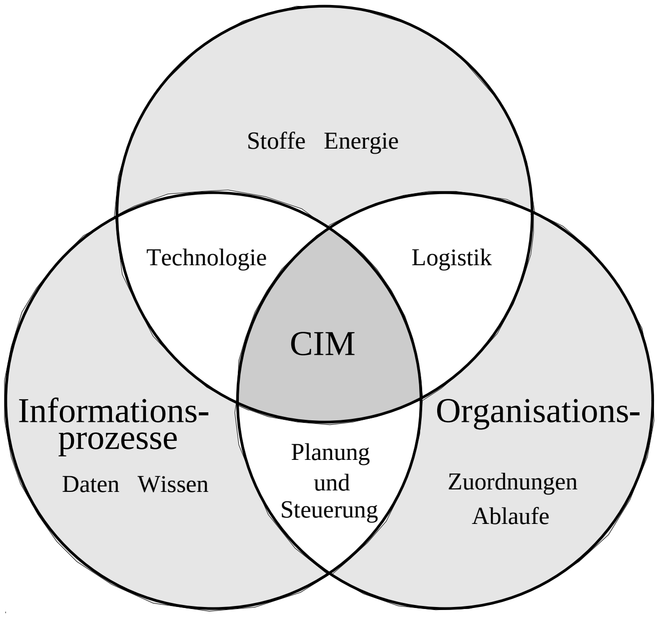


Abbildung 2.9: Industrielle Produktionstechnik

- Schulung
  - Wissenssicherung
  - Wissensvermittlung

2.6.2 Kriterien für den Einsatz von Expertensystemen

Allgemein basieren die Ansätze zur Unterscheidung von konventionellen und wissensbasierten Programmiermethoden auf den Kriterien der Ablauf- und Kontrollmechanismen. Bei der konventionellen Programmentwicklung bestimmt der Systementwickler den Programmablauf bereits bei der Entwicklung eines Ablaufdiagramms und setzt dieses Diagramm dann in Programmcode um. Das setzt voraus, daß jede Situation bereits bei der Entwicklung bekannt ist und berücksichtigt wird. Als Folge dieser Technik werden sowohl die eigentliche Problemlösung als auch die Ablaufsteuerung und die zu verarbeitenden Daten untrennbar in einem Algorithmus miteinander vermischt.

Bei der Entwicklung von Expertensystemen legt der Entwickler nur fest, welche Aktionen in bestimmten Situationen ausgeführt werden sollen, ohne aber Einfluß auf die Reihenfolge der Abarbeitung der Regeln zu nehmen. Die Ablaufsteuerung des Expertensystems entscheidet autonom und situationsabhängig über den jeweils einzuschlagenden Weg. Durch Festlegung von Wertigkeiten und Meta-Regeln kann die Ablaufsteuerung beeinflußt werden.

Konventionelle Datenverarbeitung und wissensbasierte Datenverarbeitung lassen sich nach inhaltlichen Kriterien unterscheiden:

Datenverarbeitung	Wissensverarbeitung
Automatisierung meist monotoner, klar strukturierter und wohldefinierter Informationsverarbeitungsprozesse	Automatisierung von hochkomplexen Informationsstrukturen inkl. dem Umgang mit diffusem Wissen
hauptsächlich Verarbeitung homogener Massendaten	hauptsächlich Verarbeitung heterogen strukturierter Wissenstrukturen
Komplexität entsteht hauptsächlich durch den Umfang der Datenmenge	Komplexität entsteht durch Vielfalt und Umfang der Wissensstrukturen
bei formaler Ein-/Ausgabespezifikation ist prinzipiell die Möglichkeit eines Korrektheitsbeweises gegeben	Verarbeitung erfolgt durch Heuristiken und diffuses Wissen, daher sind Korrektheitsbeweise in aller Regel nicht möglich

Ebenso ist eine Unterscheidung nach formalen Kriterien möglich:

Datenverarbeitung	Wissensverarbeitung
Verarbeitung von numerischen Daten	Verarbeitung von symbolischen Ausdrücken
wenige Datentypen, aber viele Instanzen eines Typs	viele beliebig komplexe Datenstrukturen, häufig wenige Instanzen eines Typs
typische höhere Programmiersprachen sind FORTRAN, PASCAL, C, COBOL	typische höhere Programmiersprachen sind LISP, PROLOG, Smalltalk
konventionelles, strukturiertes Programmieren	oft objektorientiertes Programmieren
Verarbeitungsablauf ist explizit festgelegt	Verarbeitungsablauf ist nur implizit oder gar nicht vorgegeben
unvollständige Eingaben werden zurückgewiesen	Verarbeitung unvollständiger Strukturen möglich
Modifikationen sind selten	häufige Modifikationen des Systems sind üblich

Hieraus lassen sich einige Problembereiche als für den Einsatz von Wissensverarbeitung ungeeignet herauskristallieren:

- Problembereiche mit vorwiegend arithmetischen Berechnungen
  z.B. Finite Elemente Methode

- Problembereiche mit hohen Realzeitanforderungen
  z.B. Überwachung hochfrequenter Prozesse

- Problembereiche, deren Lösungsmethodik eindeutig ist und nicht modifiziert wird
  z.B. statistische Methoden

- Problembereiche mit vielen homogenen Massendaten
  z.B. statische Lagerbestandsverwaltung

Für eine schlüssige Beurteilung von Anwendungsfeldern reichen diese Hinweise jedoch nicht aus; im folgenden werden zwei Kriterienkataloge aufgelistet, die in Form einer Checkliste als Entscheidungsgrundlage dienen können.

- positive Kriterien
  - Teilbereiche der Problemstellung unterliegen dynamischen Änderungen
    z.B. Motordiagnose
  - Die zur Problemlösung notwendigen Informationen sind z.T. unvollständig oder inkonsistent
    z.B. Fehleranfälligkeit einer Prüfeinrichtung zur Diagnose von Schaltplatten
  - Die zur Problemlösung notwendigen Informationen sind nicht nur objektiver Natur, sondern enthalten auch subjektive Eindrücke
    z.B. Anlagendiagnose
  - Die Problemstellung ist so komplex, daß eine vollständige Erfassung und Programmierung aller Lösungswege zu aufwendig wird
    z.B. Maschinendiagnose, Arbeitsplanung
  - Das Wissensgebiet ist nicht klar abgrenzbar und unterliegt Einflüssen verschiedener Wissensgebiete
    z.B. CNC-Maschinendiagnose, Anlagen mit Prozeßüberwachung
  - Lösungsvorschläge und Lösungsstrategien müssen vom System erklärt werden
    z.B. Kraftwerksüberwachung, medizinische Diagnose
  - Das System soll nach der Fertigstellung auch bedingt zu Schulungszwecken eingesetzt werden
    z.B. Instandhaltungs-, Serviceabteilungen
  - Es gibt mehrere zulässige, unter Umständen sehr ähnliche Lösungen mit sehr ähnlichen Symptombildern, die, möglichst gewichtet, alle als potentielle Lösungen vorgeschlagen werden sollen
    z.B. Maschinen-, Anlagendiagnose
  - Aufgrund der Problemstruktur ergeben sich neue Daten und Randbedingungen
    z.B. medizinische Diagnosen, Überwachung chemischer Prozesse
  - Der Problemfall ist dadurch gekennzeichnet, daß mehrere generierte Alternativen kontextabhängig bewertet werden müssen, ohne daß eindeutige mathematische Bewertungskriterien zur Verfügung stehen
    z.B. Arbeitsplanung
  - Die möglichen Lösungen sind bei der Programmerstellung nicht vollständig bekannt
    z.B. Arbeitsplanung
  - Die Erfolgsaussichten einer eingeschlagenen Lösung sind zu Beginn nur unter Unsicherheit einzuschätzen
    z.B. komplexe Diagnose, aufwendige Planungsprobleme
  - Die Problemlösung kann anhand einer Vielzahl von Beispielen erklärt werden, die bereits einen Grundstock der Lösungssystematik bilden. Diese Beispiele können einen Teil der Wissensbasis darstellen
    z.B. Diagnose, Arbeitsplanung
  - Die Problemstruktur erfordert einen interaktiven Dialog zwischen System und Mensch, bei dem der zur Lösung notwendige Informationsvorrat nicht immer vollständig zu Beginn einer Sitzung erfragt werden kann, so daß das System bei fehlenden Fakten automatisch Fragen an den Anwender generiert
    z.B. Maschinendiagnose

  - Mehrere Experten verfügen über ein ausgewiesenes Fachwissen jeweils eines Teilbereichs der Problemstellung
    z.B. Konstruktion, Arbeitsplanung
  - Wird die Entwicklung eines Prototyps gefordert, bieten Expertensysteme die Möglichkeit, den Prototyp iterativ weiterzuentwickeln
    z.B. alle komplexen Problemstellungen, deren Lösbarkeit nachgewiesen werden muß
  - Das Wissen um die Lösung eines Problembereichs wächst kontinuierlich und unterliegt zumindest teilweise dynamischen Änderungen
    z.B. Diagnosesystem
  - Das Wissen um die Lösung einer Problemstellung soll gesichert, dupliziert oder weiter verfügbar gemacht werden
    z.B. Serviceabteilung mit intelligenten Diagnosehilfen

- k.o.-Kriterien
  - Die Problemstellung unterliegt sehr hohen Echtzeitanforderungen
    z.B. Werkzeugbruchüberwachung
  - Die Komplexität der Problemstellung ergibt sich aufgrund von arithmetischen Berechnungen
    z.B. Finite-Elemente-Methode
  - Ein Experte des Anwendungsgebietes steht nicht zur Verfügung
  - Die Frage der Verantwortung und Wartung des Systems im Betriebszustand ist ungewiß
  - Das Entwicklungsprojekt steht unter Zeitdruck und die zur Verfügung stehenden Mittel sind gering
  - Das Teilprojekt stellt einen kritischen Pfad eines übergeordneten Projektes dar
  - Die Wissensinhalte der Problemstellung lassen sich nicht oder nur schwer verbalisieren

Die oben genannten Kriterien können zur konkreten Beurteilung von produktionstechnischen Gesichtspunkten herangezogen werden. Falls eines der k.o.-Kriterien zutrifft, erübrigen sich alle weiteren Untersuchungen, da eine wissensbasierte Lösung nur mit einem sehr hohen Risiko realisiert werden könnte.

Abb. ?? zeigt die Bewertung eines Planungsproblems zur Maschinen- und Spannmittelauswahl bei der Drehbearbeitung aufgrund der angegebenen Kriterien durch das CIM-Centrum Kaiserslautern (CCK). Ist die Bewertungssumme B in dem dargestellten Schema positiv, erscheint eine wissensbasierte Problemlösung aus technischer Sicht sinnvoll. Ansonsten sollte eine konventionelle Methode angestrebt werden. Das CCK hat das Schema auch auf weitere in der Entwicklung befindliche Expertensysteme im Bereich der technischen Diagnose und Planung angewendet und kam zu folgenden Bewertungssummen:

Diagnose von CNC-Meßgeräten (KODEX):	30 Punkte
Diagnose für Schaltplatten eines PKW-Getriebes:	21 Punkte
Expertensystem zur Auswahl keramischer Schneidstoffe (EWAKS):	3 Punkte
Expertensystem zur Zerspanbarkeit von Stahl (ESZ):	57 Punkte
Expertensystem zur Kostenplanung beim Bohren (BOHREX):	-3 Punkte

Die dargestellte Bewertungsmethode ist nicht unumstritten, weil eine Abgrenzung von Expertensystemen und konventionellen EDV–Lösungen nicht unproblematisch und eindeutig vollzogen werden kann, und weil praktisch alle Teile der KI mit entsprechendem Aufwand auch konventionell zu realisieren sind.

Betrachtet man die beiden Lösungen von einem hohen Abstraktionsniveau aus, so ist der wohl bedeutende Vorteil von Expertensystemen darin zu sehen, daß der Programmierer sich primär mit der Repräsentation des Sachwissens in einer der Problemstruktur angemessenen Form beschäftigt und das Kontrollwissen zunächst einmal sekundären Charakter hat.

k.o. Kriterien

-hohe Echtzeitanforderungen
-Komplexität durch Arithmetik
-Experte nicht verfügbar
-Problemstellung nicht verbalisierbar
-Entwicklung unter Zeitdruck
-Teilprojekt kritischer Pfad
-Verantwortung ungewiss

Kriterium	Gewichtung G (1) Diagnose	Gewichtung G (1) Planung	Erfüllungsgrad E (2)	G * E (3)
1 Dynamik der Problemstellung	2	2	*3*	*6*
2 Unvollständiger Wissensvorrat	3	3	*3*	*9*
3 Informationsvorrat z.T. subjektiv	1	1	*-3*	*-3*
4 Modellbildung nicht möglich	2	2	*0*	*0*
5 Unterschiedliche Wissensgebiete	1	1	*-3*	*-3*
6 Erklärungsfähigkeit notwendig	3	3	*-3*	*-9*
7 Systemeinsatz auch zur Schulung	1	1	*3*	*3*
8 Mehrere ähnliche Lösungen möglich	3	2	*3*	*9*
9 Kontextabhängige Daten	1	1	*-3*	*-3*
10Schwierige Bewertung von Alternativen	1	2	*3*	*3*
11Häufiges Generieren neuer Lösungen	1	3	*-3*	*-3*
12Schwierige Bewertung von Lösungen	2	3	*3*	*6*
13Viele Beispiele vorhanden	3	1	*3*	*9*
14Viel interaktiver Dialog	1	1	*-3*	*-3*
15Mehrere Experten	2	2	*-3*	*-6*
16Prototypentwicklung notwendig	3	3	*3*	*9*
17Dynamischeer Wissenszuwachs	2	2	*-3*	*-6*
18Sicherung von Wissen gefordert	1	1	*3*	*3*

Bewertung B = $\sum_{n=1}^{18} S_n = \boxed{+21}$

(1)
3 = sehr wichtig
2 = wichtig
1 = unwichtig

(2)
-3 = nicht erfüllt
0 = unbekannt
3 = erfüllt

(3)
Maximale Summe in Punkten:
Diagnose = 99
Planung = 102

Abbildung 2.10: Bewertungstabelle für Planungsproblem Spannmittel- und Maschinenauswahl

2.6.3 Expertensysteme in der Arbeitsplanung

Der Arbeitsplanung als Mittelpunkt des Informationsflusses zwischen Konstruktion und Fertigung kommt in einer rechnerintegrierten Produktionsstruktur eine besondere Bedeutung zu, da sie bei einer CAD/CAM-Kopplung Schnittstellenfunktion übernimmt und für die Produktionsplanung und -steuerung umfangreiche Basisdaten liefert (Abb. ??).

Besonders im Bereich des Maschinenbaus mit ausgeprägter Einzel- und Kleinserienfertigung überwiegt der Anteil der kurzfristigen Planungsaktivitäten deutlich. Ausgehend von dieser Tatsache läßt sich die Bedeutung der Arbeitsplanung als zeitintensive und vor allem kostenverantwortliche Tätigkeit gar nicht hoch genug einschätzen, so daß die Forderung nach einer Systematisierung und Automatisierung durch den Einsatz rechnerunterstützter Hilfsmittel zwangsläufig auftritt. Ziel der Arbeitsplanung ist es, eine Fertigungsaufgabe, die durch die Zeichnung und Stückliste beschrieben ist, den firmenspezifischen Fertigungsbedingungen und Anlagen unter Beachtung der wirtschaftlichen und organisatorischen Rahmenbedingungen zuzuordnen. Abhängig vom jeweiligen Unternehmen und dem Produktspektrum ergeben sich spezifische Anforderungen bezüglich des Detaillierungsgrades und der Genauigkeit in der Planung.

Wissensbasierte Systeme zur Arbeitsplanung

Die Schwierigkeiten der rechnerunterstützten Arbeitsplanung beruhen auf der Tatsache, daß die benötigten Informationen in hohem Grad qualitativen Änderungen unterworfen und oftmals nur als heuristisches Wissen in den Köpfen der verschiedenen Arbeitsplaner vorhanden sind. Solche Informationen lassen sich nur bedingt mit Hilfe herkömmlicher Daten- und Programmstrukturen effizient verarbeiten. Bei Expertensystemen ist die Planungslogik in keine starre Ablaufstruktur eingebunden, sondern liegt getrennt in Form von Planungswissen und Inferenzmechanismus vor. Diese Trennung bietet den entscheidenden Vorteil, daß Wissen quasi ungeordnet und inkrementell erweiterbar implementiert werden kann. Der einmal realisierte Inferenzmechanismus bleibt im allgemeinen unberührt.

Fallbasierte Planer in der Arbeitsplanung

Die Arbeiten an der Entwicklung generativer Planungsexpertensysteme haben gezeigt, daß neben den Schwierigkeiten zur Bewältigung der Beschreibungsproblematik der Aspekt Dynamik und das Verhältnis zwischen Planungsaufwand und der Qualität der Lösung kaum Berücksichtigung finden. Insbesondere fehlen Konzepte, um die Planungslogik mit vertretbarem Aufwand an die sich stets ändernde Technologie anzupassen. In aller Regel ist der Aufwand zur Aktualisierung des Planungswissens recht groß und es ist schwierig, die Qualität der erzeugten Pläne über einen längeren Zeitraum auf einem ausreichend hohen Niveau zu halten.

Entgegen dem Rechnersystem besitzt der Arbeitsplaner die Fähigkeit, permanent zu lernen, und sein vorhandenes Wissen zu ergänzen und anzupassen. Ohne das entsprechende *feed-back* aus der Fertigung, also einer Kopplung zwischen Planung und Planausführung, wäre ein Arbeitsplaner langfristig nicht in der Lage, gute Pläne zu erzeugen.

Die Informationen aus der Planausführung (Fertigung) tragen dabei weniger zu einer Erweiterung der grundlegenden Planungslogik bei als vielmehr zur Anreicherung des Erfahrungswissens. Die Analyse des Problemlösungsverhaltens eines Arbeitsplaners, aber auch die Tatsache, daß die Mitarbeiter der Arbeitsplanung meist über langjährige Erfahrung verfügen, zeigen deutlich, daß erst die Fähigkeit des Abstrahierens, also die Auswahl und Anpassung alter Planungsergebnisse an eine geforderte Aufgabenstellung, zu guten Ergebnissen führt. Diese Vorgehensweise erlaubt dem Arbeitsplaner, im Sinne einer Ähnlichkeitsplanung die Komplexität des Problems zu reduzieren und so mit vertretbarem Aufwand zu qualitativ guten Plänen zu gelangen.

Eine ähnliche Vorgehensweise wird beim fallbasierten Planen verfolgt. In Kombination mit einem generativen Planungssystem scheint ein solcher Ansatz im Bereich der Arbeitsplanung sehr vielversprechend zu sein,weil beide Planungstechniken für sich betrachtet Vor- und Nachteile besitzen, die sich jedoch in einer Kombination beider Verfahren größtenteils kompensieren.

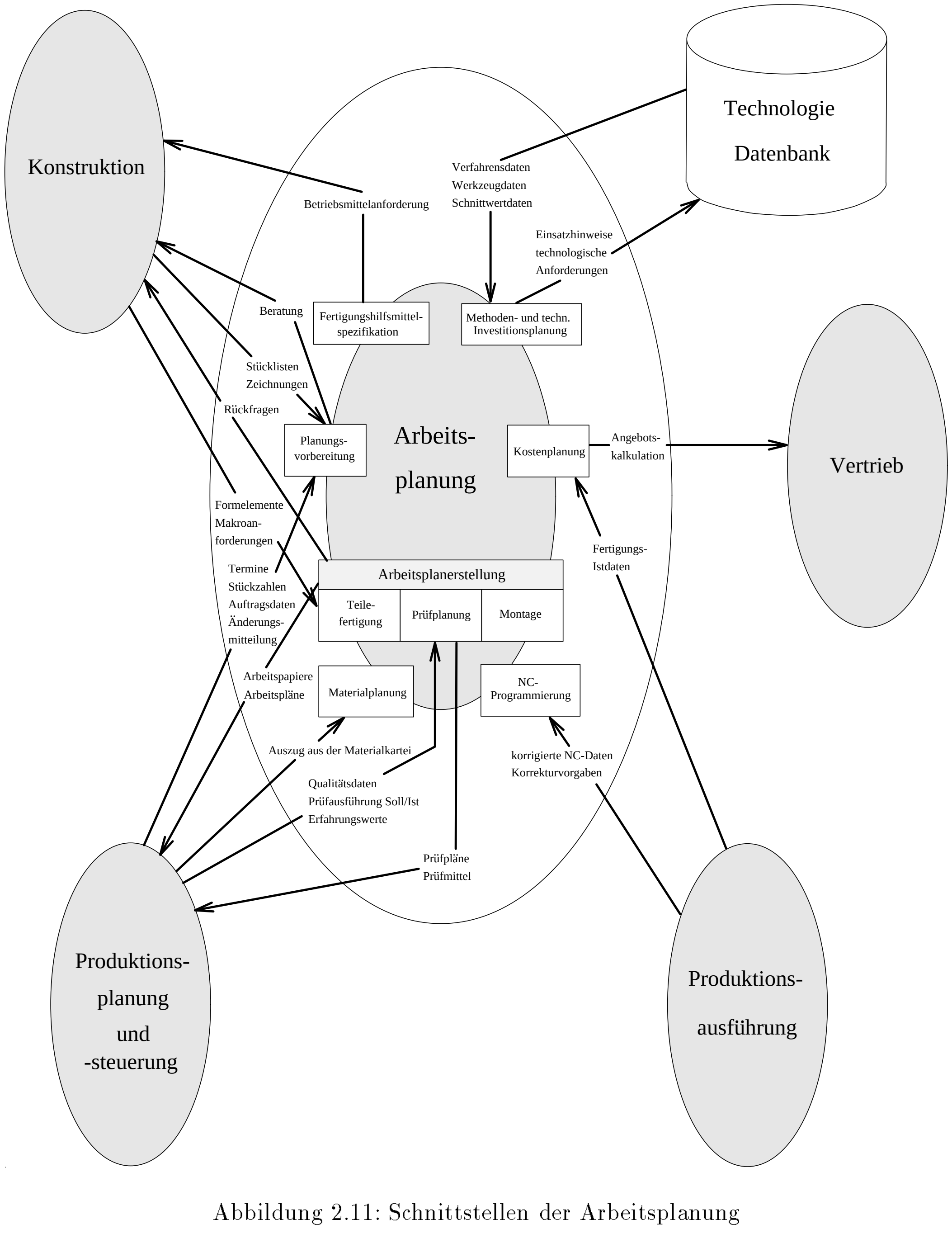


Abbildung 2.11: Schnittstellen der Arbeitsplanung

Kapitel 3

Arbeitsdomäne Drehen

Als eine reale Anwendungsdomäne wurde die Arbeitsplanerstellung für rotationsssymmetrische Drehteile mit Hilfe der Zerspanart **Drehen** gewählt. Wesentliche Aufgabe der Arbeitsplanerstellung ist hierbei die Auswahl und zeitliche Anordnung von Bearbeitungsschritten, so daß ein Rohteil in ein Fertigteil übergeführt werden kann. Als Randbedingungen müssen die vorhandenen Betriebsmittel Maschine, Spannmittel und Werkzeug beachtet werden; außerdem muß der fertige Plan zeit- und kostenoptimiert sein. Somit bietet sich mit der Arbeitsdomäne Drehen eine genügend komplexe Anwendungsumgebung für ein fallbasiertes Planungssystem. Im folgenden Kapitel wird diese Domäne aus der Sicht des Maschinenbauers im einzelnen vorgestellt.

3.1 Grundlagen der Zerspanung – Definitionen

3.1.1 Bezugsebenen

Um die Winkel am Schneidkeil definieren zu können, wird ein rechtwinkliges Bezugssystem zugrunde gelegt (Abb. 3.1).

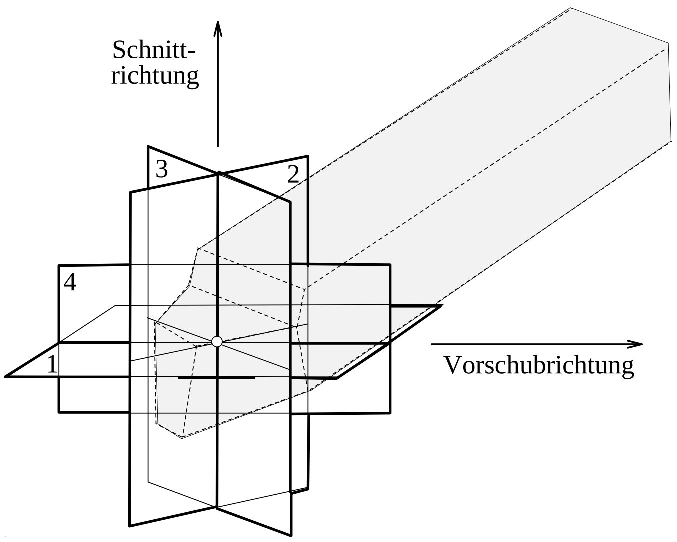


Abbildung 3.1: Bezugssystem zur Definition der Winkel am Schneidkeil

Es besteht aus drei Ebenen: der Werkzeugbezugsebene, der Schneidenebene und der Keilmeßebene. Die Arbeitsebene wurde als zusätzliche Hilfsebene eingeführt.

- *Werkzeugbezugsebene* 1
  ist eine Ebene durch den betrachteten Schneidenpunkt, senkrecht zur Schnittrichtung und parallel zur Vorschubsrichtung.
- *Schneidenebene* 2
  ist eine die Hauptschneide enthaltende Ebene, senkrecht zur Werkzeugsbezugsebene.
- *Keilmeßebene* 3
  ist eine Ebene senkrecht zur Schneidenebene und senkrecht zur Werkzeugsbezugsebene.
- *Arbeitsebene* 4
  ist eine Ebene, die die Schnittrichtung und die Vorschubrichtung enthält. In ihr vollziehen sich die Bewegungen, die an der Spanentstehung beteiligt sind.

3.1.2 Flächen, Schneiden und Ecken am Schneidkeil

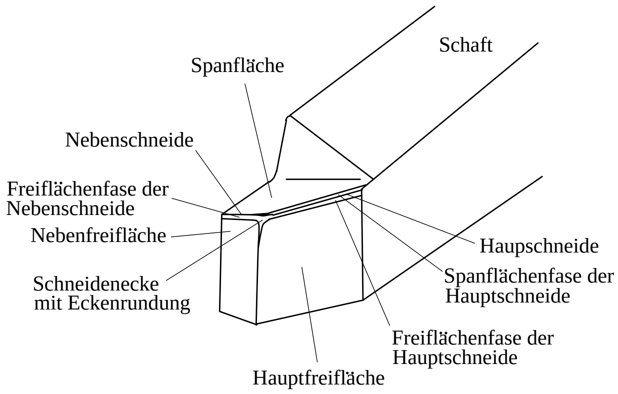


Abbildung 3.2: Flächen, Schneiden und Ecken am Schneidkeil

- *Freiflächen*
  sind die Flächen am Schneidkeil, die den entstehenden Schnittflächen zugekehrt sind. Wird eine Freifläche angefast, dann bezeichnet man diese Fase als Freiflächenfase.
- *Spanflächen*
  sind die Flächen, über die der Span abläuft. Wird die Spanfläche angefast, dann bezeichnet man diese Fase als Spanflächenfase.
- *Schneiden*
  - *Hauptschneiden*
    sind die Schneiden, deren Schneidkeil, bei Betrachtung in der Arbeitsebene, in Vorschubrichtung weist.
  - *Nebenschneiden*
    sind Schneiden, deren Schneidkeil in der Arbeitsebene nicht in Vorschubrichtung weist.
- *Ecken*
  - *Schneidenecke*
    ist die Ecke, an der Haupt- und Nebenschneide mit gemeinsamer Spanfläche zusammentreffen.
  - *Eckenrundung*
    ist die Rundung der Schneidenecke (der Rundungsradius r wird in der Werkzeugbezugsebene gemessen).

3.1.3 Die Winkel am Schneidkeil

- *Winkel, die in der Werkzeugbezugsebene gemessen werden* (Abb. ??)

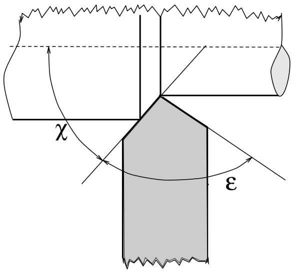


Abbildung 3.3: Einstellwinkel χ; Eckenwinkel ϵ

  – *Einstellwinkel* χ
    ist der Winkel zwischen Arbeitsebene und Schneidenebene.

    Der Einstellwinkel bestimmt die Lage der Hauptschneide zum Werkstück (Abb. ??). Vom Einstellwinkel ist bei gegebener Schnittiefe a_p die Eingriffslänge b ($\hat{=}$ Spanungsbreite) der Hauptschneide abhängig. Je kleiner der Einstellwinkel, um so größer wird die Eingriffslänge der Hauptschneide. Der Einstellwinkel bestimmt aber auch die Kräfte beim Zerspanen. Je größer der Einstellwinkel wird, um so größer wird die Vorschubkraft F_f und um so kleiner die Passivkraft F_p (Abb. ??). Deshalb erfordern labile Werkstücke immer einen großen Einstellwinkel.

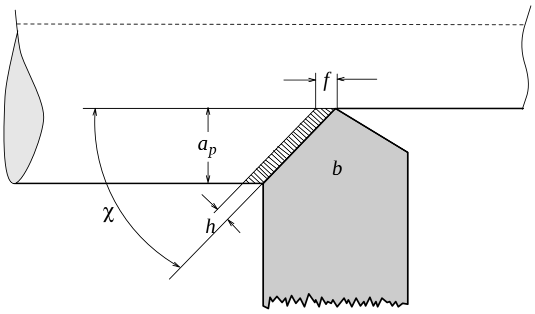


Abbildung 3.4: Eingriffslänge b ist bei gegebener Schnittiefe a_p abhängig vom Einstellwinkel χ

    * Kleine Einstellwinkel (um 10^o)
      ergeben große Passivkräfte F_p, die das Werkstück durchbiegen. Deshalb werden kleine Einstellwinkel nur bei sehr steifen Werkstücken angewandt.
    * Mittlere Einstellwinkel (45 bis 70^o)
      werden für stabile Werkstücke eingesetzt. Ein Werkstück gilt als stabil, wenn:

$$l \leq 6 \cdot d \tag{3.1}$$

l	in mm	Länge des Werkstücks
d	in mm	Durchmesser des Werkstücks

∗ Große Einstellwinkel (70 bis 90^o)
verwendet man bei langen labilen Werkstücken. Darunter versteht man Werkstücke, bei denen

$$l > 6 \cdot d \tag{3.2}$$

ist. Bei $\chi = 90^o$ ist die Passivkraftkomponente (Abb. ??) gleich Null. Dadurch ist beim Zerspanvorgang keine Kraft mehr vorhanden, die das Werkstück durchbiegen kann.

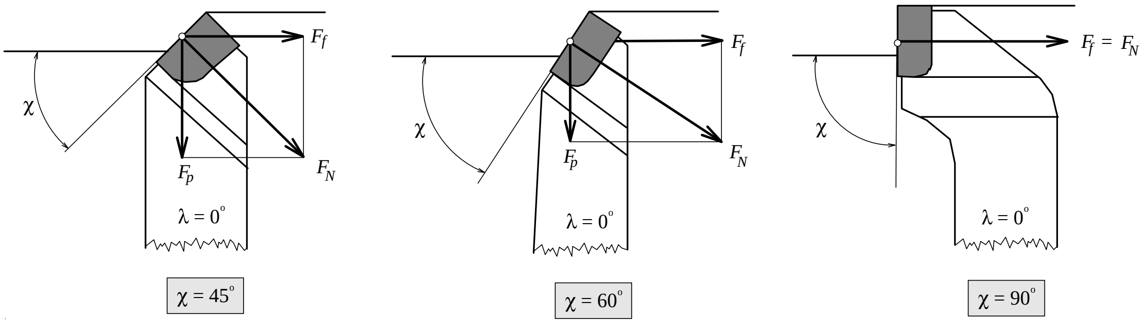


Abbildung 3.5: Einfluß des Einstellwinkels χ auf die Vorschubkraft F_f und die Passivkraft F_p

Der Einstellwinkel bestimmt die Spanungsgrößen nach folgenden Formeln:

∗ Eingriffslänge b
ist die Breite des abzunehmenden Spanes senkrecht zur Schnittrichtung, gemessen in der Schnittfläche.

$$b = \frac{a_p}{\sin \chi} \tag{3.3}$$

b	in mm	Eingriffslänge
a_p	in mm	Schnittiefe
χ	in ^{o}	Einstellwinkel

∗ Spanungsdicke h
ist die Dicke des abzunehmenden Spans senkrecht zur Schnittrichtung, gemessen senkrecht zur Schnittfläche.

$$h = f \cdot \sin \chi \tag{3.4}$$

h	in mm	Spanungsdicke
f	in mm	Vorschub pro Umdrehung

∗ Spanungsquerschnitt A
ist der Querschnitt des abzunehmenden Spanes, senkrecht zur Schnittrichtung.

$$A = a_p \cdot f = b \cdot h \tag{3.5}$$

A	in mm^2	Spanquerschnitt

– *Eckenwinkel* ϵ
ist der Winkel zwischen Haupt- und Nebenschneide.

Der Eckenwinkel ist meistens 90^o. Nur bei der Bearbeitung scharfer Ecken wird ϵ kleiner als 90^o gewählt. Beim Kopierdrehen verwendet man Eckenwinkel zwischen 50 und 58^o, bei schwerer Zerspanung kann ϵ bei Schruppdrehmeißeln bis 130^o sein.

- *Winkel, der in der Schneidenebene gemessen wird*
  Neigungswinkel λ (Abb. ??)
  ist der Winkel zwischen Werkzeugbezugsebene und Hauptschneide. Der Neigungswinkel ist negativ, wenn die Schneide von der Spitze her ansteigt. Er bestimmt, welcher Punkt der Schneide zuerst in das Werkzeug eindringt.

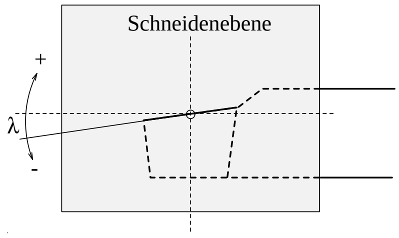


Abbildung 3.6: Neigungswinkel λ

  Der Neigungswinkel beeinflußt die Ablaufrichtung des Spans und bestimmt die Neigung der Hauptschneide.

  – Negativer Neigungswinkel
    verschlechtert den Spanabfluß, aber entlastet die Schneidenspitze, weil bei negativem Neigungswinkel nicht die Spitze, sondern die Schneidenbrust zuerst in das Werkstück eindringt. Deshalb wird der negative Neigungswinkel für Schruppwerkzeuge und Werkzeuge für unterbrochenen Schnitt eingesetzt. Man arbeitet dort mit $\lambda = -3$ bis -8^o.
  – Positiver Neigungswinkel
    verbessert den Spanablauf.

- *Winkel, die in der Keilmeßebene gemessen werden*
  (Abb. ??)

  – *Freiwinkel* α
    ist der Winkel zwischen Freifläche und Schneidenebene.
    Die normale Größenordnung des Freiwinkels liegt bei 6 bis 10^o.
    * Große Freiwinkel
      · führen zu Wärmestau in der Schneidenspitze
      · schwächen den Schneidkeil (Ausbruchgefahr)
    * Kleine Freiwinkel
      · führen zur Verstärkung des Schneidkeils
      · verbessern die Oberfläche, solange das Werkzeug nicht drückt. Drückt das Werkzeug jedoch, kommt es zur Erwärmung des Werkzeugs und zu großem Freiflächenverschleiß.
      · wirken schwingungsdämpfend
  – *Keilwinkel* β
    ist der Winkel zwischen Freifläche und Spanfläche.
    Der Keilwinkel soll für harte und spröde Werkstoffe groß und für weiche Werkstoffe klein sein.
  – *Spanwinkel* γ
    ist der Winkel zwischen Spanfläche und Werkzeugbezugsebene.
    Beim Drehen mit Hartmetallwerkzeugen liegen die Spanwinkel bei der Bearbeitung von Stahl mittlerer Festigkeit zwischen 0 und $+6^o$, in Ausnahmefällen bis $+18^o$. Bei Vergütungsstählen und Stählen hoher Festigkeit verwendet man Spanwinkel zwischen -6 und $+6^o$.

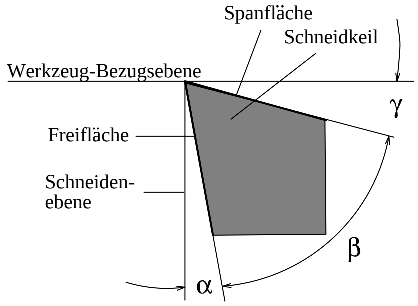


Abbildung 3.7: Freiwinkel α; Keilwinkel β; Spanwinkel γ

* Große Spanwinkel
  werden bei weichen Werkstoffen (weiche Stähle, Leichtmetall. Kupfer), die mit zähen Hartmetallen bearbeitet werden, verwendet. Je größer der Spanwinkel, um so
  · besser ist der Spanabfluß
  · kleiner ist die Reibung
  · geringer ist die Spanstauchung
  · besser wird die Oberfläche des Werkstücks
  · kleiner werden die Schnittkräfte

  Große Spanwinkel haben aber auch Nachteile. Sie
  · schwächen den Schneidkeil
  · verschlechtern die Wärmeabfuhr
  · erhöhen die Gefahr des Schneidenausbruchs

  Sie verkleinern also die Standzeit des Werkzeugs.
* Kleine Spanwinkel
  bis hin zu negativen Spanwinkeln wendet man bei der Schruppbearbeitung und bei Werkstoffen mit hohen Festigkeiten an. Als Werkzeugstoff werden hierfür abriebfeste Hartmetalle eingesetzt. Kleine Spanwinkel
  · stabilisieren den Schneidkeil
  · erhöhen die Standzeit der Werkzeuge
  · ermöglichen das Drehen mit hohen Schnittgeschwindigkeiten
  · verringern die Bearbeitungszeit

  Weil die Schnittkräfte aber mit kleiner werdendem Spanwinkel steigen, haben kleine Spanwinkel zur Folge:
  · Anstieg der Schnittkräfte
  · Anstieg der erforderlichen Antriebsleistung

Für den Freiwinkel, den Keilwinkel und den Spanwinkel gilt immer die Beziehung:

$$\alpha + \beta + \gamma = 90^o \tag{3.6}$$

Sind die Flächen angefast (Abb. ??), dann bezeichnet man die Fasenwinkel als:

Fasenfreiwinkel α_f
Fasenkeilwinkel β_f
Fasenspanwinkel γ_f

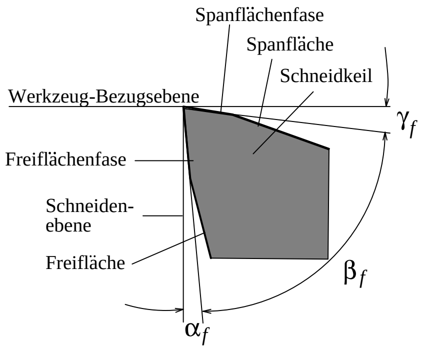


Abbildung 3.8: Schneidkeil mit Fasen: Freiwinkel α_f; Keilwinkel β_f; Spanwinkel γ_f

Auch hier gilt die Beziehung:

$$\alpha_f + \beta_f + \gamma_f = 90^o \tag{3.7}$$

Sowohl die Spanflächen- als auch die Freiflächenfase verstärken den Schneidkeil. Durch die Spanflächenfase entsteht eine Schneidengeometrie, die zwei Vorteile vereinigt:

1. der große Spanwinkel verbessert den Spanablauf.

2. der kleine (bis negative) Fasenspanwinkel γ_f verstärkt den Schneidkeil, verbessert die Wärmeabfuhr und vermindert die Ausbruchgefahr.

Die Breite der Spanflächenfase darf nicht zu groß sein, weil sonst der Span nicht mehr auf der Spanfläche ablaufen kann. Für die Einstellwinkel zwischen 60^o und 90^o kann man die Breite der Spanflächenfase rechnerisch wie folgt bestimmen:

$$b_{f_\gamma} \approx 0,8 \cdot f \tag{3.8}$$

b_{f_γ}	in mm	Breite der Spanflächenfase
f	in mm	Vorschub pro Umdrehung

3.2 Drehverfahren

Drehen ist ein Zerspanverfahren, bei dem die Schnittbewegung vom Werkstück und die Hilfsbewegung (Vorschub- und Zustellung) vom Werkzeug ausgeführt werden. Vorschub- und Zustellung werden bei den meisten Drehmaschinen mittels Längs- und Querschlitten erzeugt (Abb. ??). Das zum Drehen verwendete Werkzeug, der Drehmeißel, hat *eine* Hauptschneide.

Einfache Drehteile erhalten ihre Form durch eine Vorschubbewegung in Richtung der Drehachse oder senkrecht dazu. Die zugehörigen Verfahren werden benannt nach der Richtung der Vorschubbewegung, die während der Bearbeitung abläuft. Die Kontur des Fertigteils entsteht meist durch mehrere Schnitte. Die Zustellung erfolgt vor jedem Schnitt außerhalb des Werkstücks.

Sollen kegelige Werkstücke oder Werkstücke mit gekrümmten Begrenzungslinien hergestellt werden, so sind drei Verfahren anwendbar: Formdrehen, Drehen mit schräg gestelltem Oberschlitten (Kegeldrehen) und Drehen mit gleichzeitiger und gesteuerter Bewegung von Längs- und Querschlitten (Kopierdrehen, NC-Drehen).

3.2.1 Längsdrehen

Beim Längsdrehen bewegt sich der Drehmeißel parallel zur Werkstückachse, in der Regel von rechts nach links (Abb. ??). Es wird angewandt, um einem zylindrischen Werkstück ein bestimmtes Durchmessermaß zu geben.

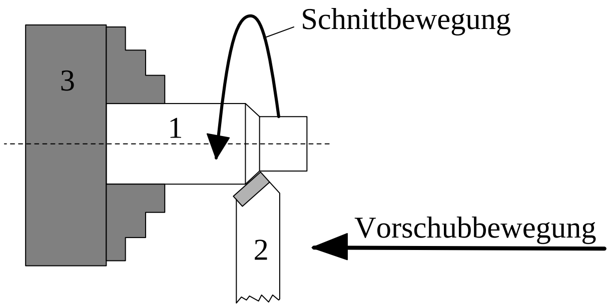


Abbildung 3.9: Schnitt- und Vorschubbewegung beim Längsdrehen
1 Werkstück, 2 Werkzeug, 3 Spannmittel

3.2.2 Plandrehen

Beim Plandrehen (Abb. ??) bewegt sich das Werkzeug senkrecht zur Werkstückachse. Es wird angewandt, um eine Endfläche oder einen Absatz zu bearbeiten. Die Bewegungsrichtung des Drehmeißels ist abhängig von der Art der Bearbeitung, der Schneidenform und der Stellung des Werkzeugs sowie der Form des Werkstücks (Hohlteil, Vollteil). Beim Schruppen wird eine Bewegung von außen nach innen, beim Schlichten von innen nach außen bevorzugt.

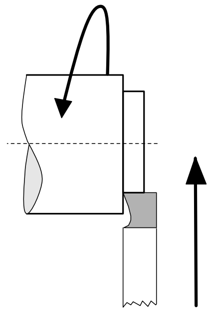


Abbildung 3.10: Vorschubrichtung des Werkzeugs beim Plandrehen

3.2.3 Stechen (Stechdrehen)

Beim Stechen bewegt sich das Werkzeug senkrecht oder parallel zur Werkstückachse. Die Kontur entsteht meist durch einmaliges Stechen auf Fertigtiefe.

Einstechen

Das Einstechen wird angewandt, um eine Nut bestimmter Form zu erzeugen. Wenn die Nutform wie in Abbildung ?? gerade ist und parallel zur Werkstückachse verläuft, dann ist beim Einstechen die gesamte Breite der Hauptschneide des Stechdrehmeißels im Einsatz. Der Eckenwinkel beträgt hier 90^o.

Abstechen

Wenn man ein fertigbearbeitetes Werkstück von der Stange abstechen will, dann geschieht dies mit dem Abstechverfahren. Im Gegensatz zum Einstechen ist beim Abstechen die Hauptschneide zur

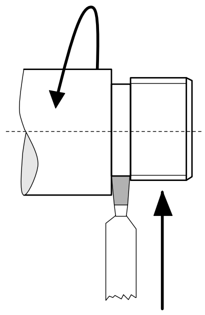


Abbildung 3.11: Einstechen einer Nut

Werkstückachse geneigt (Abb. ??). Der Eckenwinkel des Abstechstahls ist kleiner als 90^o. Dadurch entstehen zwei verschiedene Zapfendurchmesser (d_1, d_2). Dies hat zur Folge, daß in der Endphase des Abstechvorgangs das am kleineren Zapfendurchmesser hängende Teil ohne Restzapfen abbricht.

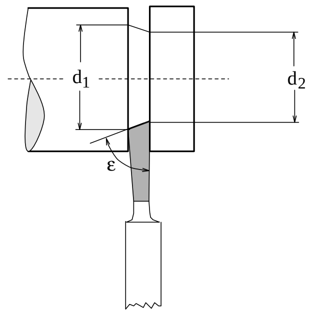


Abbildung 3.12: Lage der Hauptschneide des Stechdrehmeißels beim Abstechen $\epsilon < 90^o$

Ausstechen

Das Ausstechen ist ein Stechdrehen, bei dem die Vorschubrichtung des Stechdrehmeißels parallel zur Werkstückachse liegt. Es wird angewendet, um aus einer Platte eine große Scheibe herauszutrennen oder zur Herstellung von Nuten an Stirnflächen.

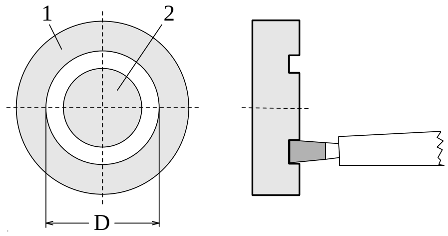


Abbildung 3.13: Ausstechen einer großen Bohrung
1 Werkstück, 2 ausgestochenes Abfallstück, D Bohrungsdurchmesser im Werkstück

3.2.4 Formdrehen

Beim Formdrehen (Abb. ??) hat die Hauptschneide des Drehmeißels die am Werkstück zu erzeugende Form. Die Form wird in der Regel im Einstechverfahren in das Werkstück eingebracht. So erzeugte Werkstücke bezeichnet man auch als Formdrehteile und die Werkzeuge als Formdrehmeißel.

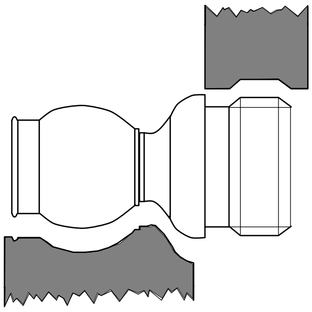


Abbildung 3.14: Mit Formdrehmeißel im Stechverfahren hergestelltes Formdrehteil

3.2.5 Kegeldrehen

Kegeldrehen ist ein Längsdrehen, bei dem sich der zu erzeugende Durchmesser stetig ändert. Es wird angewandt, um konische Wellen zu erzeugen. Der Kegelwinkel α läßt sich nach folgender Gleichung berechnen:

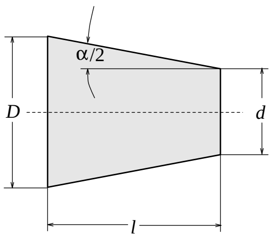


Abbildung 3.15: Kenngrößen des Kegels

$$tg\frac{\alpha}{2} = \frac{D-d}{2l} \qquad (3.9)$$

3.2.6 Kopierdrehen und NC-Drehen

Beim Kopierdrehen wird die Werkstückform von einem Formspeicher (Kegellineal, Schablone) abgegriffen und auf den Längs- und Querschlitten übertragen.

Statt der analogen Abbildung der Sollform des Werkstücks in der Schablone kann die Form auch durch die charakteristischen Maße numerisch beschrieben werden. Diese Beschreibung wird von der NC-gesteuerten Maschine in den Steuerungsteil eingelesen und in Signale für die regelbaren Antriebe (Längs- und Querschlitten) umgesetzt. Mit NC-gesteuerten Maschinen lassen sich alle mathematisch beschreibbaren Konturen an Drehteilen erzeugen, sofern die Schneidenform des Werkzeugs einen Schnitt ermöglicht.

3.2.7 Gewindedrehen

Gewindedrehen ist ein Längsdrehen, bei dem der Vorschub der Steigung des zu erzeugenden Gewindes entspricht. An der Drehmaschine wird der zum Gewindeschneiden erforderliche genaue Vorschub durch die Leitspindel und das Vorschubgetriebe erzeugt. Das Werkzeug zum Gewindedrehen ist der Gewindedrehmeißel (Abb. ??). Bei der numerisch gesteuerten Drehmaschine wird der Vorschub durch eine elektrische Verbindung zwischen Hauptspindel und Vorschubantriebsmotor angepaßt.

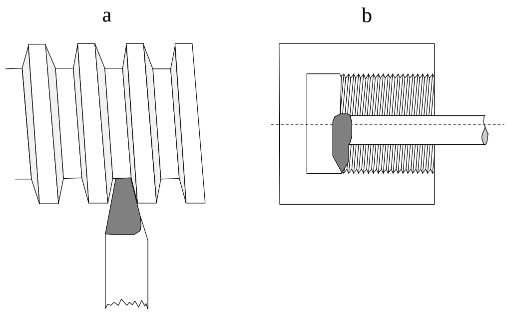


Abbildung 3.16: Werkstück mit Anordnung des Drehmeißels beim Gewindedrehen
a Außengewinde, b Innengewinde

3.3 Werkstückspannung

3.3.1 Radiale Lagebestimmung

Dreharbeiten werden meistens an zylindrischen Werkstücken durchgeführt, die zentrisch einzuspannen sind. Die Mittelachse des Werkstücks soll mit der Rotationsachse der Drehmaschinenspindel fluchten. Durch Anbringen von *Zentrierbohrungen* (Abb. ??) an beiden Werkstückenden kann die Mittelachse so markiert werden, daß die Zentrierspitzen der Drehmaschine eingreifen können. Am Außendurchmesser des Werkstücks oder am Innendurchmesser bei Hohlkörpern kann die radiale Lagebestimmung auch durch zentrisch spannende Dreibacken-Futter erreicht werden.

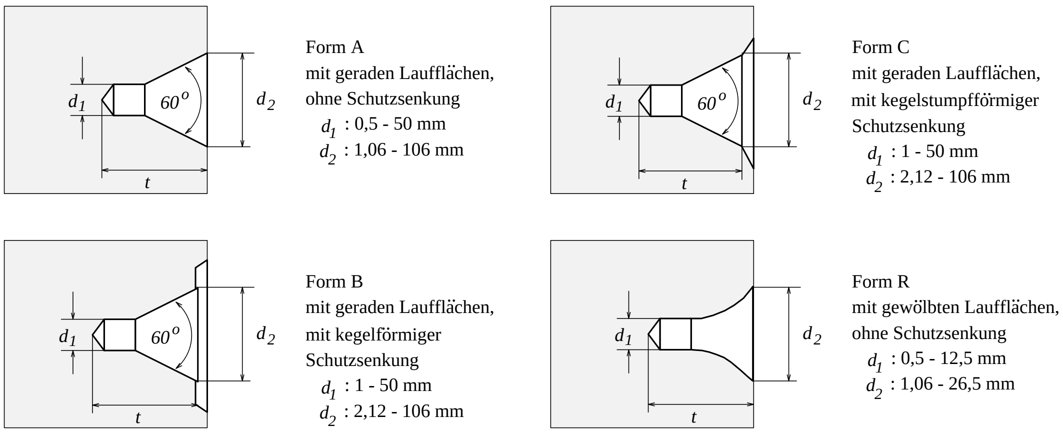


Abbildung 3.17: Zentrierbohrungen nach DIN 332

Die radiale Lage muß auch unter der Berücksichtigung der angreifenden Zerspankräfte und der *Werkstückelastizität* erhalten bleiben. Man kann unter dieser Betrachtungsweise stabile, halbstabile und unstabile Werkstücke unterscheiden.

Zur Verbesserung der Stabilität werden an schlanken Werkstücken Lünetten verwendet, die zusätzliche Stützstellen erzeugen. Bei futterspannenden Werkstücken kann mit einer Reitstockspannung am freien Ende die Starrheit vergrößert werden.

3.3.2 Axiale Lagebestimmung

Die axiale Lagebestimmung ist nötig, um der Werkstückbezugskante eine *wiederholbare Lage* im *Koordinatensystem* der Drehmaschine zuzuordnen (Abb. ??). Zu diesem Zweck müssen Anschlagpunkte an den Spannelementen vorhanden sein. Das sind im einfachsten Fall die Stirnseiten der Spannbacken, gegen die das Werkstück gelegt wird. Es können aber auch besondere Anschlagelemente verwendet werden, die in der Hohlspindel des Antriebs oder auf dem Werkzeugträger angeordnet sind.

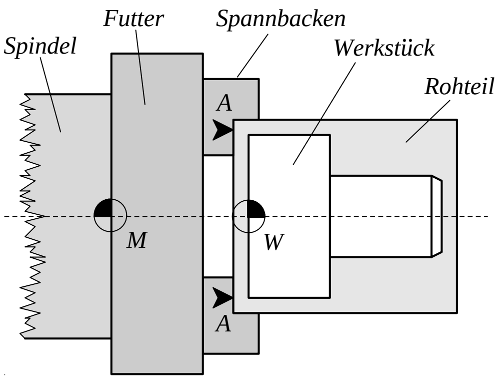


Abbildung 3.18: Axiale Lagebestimmung an einem im Futter gespannten Werkstück

M Nullpunkt des maschinengebundenen Koordinatensystems
A Anschlagpunkte
W Werkstückbezugspunkt

3.3.3 Übertragung der Drehmomente und Kräfte

Die Werkstückeinspannung wird mit Kräften und Momenten verschiedener Größe belastet:

1. der Gewichtskraft des Werkstücks
2. dem Beschleunigungsmoment beim Einschalten der Spindel
3. dem Schnittmoment, das von der Schnittkraft verursacht wird
4. der Vorschub- und der Passivkraft
5. der Reitstockkraft
6. Wechselkräften, die von Schwingungen herrühren.

Die unveränderliche Lage des Werkstücks muß dabei von der Einspannung gewährleitet werden. Diese kann kraftschlüssig durch Reibung, formschlüssig oder beidartig wirksam sein. In jedem

Fall muß das *Haltemoment* größer als die belastenden Drehmomente sein. Berechnungen sind oft unzuverlässig, weil der Reibungsbeiwert nur geschätzt werden kann. Tabellenwerte der Spannmittelhersteller, in die der Spanndurchmesser und die Spannkraft besonders eingehen, sind zuverlässige Unterlagen.

3.3.4 Spannelemente

Planscheiben

Sie werden zum Spannen von großen flachen oder nicht rotationssymmetrischen Teilen eingesetzt. Bei der Planscheibe ist jede Spannbacke einzeln verstellbar.

Selbstzentrierende Spannfutter

Die selbstzentrierenden Spannfutter haben meist 3 Spannbacken, die das Werkstück spannen und sich beim Spannen selbst zentrieren. Es gibt auch Spannfutter mit 2 oder 4 Spannbacken. Bei den selbstzentrierenden Spannfuttern werden alle Backen gleichzeitig durch

1. eine Spirale (Spiralspannfutter)
2. ein Spiralringfutter mit verstellbaren Backen
3. Keilstangen

mit einem Steckschlüssel verstellt.

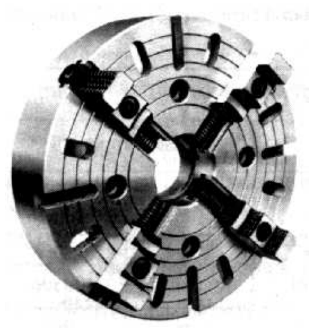


Abbildung 3.19: Planscheibe mit vier einzeln verstellbaren Backen

Spannzangen

An halb- oder vollautomatischen Maschinen arbeitet man überwiegend von der Stange. Als Spannelement verwendet man dabei die Spannzange. Man unterscheidet kraft- und formschlüssige Spannzangen. Bei den kraftschlüssigen Spannzangen (Abb. ??) wird der Kegel der Spannzange in den Gegenkegel des Spindelkopfes hineingezogen. Dabei drücken die elastischen Spannsegmente des Spannkegels auf die zu spannende Stange und halten sie fest. Weil aber das Stangenmaterial durch die vorhandene Durchmessertoleranz im Durchmesser abweicht, wird die Spannzange, je nach Toleranz, unterschiedlich weit in den Spannkegel hineingezogen. Dadurch ändert sich ihre Nullage. Dies kann auch am Werkstück zu Längenabweichungen führen.

Bei den formschlüssigen Spannzangen (Abb. ??) liegt die eigentliche Spannzange axial fest. Die Spannung wird durch eine Druckstange erzeugt, deren Innenkegel sich über den Außenkegel der Spannzange schiebt.

Der Spannbereich der Spannzangen umfaßt nur einige Zehntel Millimeter (im Mittel 0,2 mm). Man benötigt deshalb für die verschiedenen Stangendurchmesser eigene Spannzangen, deren Durchmesser dem Stangendurchmesser entsprechen.

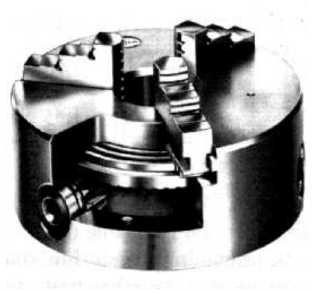


Abbildung 3.20: Spiralspannfutter

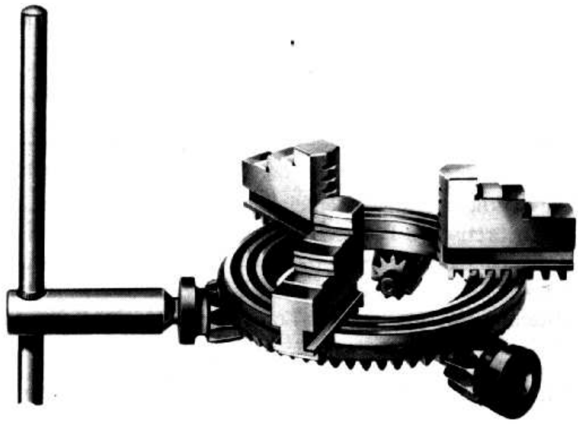


Abbildung 3.21: Spiralspannfutter

Spanndorne

Der Spanndorn (Abb. ??) wird zur Aufnahme von Werkstücken mit Bohrung benötigt. Der sogenannte Spreizdorn ist eine geschlitzte Hülse mit Innenkegel. In diese Hülse wird ein Gegenkegel mit Hilfe einer Spannmutter eingezogen. Dadurch spreizt sich der geschlitzte Dorn im Außendurchmesser und spannt das Werkstück.

Stirnseitenmitnehmer

Stirnseitenmitnehmer benötigt man für die Mitnahme von Wellen, die zwischen Spitzen gedreht werden sollen (Abb. ??).

Die Körnerspitze, in der die Welle zentriert wird, kann feststehend oder mitlaufend sein. Zur Mitnahme von Rohren verwendet man Drehgreifer (Abb. ??).

3.4 Werkstückgestalt

Bei Betrachtungen der Werkstückform müssen *Grobgestalt*, *Feingestalt* und der *Gefügeaufbau* unterschieden werden. Die Gestalt des Rohteils macht sich auf den Bearbeitungsablauf beim Drehen bemerkbar. Die Gestalt des fertigen Werkstücks wirkt sich in seinen Gebrauchseigenschaften aus.

Abbildung 3.22: Spiralringfutter mit verstellbaren Backen

Abbildung 3.23: Spiralringfutter mit verstellbaren Backen

3.4.1 Grobgestalt

Für die Bestimmung der Grobgestalt sind *Formenordnungen* aufgestellt worden, die Drehteile nach den Gesichtspunkten der Schlankheit, der Größe, der Wandstärke bei Hohlkörpern und der räumlichen Zusammensetzung komplexer Formen unterscheiden.

Die Formenordnungen haben den Zweck, gleichartige Werkstücke zu Formengruppen zusammenzufassen, die auf gleiche Art gefördert, sortiert, ausgerichtet, gespannt und möglichst auch bearbeitet werden können. Die Schwierigkeit besteht darin, daß eine sehr feine Gliederung notwendig ist. Diese wiederum vermehrt den organisatorischen Aufwand. Die Grobgestalt der Rohteile weicht oft stark von der Form der Fertigteile ab. Bei ihrer Herstellung durch Schmieden und Gießen können auch starke Unterschiede von Teil zu Teil entstehen. Sie machen sich bei der Drehbearbeitung in unterschiedlichen Schnittkräften und unterschiedlichen elastischen Verformungen des Werkstücks und des Werkzeugs bemerkbar. Deshalb soll die Bearbeitung in *mehreren Stufen*, etwa *Schruppen* und *Schlichten*, bei besonders großen Genauigkeitsanforderungen zusätzlich durch *Feindrehen* erfolgen.

Die Genauigkeitseigenschaften von Werkstücken werden als *Form-* und *Lagetoleranzen* in DIN 7184 beschrieben. Im einzelnen lassen sich *Geradheit*, *Ebenheit*, *Rundheit* und *Zylinderform* feststellen (s. Anhang ??). Bei mehreren Bearbeitungsstellen kommen *Parallelität*, *Rechtwinkligkeit*, *Konzentrizität*, *Planlauf* und *Rundlauf* hinzu. Abweichungen von der genauen Form werden als Gestaltabweichung 1. und 2. Ordnung nach DIN 4760 mit *Formabweichung* und *Welligkeit* bezeichnet. Die verschiedenen Form- und Lagetoleranzen lassen sich in Abhängigkeit von Nennmaß und Toleranzwert in ISO–Toleranzreihen (IT) einteilen (s. Anhang ??). Die mit Schlichtdrehen erreichbaren Maßgenauigkeiten liegen bei IT8 bis IT7, beim Feinschlichten sind bei optimalen

Abbildung 3.24: Keilstangenfutter

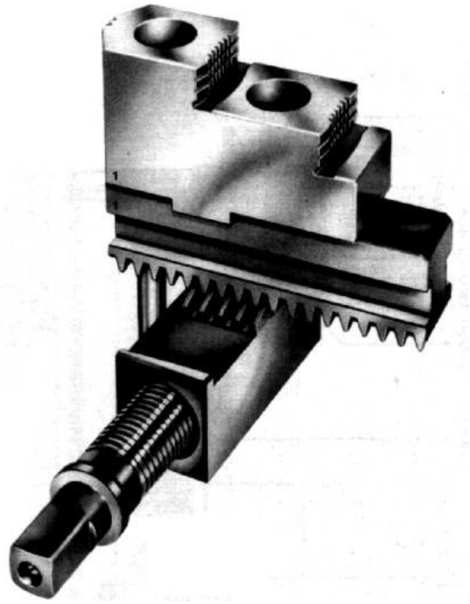


Abbildung 3.25: Keilstangenfutter

Drehbedingungen Genauigkeiten von IT6 zu erreichen.

3.4.2 Feingestalt

Die Oberfläche gedrehter Werkstücke ist von Spuren der Werkzeugschneiden gezeichnet. Man kann die *Rillen*, die von der Meißelform und der Vorschubbewegung erzeugt werden, und die *Riefen*, die ihre Ursache hauptsächlich in Verschleißspuren der Schneide haben, unterscheiden (s. Abb. ??). Beide tragen nach DIN 4760 als Gestaltabweichungen 3. und 4. Ordnung zur *Rauhheit* des Werkstücks bei.

Die *Rauhtiefe*, die die Schneidenform im Zusammenhang mit dem Vorschub erzeugt, läßt sich berechnen. Abb. ?? zeigt den Eingriff einer Schneide mit der Eckenrundung r, der Schnittiefe a_P und dem Vorschub f pro Werkstückumdrehung. Die theoretisch erzeugte Rauhtiefe R_{th} berechnet sich nach

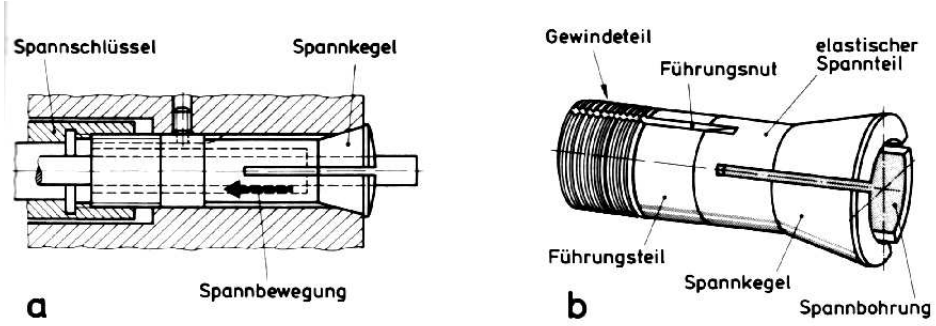


Abbildung 3.26: a kraftschlüssige Spannzangen-Spanneinrichtung, b auswechselbare Spannzange

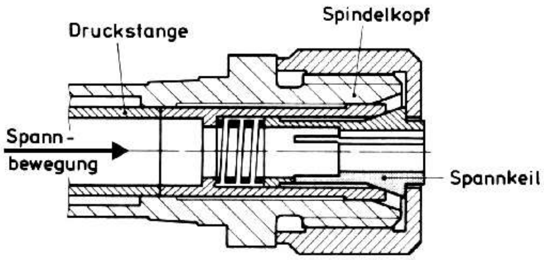


Abbildung 3.27: Formschlüssige Spannzangen-Spanneinrichtung

$$R_{th} = r - \sqrt{\frac{f^2}{4}} \tag{3.10}$$

Durch Vereinfachung des Ausdrucks über eine Reihenentwicklung nach Taylor erhält man

$$R_{th} \approx \frac{f^2}{8r} \tag{3.11}$$

Das bedeutet, daß die Rauhtiefe quadratisch mit dem Vorschub zunimmt und linear mit der Vergrößerung der Schneidenrundung kleiner wird. Für das Feindrehen empfiehlt sich daher vor allem ein kleiner Vorschub bis herab auf 0,1 oder gar 0,05 mm je Werkstückumdrehung. Eine Vergrößerung der Schneidenrundung ist auch eine günstige Maßnahme. In technischen Zeichnungen des Werkstücks wird die zulässige Rauhtiefe durch sog. Oberflächen- oder Bearbeitungszeichen dargestellt (s. Anhang ??). Bei Anwendung dieser Oberflächenzeichen muß in der Zeichnung (zweckmäßig im Schriftfeld oder darüber) die jeweils zugrundeliegende Reihe 1,2,3 oder 4 der jeweils zulässigen größten Rauhtiefe angegeben werden, z.B. „ Oberflächen Reihe 3 DIN 3141“. Weil mit dem Bearbeitungszeichen die Rauhigkeit meist vorgegeben ist, muß man wissen, mit welchem Vorschub bei gegebenem Spitzenradius r gefahren werden muß. Deshalb stellt man die Gleichung ?? nach dem Vorschub um.

$$f = \sqrt{8 \cdot r \cdot R_{th}} \tag{3.12}$$

Um die Werte nicht immer ausrechnen zu müssen, kann man für einige Formrauhigkeiten aus Tabelle ?? die erforderlichen Vorschübe bei gegebenem Spitzenradius r herauslesen.

Abbildung 3.28: Hülsenspanndorn

Abbildung 3.29: Stirnseitenmitnehmer

3.5 Drehwerkzeuge

3.5.1 Werkzeugausführungsformen

Ein Drehmeißel besteht aus Schaft und Schneidenkopf. Je nachdem, wie die Lage des Schneidenkopfes zum Schaft ist, unterscheidet man zwischen geraden, gebogenen und abgesetzten Drehmeißeln (s. Anhang ??).

Ein weiteres Unterscheidungsmerkmal ist die Bearbeitungsrichtung. Einen Drehmeißel, der von rechts nach links arbeitet, bezeichnet man als rechten, und einen Drehmeißel, der von links nach rechts arbeitet, als linken Drehmeißel.

Der Schaft kann quadratisch oder rechteckig sein. Bei den rechteckigen Schäften ist das Verhältnis von Schafthöhe zu Schaftbreite $h : b = 1,6 : 1$.

Wenn bei diesen Drehmeißeln nach mehrmaligem Nachschliff die Hartmetallplatte ausgewechselt werden muß, dann ist das bei diesen Werkzeugen sehr zeit- und kostenaufwendig. Überlegungen, wie man die Werkzeugkosten für die Wiederherrichtung der Werkzeuge senken kann, führten zur Entwicklung der Klemmhalter.

Im Klemmhalter werden Hartmetallschneidplatten durch ein Klemmsystem festgehalten (Abb. ??). Die Schneidplatten gibt es in verschiedenen Formen und Größen und verschiedenen Span- und Freiwinkeln (s. Anhang ??, Abb. ??). So hat eine quadratische Schneidplatte mit einem Spanwinkel von 0^o acht Schneiden. Durch Verdrehen der Platte im Klemmhalter bzw. durch Wenden der Platte können nacheinander alle acht Schneiden zum Einsatz gebracht werden. Wegen dieser

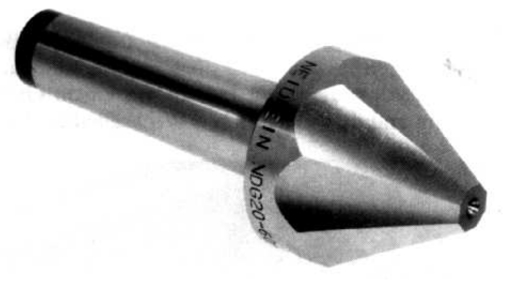


Abbildung 3.30: Drehgreifer zur Mitnahme von Rohren

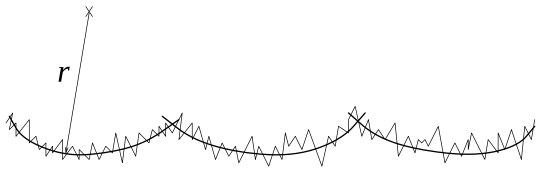


Abbildung 3.31: Oberflächenprofil an einem gedrehten Werkstück. Rillen als Abbildung der Schneidenecke mit dem Radius r und Riefen infolge Schneidenverschleißes und unterschiedlicher Werkstoffverfestigung

Möglichkeit des Umwendens bezeichnet man diese Platten als Wendeschneidplatten.

3.5.2 Sonderdrehwerkzeuge zum Gewindedrehen

Außengewinde

Zur Erzeugung von Außengewinden wird im Normalfall ein spitzer Drehmeißel DIN 4975 verwendet, bei dem der Spitzenwinkel dem Flankenteil des zu erzeugenden Gewindes entspricht. Zum Gewindeschneiden werden sowohl Hartmetall- als auch Schnellstahlwerkzeuge eingesetzt.

Bei den Schnellstahlwerkzeugen verwendet man bevorzugt Formgewindestähle. Bei dem in Abb. ?? abgebildeten Halter hat das auswechselbare Messer auf seiner ganzen Länge einen gleichbleibenden Flankenwinkel. Beim Nachschleifen wird das Messer nur oben an der ebenen Spanfläche geschliffen. Dadurch bleibt das Gewindeprofil, das nicht nachgeschliffen wird, bis zum Verbrauch des Messers voll erhalten.

Der Steigungswinkel wird mit Hilfe des schwenkbaren Halterkopfes eingestellt.

An Stelle des Formmessers verwendet man auch Formscheiben. Solche Formscheiben (Abb.

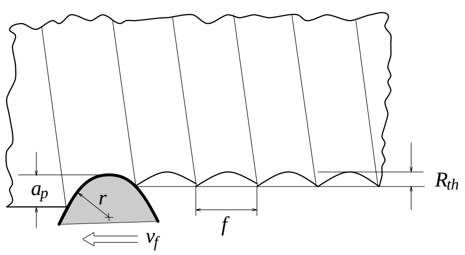


Abbildung 3.32: Abbildung der Werkzeugform auf der Werkstückoberfläche

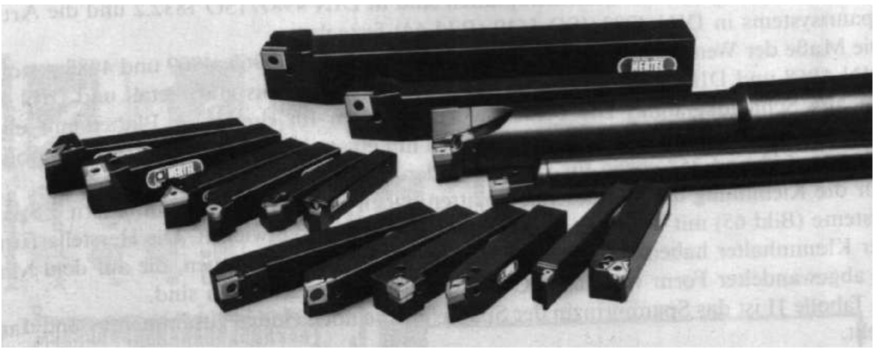


Abbildung 3.33: Klemmhalter für verschiedene Formen und Größen von Wendeschneidplatten

Abbildung 3.34: Formen und Größen der Wendeschneidplatten

??) haben Durchmesser von 30 - 100 mm. Auch hier bleibt beim Nachschleifen das Gewindeprofil erhalten. Eine solche Formscheibe kann bis auf $\frac{1}{4}$ ihres Umfangs nachgeschliffen werden. Beim Gewindedrehen mit der Formdrehscheibe muß die Mitte der Formscheibe um das Maß x über der Werkstückmitte stehen. Dies ist notwendig, damit der Formdrehmeißel frei schneiden kann. Der Freiwinkel α ergibt sich aus der Tangente an der Rundung der Formscheibe beim Berührpunkt mit dem Werkstück und der Vertikalen. Der erforderliche Höhenversatz x läßt sich rechnerisch bestimmen:

$$x = \frac{D}{2} \cdot \sin \alpha \tag{3.13}$$

Innengewinde

Der Innengewindedrehmeißel ist ein abgebogener Formdrehmeißel. Er kann aus Schnellarbeitsstahl oder auch als Hartmetallwerkzeug ausgeführt sein. Auch hier setzt sich der Klemmhalter mit speziellen Vollprofilplatten für Gewinde immer mehr durch (Abb. ??.a).

Gewindestrehler (Abb. ??.c) sind mehrschneidige Werkzeuge. Bei ihnen wird die Zerspanungsarbeit beim Gewindeschneiden auf mehrere Schneiden aufgeteilt. Deshalb kann das zu erzeugende Gewinde mit dem mehrschneidigen Gewindestrehler in einem Durchgang erzeugt werden. Der Innengewindestrehler kann aber auch als mehrschneidiger Formscheibendrehmeißel ausgebildet sein.

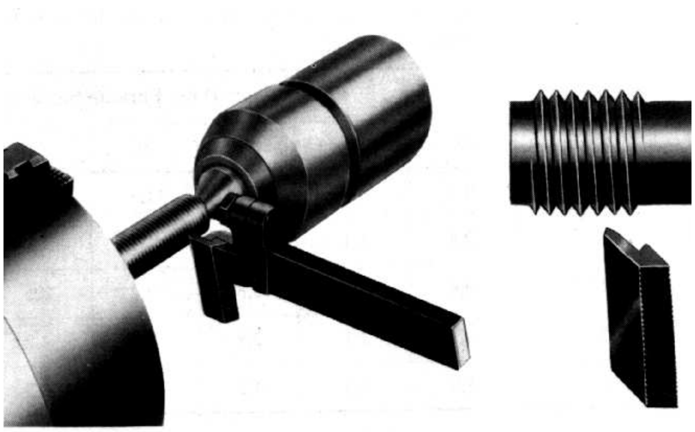


Abbildung 3.35: Drehmeißelhalter mit eingesetztem Formmesser

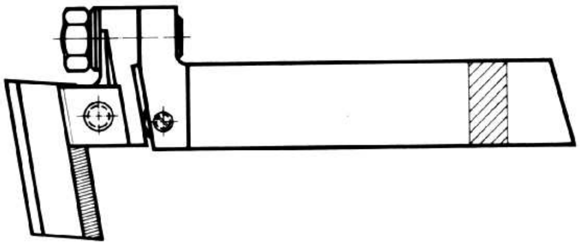


Abbildung 3.36: Seitenansicht des Halters mit Messer

3.5.3 Formdrehmeißel

Formdrehmeißel zur Herstellung beliebiger Werkstückformen zeigt Abb. ??.

3.5.4 Werkzeuge zum Kopieren

Drehmeißel zum Kopieren werden mit Eckwinkeln von 55^o oder 60^o ausgeführt. Überwiegend setzt man dafür Klemmhalter mit Spanformern ein, bei denen die Spanformstufenbreite variiert werden kann. Die in Abbildung ?? gezeigten WIDAX-Halter SKP haben rhombische Schneidplatten mit einem positiven Spanwinkel von 6^o.

Diese Halter sind für alle Kopierverfahren (Längs-, Innen- und Außenkopieren) sowie für das entsprechende Konturdrehen auf NC-Drehmaschinen geeignet.

3.6 Die Standzeit T

Definition: Die Standzeit T ist die Zeit in Minuten, in der die Schneide unter dem Einfluß der Zerspanungsvorgänge zwischen zwei Anschliffen arbeitsfähig bleibt.

Arbeitsfähig ist die Schneide, bis eine bestimmte Verschleißgröße erreicht ist.

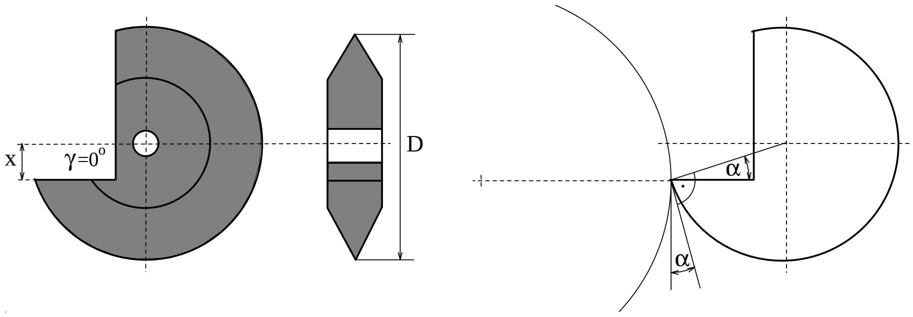


Abbildung 3.37: Scheibendrehmeißel

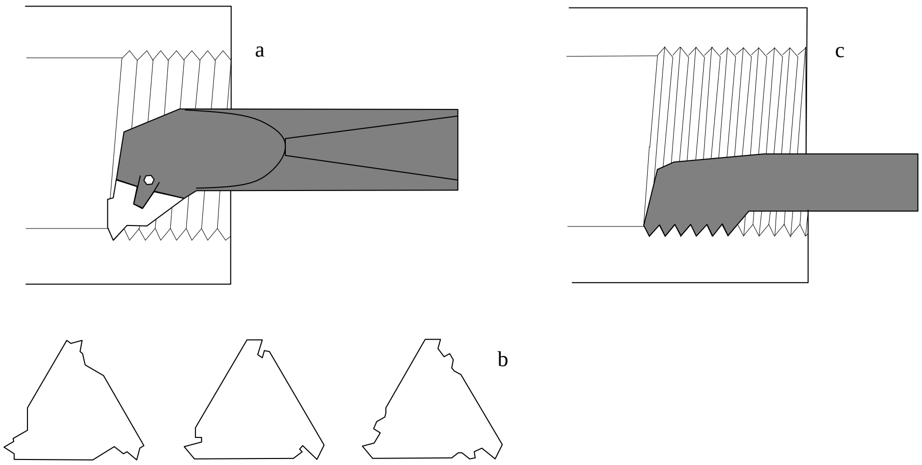


Abbildung 3.38: a Klemmhalter für das Drehen von Innengewinden, b Vollprofilplatten für Innengewinde, c Gewindestrehler

3.6.1 Ursachen für die Abstumpfung

Temperaturbedingte Abstumpfung

Wenn bei einem Werkzeug die sog. Erliegungstemperatur erreicht ist (Werkzeugstahl 300°C, Schnellstahl 600°C), kommt es zum Abschmelzen und Ausbrechen der Schneide. Eine nicht mehr arbeitsfähige Schneide erzeugt auf dem Werkstück einen glänzenden Streifen. Diese Erscheinung bezeichnet man als Blankbremsung. Der Streifen entsteht, wenn die Schneidkante abgeschmolzen ist und die Freifläche des Werkzeugs über die Schnittfläche des Werkzeugs reibt.

Abstumpfung durch Abrieb

Bei Hartmetallen und Schneidkeramiken gibt es keine ausgeprägte Erliegungstemperatur; hier nimmt der Verschleiß anfangs rasch, später langsam zu.

3.6.2 Arten des Verschleißes

Freiflächenverschleiß

Hier wird der Verschleiß an der Freifläche gemessen (Abb. ??). Das Werkzeug gilt als stumpf, wenn eine bestimmte Verschleißmarkenbreite B erreicht ist. Je größer B, um so größer ist der

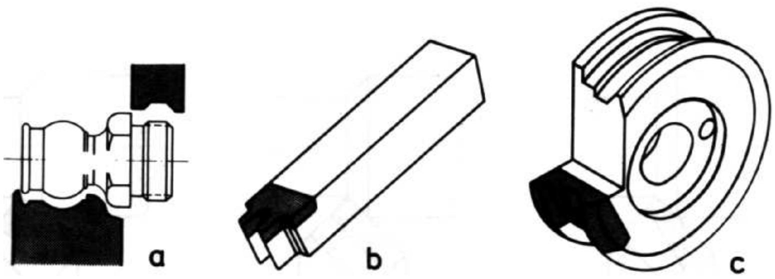


Abbildung 3.39: a mit Formdrehmeißel erzeugtes Werkstück, b Formdrehmeißel mit Hartmetallplatte, c Formdrehscheibe

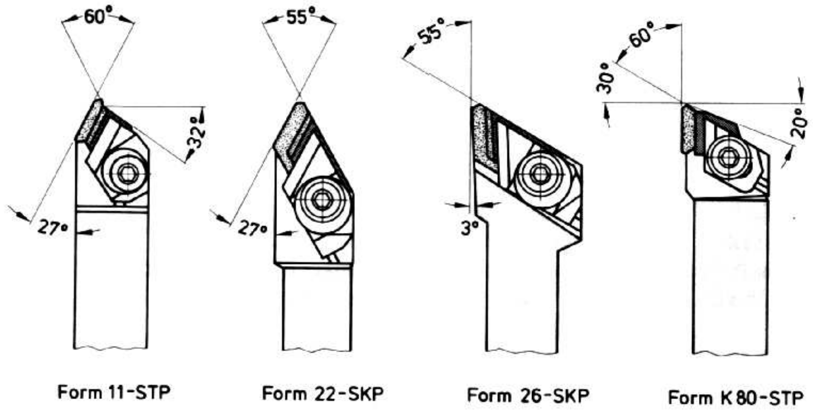


Abbildung 3.40: Klemmhalter zum Kopierdrehen für Formen mit positivem Spanwinkel

Schneidkantenversatz. Die folgende Tabelle zeigt die zulässigen Verschleißmarkenbreiten für Drehverfahren.

Verfahren	B in mm
Feindrehen	0,2
Schlichtdrehen	0,3-0,4
Schruppdrehen mittlere Spanquerschnitte große Spanquerschnitte	 0,6-0,8 1,0-1,5

Kolkverschleiß

Hier werden als Verschleißmerkmale die Kolktiefe K_T, die Kolkbreite K_B und der Kolkmittenabstand K_M gemessen. Aus der Kolktiefe und dem Kolkmittenabstand wird die Kolkkennzahl K bestimmt:

$$K = \frac{K_T}{K_M} \tag{3.14}$$

K		Kolkkennzahl
K_T	in mm	Kolktiefe
K_M	in mm	Kolkmittenabstand

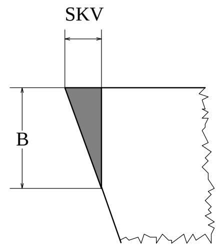


Abbildung 3.41: Freiflächenverschleiß mit der Verschleißmarkenbreite B, Schneidkantenversatz SKV

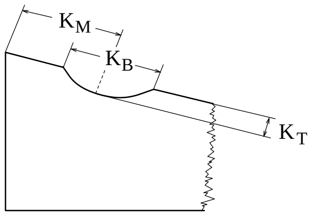


Abbildung 3.42: Spanflächenverschleiß mit der Kolktiefe K_T und dem Kolkmittenabstand K_M

Die Kolkkennzahl ist ein Maß für die Schwächung des Schneidkeils und darf deshalb einen bestimmten Grenzwert nicht überschreiten. Je nach zu zerspanendem Werkstoff und je nach Schneidwerkstoff liegen die zulässigen Kolkkennzahlen zwischen 0,1 und 0,3.

Bei größeren Schnittgeschwindigkeiten überwiegt der Kolkverschleiß. Deshalb sollte dieses Verschleißkriterium bevorzugt im Bereich von hohen Schnittgeschwindigkeiten ($v > 150$ m/min) angewandt werden. In der Praxis wird jedoch überwiegend mit der Verschleißmarkenbreite als Verschleißkriterium gearbeitet.

Außer den Verschleißmerkmalen am Werkzeug und am Werkstück ist zu beachten, daß mit zunehmendem Verschleiß auch die Zerspankräfte und die zum Zerspanen erforderliche Antriebsleistung ansteigt. Auch aus diesem Grund darf der Verschleiß an den Werkzeugen die oben genannten Grenzen nicht überschreiten.

3.6.3 Einflüsse auf die Standzeit

Die Standzeit T der Zerspanwerkzeuge ist von vielen Faktoren abhängig. Die wichtigsten sind:

- Werkstückwerkstoff
  Je größer die Verfestigung beim Stauchen des Spans ist, umso größer sind die Kräfte, die auf die Schneide wirken. Mit wachsender Pressung sowie Druck- und Biegebeanspruchung nimmt die Standzeit ab.

- Schneidenstoff
  Das Verschleißverhalten der Schneidstoffe ist haupsächlich abhängig von ihrer Härte, der Druck- und Biegefestigkeit, der Temperaturbeständigkeit und der Zähigkeit. Zunehmende Härte verringert den Abrieb. Große Druck- und Biegefestigkeit, insbesondere bei höheren Temperaturen, verbessern die Kantenfestigkeit. Je größer die kritische Temperatur, bei der z.B. Schneiden aus Schnellarbeitsstahl erliegen oder Schneiden aus Hartmetall zerbröckeln, desdo mehr Reibungswärme kann der Schneidstoff vertragen, umso größer wird also die zulässige Schnittgeschwindigkeit. Zähe Werkstoffe widerstehen stoßartiger oder schwingender Belastung besser als spröde.

- Schneidenform
  Bei großem Keilwinkel und kleinem Spanwinkel wird der beanspruchte Querschnitt der Schneide größer, die übertragbaren Kräfte wachsen entsprechend, der Verschleiß wird kleiner sein als bei schlanken und spitzen Schneiden.

- Oberfläche
  Harte und ungleichmäßige Werkstückflächen, z.B. mit Gußkante oder Schmiedehaut, rufen stoßartige oder schwingende Belastungen der Schneide hervor und verringern bei spröden Schneidstoffen die Standzeit

- Steife
  Labile Werkstücke, Spannvorrichtungen oder Werkzeuge setzen die Rattergrenze herab, gefährden also spröde Schneidstoffe.

- Spanquerschnitt
  Mit wachsendem Spanquerschnitt wächst die Schnittkraft und damit die Schneidenbelastung an. Der Vorschub beeinflußt dabei den Verschleiß stärker als die Zustellung.

- Kühlschmiermittel
  Kühlschmiermittel haben je nach Zusammensetzung eine mehr schmierende oder mehr kühlende Wirkung. Bei niedrigen Schnittgeschwindigkeiten kann durch überwiegende Schmierung, bei großen Schnittgeschwindigkeiten durch überwiegende Kühlung die Standzeit verbessert werden.

- Schnittgeschwindigkeit
  Die Schnittgeschwindigkeit beeinflußt die Standzeit T am stärksten. Die Abhängigkeit der Standzeit von der Schnittgeschwindigkeit wird in Standzeitkurven gezeigt. Daraus folgt, daß die Standzeit mit zunehmender Schnittgeschwindigkeit stark abfällt.

3.6.4 Berechnung der Standzeit

Die Standzeit läßt sich rechnerisch nach folgender Taylor-Gleichung bestimmen:

$$T = \frac{1}{C^k} \cdot \nu^k \tag{3.15}$$

T	in min	Standzeit
C	in m/min	Schnittgeschwindigkeit für $T = 1$ min
k		Konstante

Die Standzeitkurve ist also eine Exponentialfunktion. Daraus geht hervor, daß mit wachsender Schnittgeschwindigkeit die Standzeit stark abfällt.

Da der Vorschub f und die Schnittiefe a_p ebenfalls Einfluß auf die Werkzeugstandzeit haben, arbeitet man zur Ermittlung der Schnittgeschwindigkeit $v_{c_{111}}$ mit der erweiterten Taylor-Gleichung

$$v_{c_{111}} = C \cdot f^{\mathrm{E}} \cdot a_p^{\mathrm{F}} \cdot T^{\mathrm{G}} \qquad \left(\mathrm{G} = \frac{1}{\mathrm{K}}\right) \tag{3.16}$$

$v_{c_{111}}$	in m/min	Schnittgeschwindigkeit für $f = 1$ mm/U, $a_p = 1$ mm, $T = 1$ min
f	in mm	Vorschub
a_p	in mm	Schnittiefe
E,F,G		Schneidstoffkonstanten
K		Kolkkennzahl

Die $v_{c_{111}}$-Werte und Zahlenwerte für die Exponenten E, F und G werden in Richtwerttabellen der Hartmetallhersteller angegeben.

3.6.5 Kostengünstigste Standzeit

Mit der Entwicklung der Wendeschneidplatten, die in den dafür entwickelten Klemmstahlhaltern sehr schnell ausgewechselt werden können, ergeben sich neue optimale Standzeitwerte. So geht man in vielen Fällen, zugunsten einer höheren Schnittgeschwindigkeit und einer kleineren Hauptzeit, von dem früheren Normwert $T = 60$ min ab. Man nimmt rechnerisch unter Berücksichtigung der Werkzeugkosten, des Maschinenstundensatzes, der Lohnkosten, der Werkzeugpositionierzeit und des zu zerspanenden Werkstoffs die kostengünstigste Standzeit. Dabei ergeben sich Standzeitwerte von $T = 5$ - 30 min mit zugeordneten Schnittgeschwindigkeiten von 200 - 400 m/min.

3.7 Spanvolumen und Spanraumzahl

3.7.1 Spanvolumen

Beim Spanvolumen ist zwischen dem Werkstoffvolumen Q_W und dem Raumbedarf der ungeordneten Spanmenge Q_{SP} zu unterscheiden. Das Werkstoffvolumen ist das Volumen, das ein Span mit dem Querschnitt $a_p \cdot f$ und definierter Länge pro Minute einnimmt.

$$Q_W = a_p \cdot f \cdot v_c \cdot 10^3 \tag{3.17}$$

Q_W	in mm^3/min	Werkstoffvolumen
a_p	in mm	Schnittiefe
f	in mm	Vorschub
v_c	in m/min	Schnittgeschwindigkeit
10^3	in mm/m	

Das Volumen der ungeordneten Spanmenge Q_{SP} ist größer als das tatsächliche Werkstoffvolumen Q_W der gleichen Spanmenge, weil sich ja in einem Behältnis die Späne nicht lückenlos aneinanderfügen. Um wieviel größer das Volumen der ungeordneten Spanmenge Q_{SP} als das Werkstoffvolumen Q_W ist, gibt die Spanraumzahl R an.

$$Q_{SP} = R \cdot Q_W \tag{3.18}$$

Q_{SP}	in mm^3/min	Volumen der ungeordneten Spanmenge
Q_W	in mm^3/min	Werkstoffvolumen
R		Spanraumzahl

Die Größe der Spanraumzahl R ist von der Spanform abhängig.

3.7.2 Spanformen

Die sich beim Zerspanen ergebende Form der Späne ist abhängig

- von der Art und Legierung des Werkstückwerkstoffs
- vom Phosphor- und Schwefelgehalt des Werkstoffs
- von den Schnittbedingungen (Schnittgeschwindigkeit, Schnittiefe, Vorschub, Einstellwinkel)
- vom Spanwinkel und der Ausbildung der Spanformstufe

Die Beurteilung der Spanformen erfolgt nach zwei Kriterien:

Transportfähigkeit

Kurz gebrochene Späne lassen sich leicht in Behältnissen transportieren. Im Gegensatz dazu ist dies mit Bandspänen nicht möglich (Abb. ??). Sie erfordern immer eine bestimmte Aufbereitung (Brechen im Spänebrecher oder Paketieren), damit sie transportfähig werden. Solche Aufbereitungen verursachen in einem Automatenbetrieb, in dem viele Späne anfallen, hohe Kosten. Deshalb strebt man immer Spanformen an, die gut transportabel sind.

Gefahr für den Menschen an der Maschine

Bestimmte Spanformen, z.B. lange Bandspäne oder wirre Wendelspäne (Abb. ??), deren Kanten messerscharf sind, gefährden den Menschen an der Maschine.

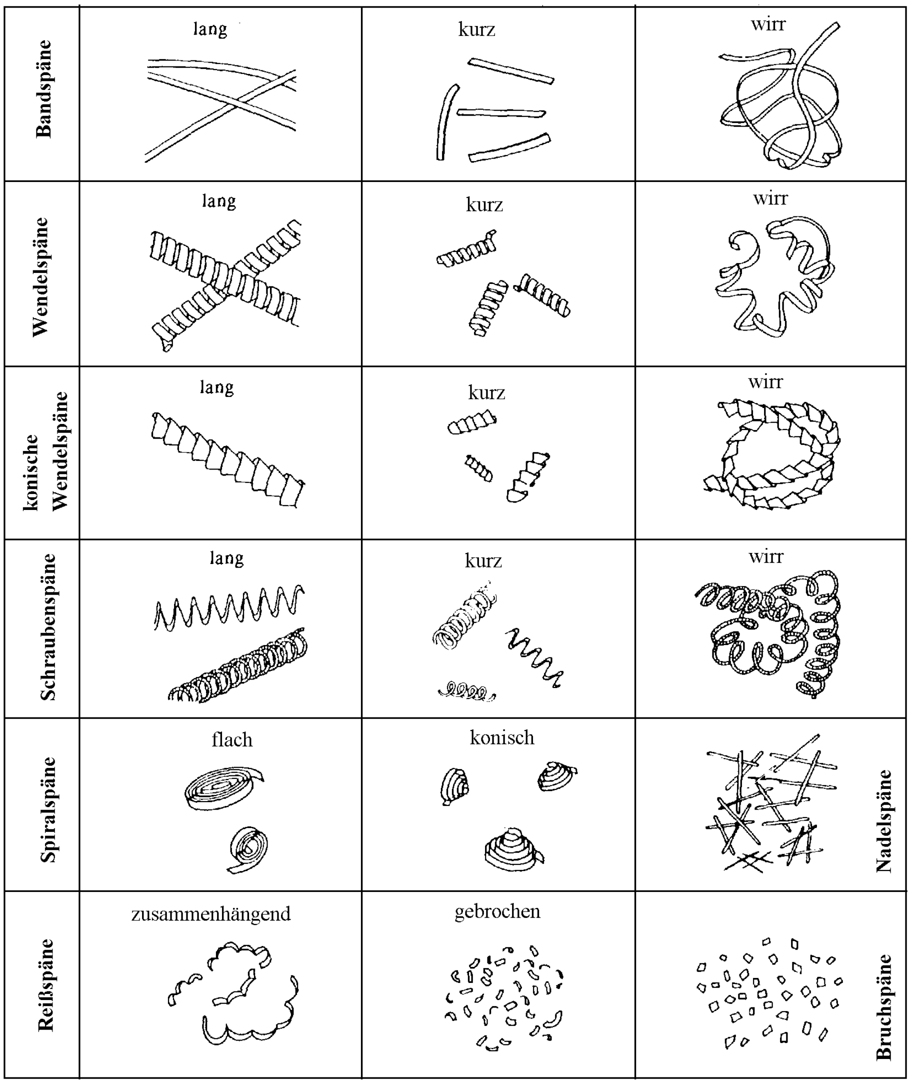


Abbildung 3.43: Spanformen beim Drehen

Kapitel 4

Beschreibung des Planungssystems

In diesem Kapitel werden zunächst grundlegende Aussagen gemacht über die Repräsentation der Werkstücke; im Anhang ?? werden die implementierten Objekte zur Darstellung der Werkstücke vorgestellt.

Daneben wird ein Überblick gegeben über den prinzipiellen Ablauf des Planungssystems, dessen verschiedene Phasen in Kapitel ?? einzeln vorgestellt werden.

4.1 Datenstrukturen zur Repräsentation der Werkstücke

Um eine einheitliche Darstellung rotationssymmetrischer Werkstücke im Rahmen von CABPLAN zu ermöglichen, wurde die Entwicklung einer geeigneten Repräsentationsform zum Gegenstand einer eigenen Diplomarbeit [13], in deren Rahmen unter anderem auch die Entwicklung des *Workpiece–Designers* als Anwendung von **GraMoD**[1] fällt. Hieran erkennt man den Stellenwert, der dieser Repräsentation eingeräumt wird. Folgende Forderungen werden an die Datenstrukturen im Rahmen eines größeren Projektes gestellt:

- Allgemeine und universelle Darstellung der Werkstücke
  um möglichst alle zu erwartenden Forderungen abzudecken und in möglichst vielen verschiedenen, aber gleichgearteten Projekten eingesetzt werden zu können
- Spezialisierung
  auf rotationssymmetrische Drehteile mit deren speziellen Features
- Redundanzfreiheit
- Flexibilität
  um auf erweiterte oder geänderte Anforderungen angepaßt werden zu können

Weil die Programmierung des CABPLAN–Projektes in Smalltalk-80 erfolgt, sollte die Werkstückrepräsentation im Rahmen der objektorientierten Programmierung geeignete Klassen zur Verfügung stellen, die die oben genannten Kriterien unterstützen.

Die vorliegende Arbeit benutzt die von [13] entwickelten Datenstrukturen zur Werkstückrepräsentation, die entsprechend erweitert wurden.

Grundsätzlich wird ein Werkstück als Verkettung von Grundelementen dargestellt. Diese primitiven Komponenten sind durch Verbindungselemente untereinander verknüpft. Die „normalen“ inneren Verbindungselemente repräsentieren die Planflächen des Werkstücks, während zwei spezielle äußere Verbindungselemente die Stirnseiten darstellen. Abbildung ?? zeigt ein einfaches

[1] Graphischer Modellierer für Drehteile

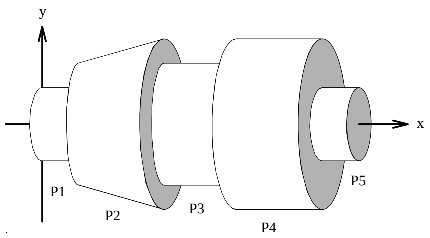


Abbildung 4.1: Werkstück aus primitiven Komponenten im Koordinatensystem

Beispiel eines Drehteils, wofür in ?? eine mögliche Darstellung mit Verbindungselementen gegeben wird.

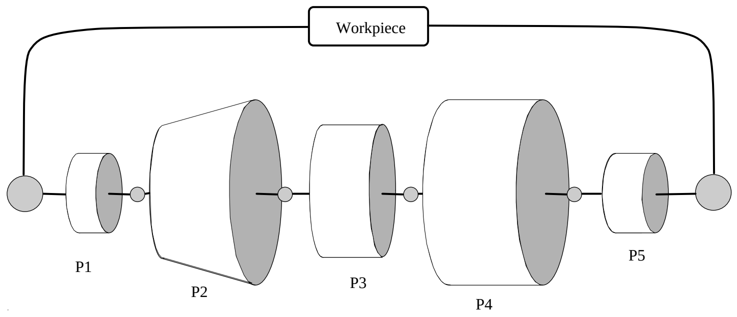


Abbildung 4.2: Darstellung als Kette aus Grundelementen und Verbindungselementen

Mehrere primitive Komponenten lassen sich zu komplexen Komponenten zusammenfassen. Ihre Darstellung soll prinzipiell genauso aussehen wie die der primitiven Komponenten, um so eine flexible Strukturierung zu ermöglichen (Abb. ??). Die komplexen Komponenten können wiederum aus zusammengesetzten Strukturen bestehen, so daß letztendlich eine Darstellung des Werkstücks in einer Baumstruktur ermöglicht wird.

Um nun eine einheitliche Darstellung aller Werkstücke zu erreichen, wird jedes Drehteil dargestellt als eine komplexe (Super-) Komponente mit ihren zwei Stirnseiten, ganz gleich ob das Werkstück aus einer oder vielen primitiven Komponenten besteht (Abb. ??). Diese Festlegung ist zur Vereinheitlichung der Methoden der einzelnen Planungskomponenten notwendig.

Weitere Bearbeitungselemente auf der Oberfläche des Werkstücks, wie z.B. Nuten, Gewinde oder Freistiche, werden den primitiven Bearbeitungselementen zugewiesen, auf denen sie ansetzen. So kann man etwa an einer Stirnseite eine Bohrung anbringen, oder eine Zylinderfläche mit einem Gewinde versehen. Die Möglichkeit, primitive Komponenten zu komplexen zusammenzufassen, hilft dem Planungssystem, bei der Bearbeitung zusammengehörige Bearbeitungsbereiche mit den gegebenen Repräsentationsformen auszudrücken; das Planungssystem greift sogar auf die von der CAD-Komponente übergebene Instanzenvariable des Werkstücks zu und strukturiert diese zu einer Baumstruktur um, die zum Anwenden der Mapping–Algorithmen geeignet ist.

Die geometrische Darstellung eines Werkstücks erfolgt stets von links nach rechts; der Ursprung seines Koordinatensystems liegt im Mittelpunkt der linken Stirnseite, und die positive x–Achse wird durch die Symmetrieachse des Werkstücks gebildet. Alle Längenangaben erfolgen in Millimetern, die Winkelangaben im Gradmaß.

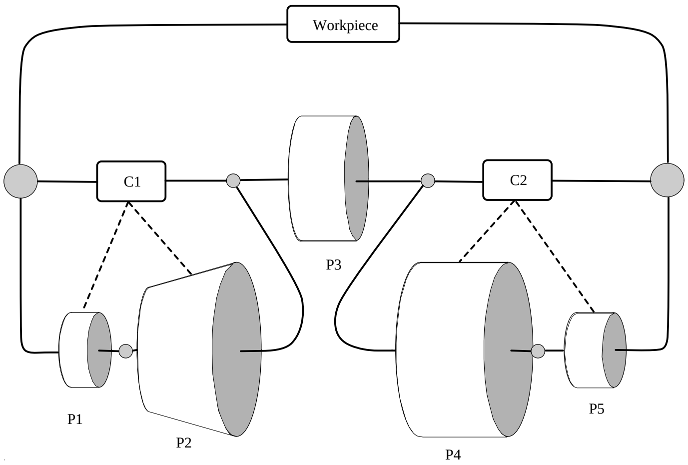


Abbildung 4.3: Verwendung von komplexen Komponenten

4.2 Die Planungskomponenten

In diesem Abschnitt wird die prinzipielle Vorgehensweise des Planungssystems zum Finden eines geeigneten Falles, der zur Lösung der Planungsaufgabe bzw. zur Steuerung der weiteren Planungsstrategie geeignet ist, beschrieben; die einzelnen Komponenten werden in Kapitel ?? vorgestellt.

4.2.1 Prinzipielle Planungsstrategie

Die Retrievalphase von CAbPlan dient dazu, aus der Fallbasis einen Fall auszuwählen, dessen Plan zur weiteren Steuerung des generativen Planungssystems dient. Sie läßt sich prinzipiell in drei Phasen unterteilen:

- Die Analysephase
  Die Analysephase dient dazu, das Werkstück anhand seiner Beschreibung in Hinblick auf die zu erreichenden Goals zu untersuchen. Wurden diese Goals spezifiziert, werden sie in generalisierter Form in speziellen Speichervariablen, den sog. **Slots**, abgelegt, die als Parameter dem semantischen Netzwerk übergeben werden. Die Generalisierung erfolgt, um eine Klasse von Fallbeispielen in der Fallbasis zusammenfassen zu können.

  Eine weitere Aufgabe der Analysephase besteht darin, die zum Mapping erforderliche Baumstruktur des Werkstücks zu erstellen.

- Die Indizierung der Fallbasis über ein semantisches Netzwerk
  Eine Indizierung der Fallbasis bewirkt eine Vorauswahl von solchen Fällen, die die in den Slots abgelegten Goals (genauer gesagt deren generalisierte Form in den Slots) erfüllen. Man kann diese Phase als Auswertung der oberflächlichen Ähnlichkeit (*surface similarity*) zwischen Werkstücken auffassen, weil die Goals direkt aus den geometrischen Spezifizierungen des Werkstücks heraus entstanden sind und eine Ähnlichkeitsfindung aufgrund hieraus abgeleiteter Attribute, den Slots, über deren Werte erfolgt. Es erfolgt ein strukturierter Zugriff auf die Fallbasis sowohl beim Retrieval als auch beim Abspeichern der vom Planungssystem erzeugten Pläne. Dies hat zur Folge, daß nur solche Fälle in Betracht gezogen werden, die aufgrund von Attributwerten „ähnlich“ zu dem Werkstück der Aufgabenstellung sind.

- Das Mapping der indizierten Fälle
  Die durch die Indizierung gefundenen Fälle werden nun nacheinander gegen das Eingabewerkstück gemappt; mit Hilfe der von der Analysephase erzeugten Baumstruktur wird so für

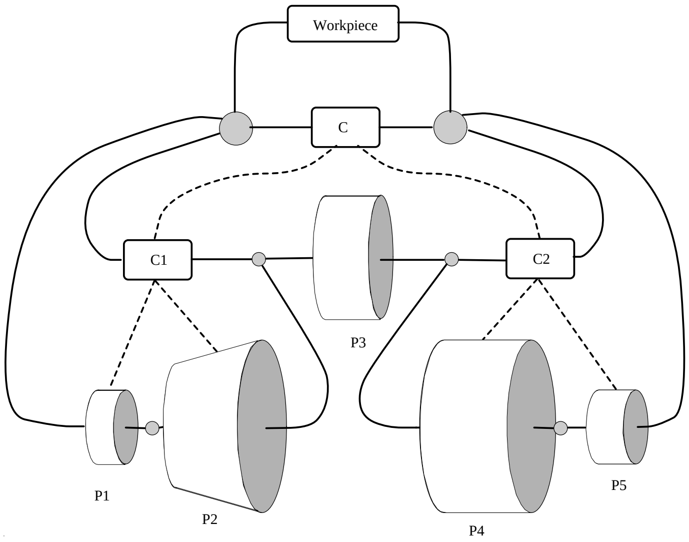


Abbildung 4.4: Werkstückstruktur mit umfassender Komponente C

eine Teilmenge der vorhandenen Fälle die strukturelle Ähnlichkeit (*deep similarity*) zwischen Werkstücken bestimmt, deren Auswertung in der Regel wegen den verwendeten Algorithmen längere Laufzeiten nach sich ziehen. Als Ergebnis wird der Plan des Werkstücks geliefert, das den geringsten Abstand zum Problem hat.

Prinzipiell lassen sich die drei Phasen als unabhängig voneinander betrachten, so daß ein Wiedereinstieg in den Retrievalablauf durch andere Planungskomponenten unterstützt wird.

4.2.2 Goals zur Indizierung von Fällen

Die wichtigste Grundlage eines fallbasierten Planers ist sein Erinnerungsvermögen an erfolgreiche Problemlösungen. Diese Erinnerung wird sowohl benutzt, um die aktuellen Ziele an den fertigen Plan anzupassen als auch, um erfolgreiche Pläne zum späteren Gebrauch abzuspeichern. Ohne diese Planorganisation hätte der fallbasierte Planer keinen Ansatzpunkt, um die Suche nach geeigneten Plänen aus der Plansammlung aufzunehmen oder die Resultate der Planung abzuspeichern.

Es gibt zwei wesentliche Gesichtspunkte, eine Plansammlung aufzubauen: Die Form der Objekte, die gespeichert werden sollen, und das Vokabular, um sie abzuspeichern. Bei einem fallbasierten Planer sind die Objekte, die abgespeichert werden sollen, eher konkrete Pläne als hiervon abstrahierte Strukturen. Das Vokabular, das benutzt wird, um diese Pläne im Planspeicher zu indizieren, ist das Vokabular der Ziele (*goals*), die der Plan erfüllt und der Probleme, die er vermeidet.

Ein fallbasierter Planer speichert spezielle Pläne für spezielle Anwendungssituationen, anstatt verallgemeinerte Versionen in die Plansammlung aufzunehmen. Das bedeutet aber nicht, daß er ohne Generalisierung auskommt. Die Verallgemeinerung wird vielmehr auf der Stufe der Indizierung durch spezielle Features vorgenommen und nicht auf der Stufe der konkreten Pläne selbst. Dadurch, daß die Pläne durch allgemeine Beschreibungen der Ziele, die sie erfüllen, als auch durch die spezifischen Ziele selbst indiziert werden, können Pläne auch für solche Situationen ausgewählt werden, in denen sie aktuelle Ziele nur teilweise erfüllen. Dies führt dazu, daß der Planer nach der Modifikation einzelne Pläne in einer Vielzahl von Anwendungem benutzen kann. Gleichzeitig wird die Spezifität der Pläne gewährleistet, die dazu benutzt wird, um in solchen Situationen, in denen die gegebenen Ziele genau mit den geforderten übereinstimmen, diese direkt zu übernehmen.

Die erste Komponente der Speicherorganisation eines fallbasierten Planers sind die Pläne selbst. Diese Pläne sind eine geordnete Menge von Schritten auf der Stufe der primitiven Aktionen des

Planers. Bei CAbPlan sind diese Pläne die Fertigungspläne für das gewünschte Werkstück, also die Aufspannungen auf den jeweiligen Spannbereichen mit ihrer Reihenfolge, die benötigten Werkzeuge und die Bearbeitungsbereiche mit den dazugehörigen Bearbeitungsaktionen. Es sind sowohl diejenigen Pläne, die dem System von einem Experten als Fallbeispiele eingegeben werden als auch die, die es selbst erzeugt hat und die in die Plansammlung übernommen wurden. Beim CHEF–System entsprechen die Pläne den Kochrezepten.

Die Darstellung der Pläne im Initialzustand des Systems ist demzufolge identisch mit der Darstellung der Pläne, die es erzeugt.

Jeder Plan, den ein fallbasierter Planer in seinem Gedächtnis hat, ist mit einer Menge von Zielen assoziiert, die er aus allgemeinen Regeln und dem Initialzustand der Planungsaufgabe (= die Beschreibung des Werkstücks) erzeugt. Dazu wird die Werkstückbeschreibung zusätzlich um eine Menge von Slots erweitert, die aber alle aus der eingegebenen Beschreibung des Werkstücks ableitbar sind. Die Eingabe an das System besteht aus folgenden Punkten:

1. Geometrie des Werkstücks
2. Features des Werkstücks
3. Oberflächengüten und Toleranzen

Mit Hilfe dieser Informationen werden nun Slots gefüllt, die dazu dienen, die Ziele zu erkennen, die dem Problem zugeordnet sind. Das Wissen über die Planungsziele ist in semantischen Netzen angeordnet, welche Regelwissen über Spannsituationen, Werkzeugwahl oder Bearbeitungsstrategien beinhalten und über dieses Wissen das eingegebene Problem einer bestimmten Menge von Fällen innerhalb der Fallbasis zuordnen, die dieselben Goals erfüllen. Die Fallbasis selbst besitzt eine s–dimensionale Struktur, wobei s die Anzahl der einzelnen Entscheidungsnetze im Netzwerk ist. Jedes Netz i wird durch einen bestimmten Exit verlassen, wobei die Nummer des Exits der i–te Beitrag zur Indizierung der Fallbasis ist (Abb. ??).

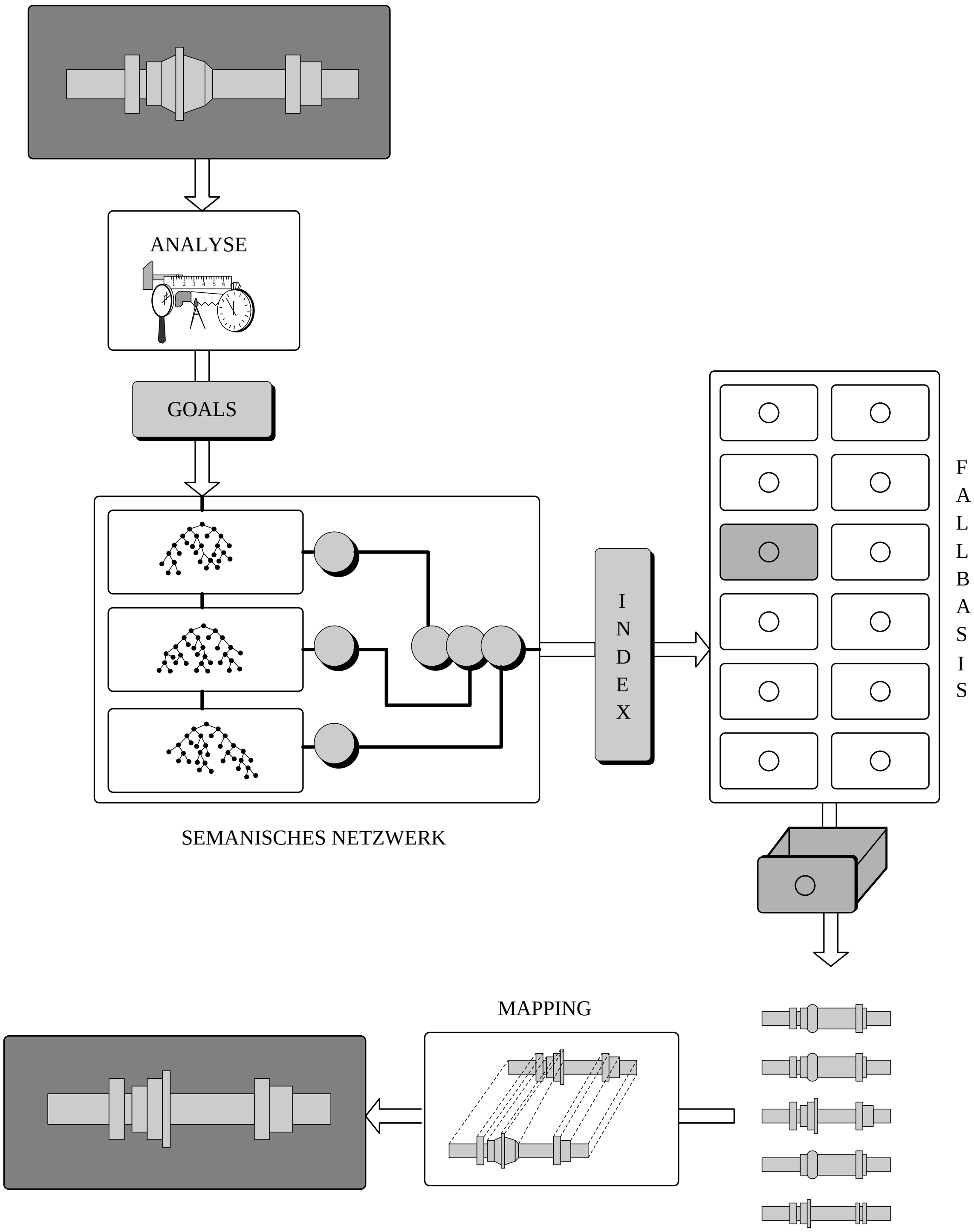


Abbildung 4.5: Ablauf des Retrievals

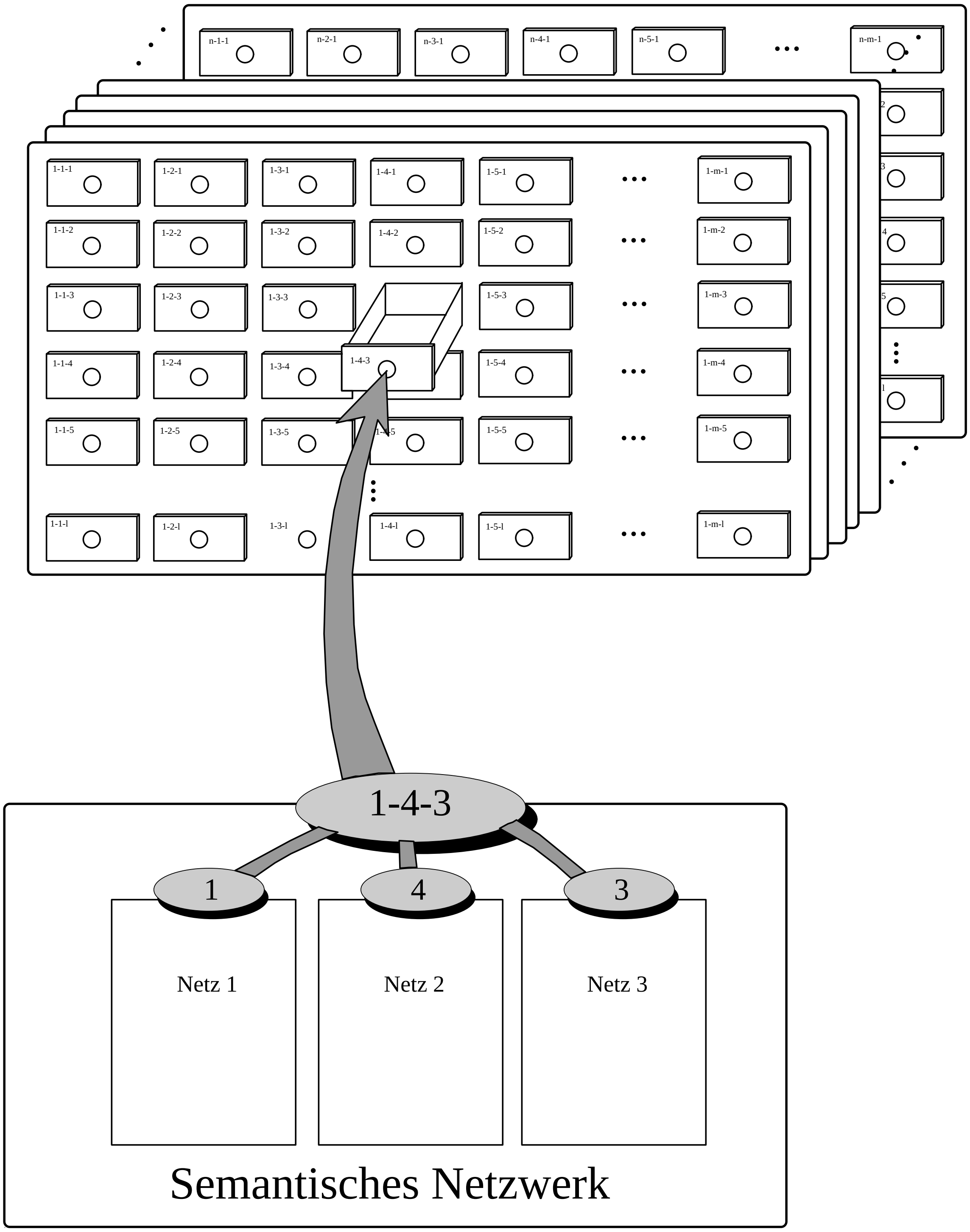


Abbildung 4.6: Indizierung einer 3–dimensionalen Fallbasis durch das semantische Netzwerk

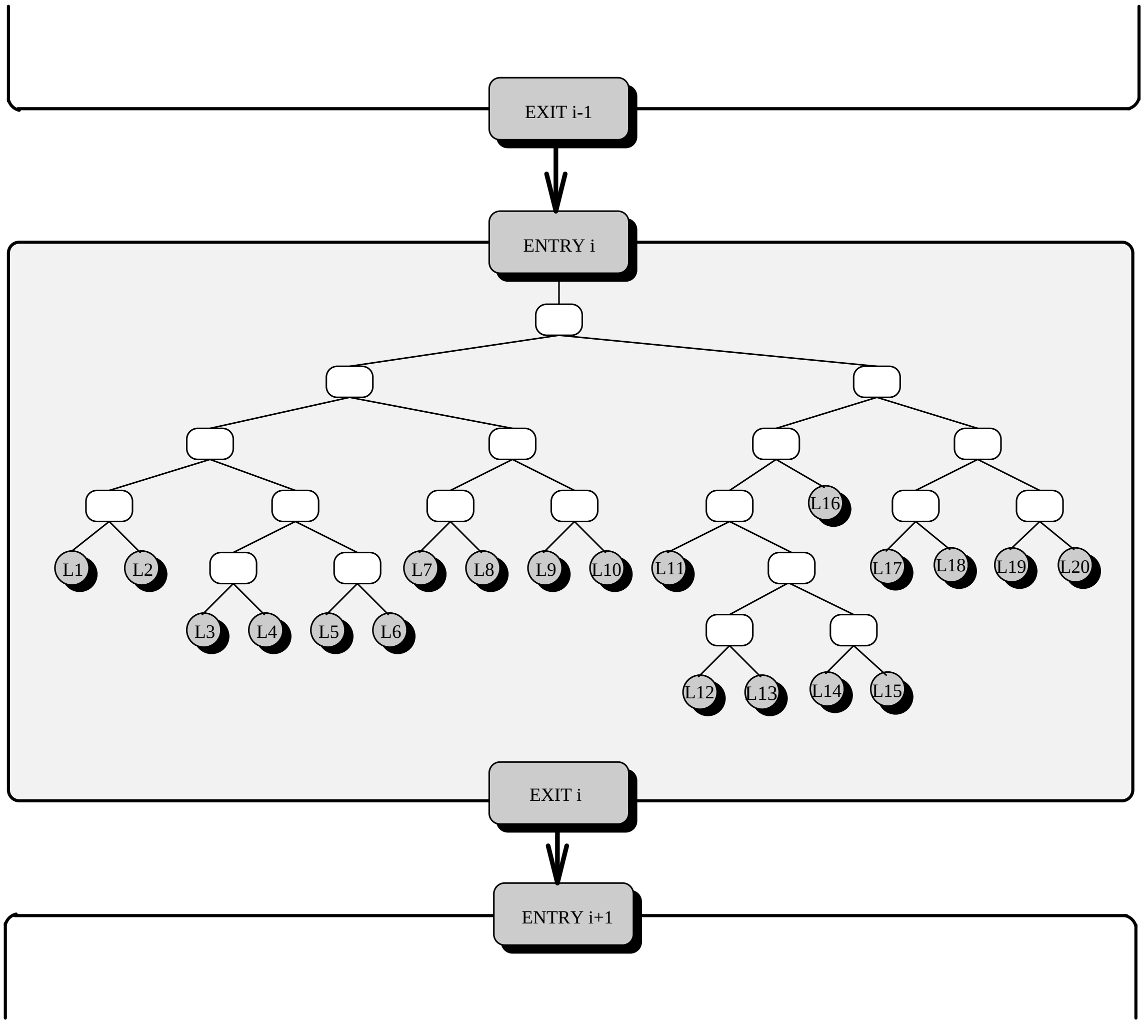


Abbildung 4.7: Semantisches Netz mit Verbindungspointern zu anderen Netzen

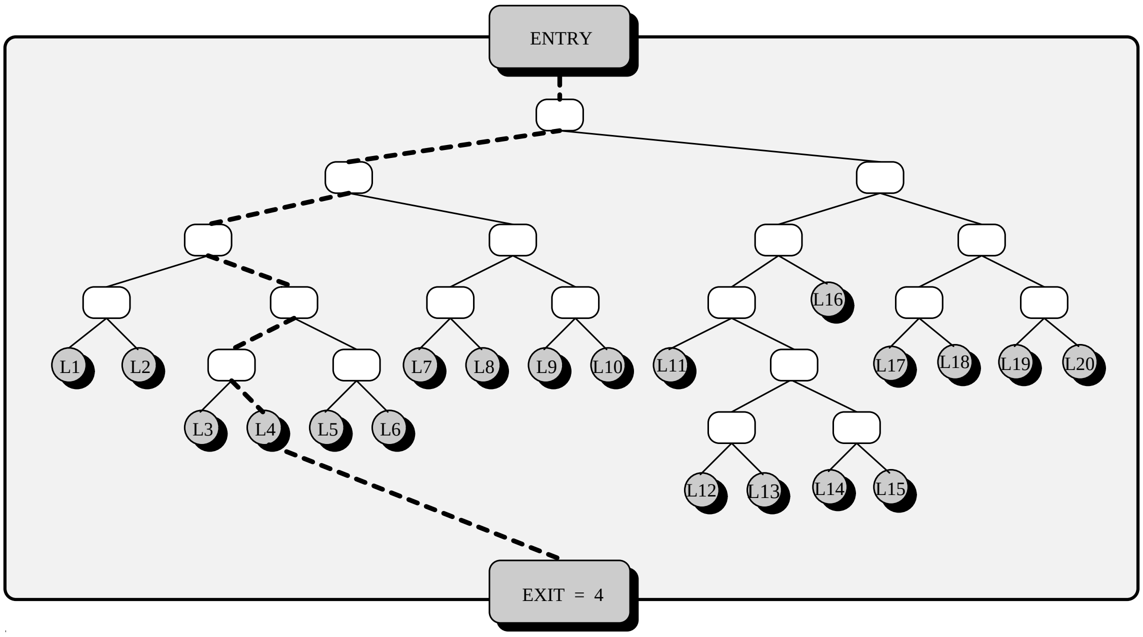


Abbildung 4.8: Ein Durchlauf durch das semantische Netz liefert Exit 4

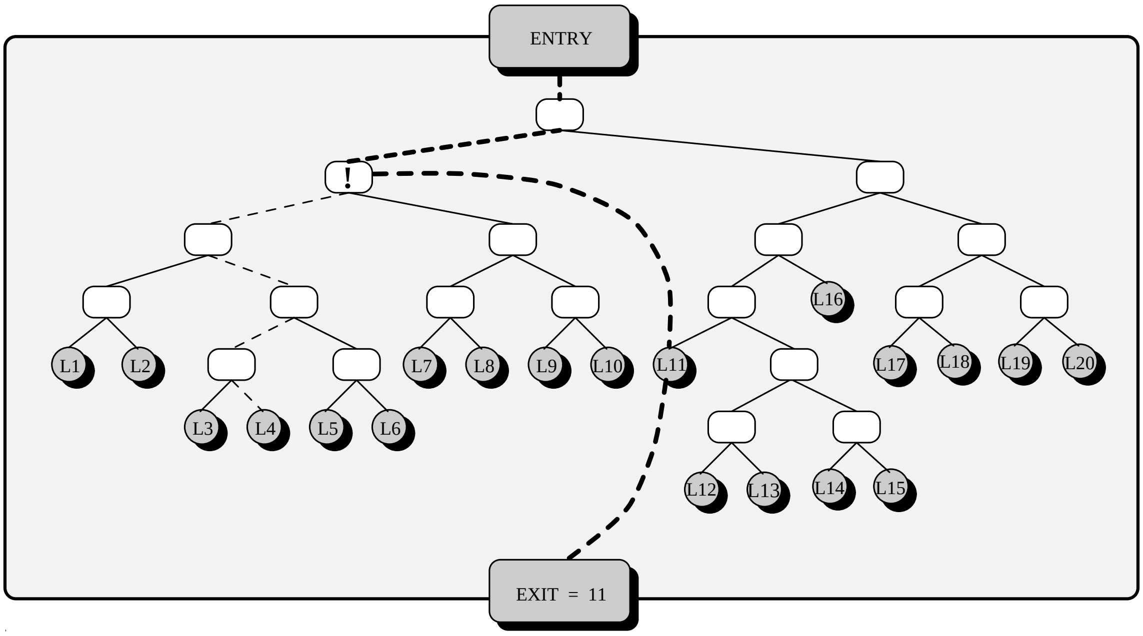


Abbildung 4.9: Umlenkung des Netzdurchlaufs durch eine Anticipator–Komponente liefert Exit 11

Kapitel 5

Planungsphasen

In diesem Kapitel werden die einzelnen Planungsphasen **Analyse**, **Indizierung der Fallbasis** und **Mapping** vorgestellt, deren prinzipielle Arbeitsweisen sowie deren Zusammenwirken in ?? beschrieben wurden. Dabei kann man die beiden letzten Phasen unter dem Begriff des **Retrievers** zusammenfassen, der ein geeignetes Fallbeispiel aus der Plansammlung liefert.

5.1 Die Analysephase

Wie in ?? gesehen, werden in einer Analysephase des Planungssystems die zu erfüllenden Goals anhand der Werkstückbeschreibung festgelegt. Im folgenden werden die Ziele, die betrachtet werden, sowie die Kriterien, nach denen sie berechnet werden, vorgestellt.

5.1.1 Klassifikationsschema

Das hier benutzte Klassifikationsschema wurde speziell vom CIM–Centrum Kaiserslautern entwickelt, um die Aufspannplanung zu unterstützen [21]. Die Zielsetzung bestand konkret darin, eine geeignete Auswahl und Abbildung von Merkmalen eines rotationssymmetrischen Drehteils auf eine Menge der möglichen Aufspannstrategien für dieses Werkstückspektrum zu finden.

Das Klassifikationsschema beinhaltet folgende Eigenschaften:

1. Länge–Durchmesser–Verhältnis

2. Äußere Grundkontur

3. Innere Grundkontur

Diese drei Merkmale werden als unabhängig voneinander betrachtet. Für die Kodierung der einzelnen Merkmalsausprägungen wurde ein numerischer Code wegen seiner Einfachheit gewählt. Es ergibt sich folgender Code–Aufbau:

L/D	äußere Grundkontur	innere Grundkontur

Abbildung 5.1: Bedeutung der einzelnen Stellen des Codes

Kodierung des Länge–Durchmesser–Verhältnisses

Das Länge–Durchmesser–Verhältnis teilt die Gesamtheit des hier betrachteten Werkstückspektrums in drei Klassen: scheibenförmige Werkstücke, wellenförmige Werkstücke und schlanke Wellen. Für die scheibenförmigen Drehteile wird dabei das Länge– Breite–Verhältnis wie folgt definiert: $0 < L/D \leq 1{,}2$ (L Länge des Werkstücks, D maximaler Werkstückdurchmesser). Für wellenförmige Drehteile gilt die folgende Beziehung: $1{,}2 < L/D \leq 6$, und für schlanke Wellen gilt: $6 < L/D \leq 12$. Ein größeres Länge–Breite–Verhältnis wird nicht zugelassen.

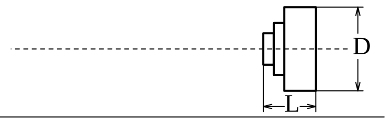
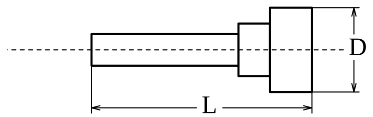
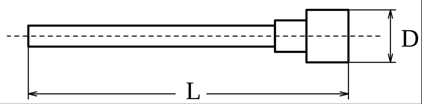
Code	Bezeichnung	Beispiel
1	Scheibe	
2	Welle	
3	schlanke Welle	

Abbildung 5.2: Kodierung des Länge–Durchmesser–Verhältnisses

Kodierung der äußeren Grundstruktur

Das zweitwichtigste Merkmal zur Klassifizierung eines Werkstücks entsprechend der Aufspannstrategie ist die äußere Grundkontur. Schema ?? beschreibt die Klassifizierung der Merkmalsausprägungen zur Beschreibung der äußeren Grundform eines rotationssymmetrischen Drehteils einschließlich ihrer Zuordnung zu einem Kodierungsschlüssel (siehe auch Kodierungstabelle in Anhang ??).

Kodierung der inneren Grundkontur

Die innere Grundform ist das dritte Merkmal zur Klassifizierung des betrachteten Werkstückspektrums. Das Schema ?? beschreibt die Klassifizierung der Merkmalsausprägungen zur Beschreibung der inneren Grundform (s. auch Kodierungstabelle in Anhang ??).

5.1.2 Außenbearbeitungen

Bei der Ermittlung der Anzahl der Aufspannungen, die nötig sind, das Werkstück herzustellen, muß zunächst einmal betrachtet werden, ob die Bearbeitungsstrecke durchgehend ist, d.h. ob die ganze Konturlänge oder nur eine Seite bearbeitet werden muß. Bei Flanchteilen, die aus einer monoton aufsteigenden Struktur bestehen, kann man unter Umständen mit nur einer Aufspannung auskommen, wenn der maximale Radius mit dem Radius des Rohteils übereinstimmt und die Oberflächengüte ausreichend ist.

5.1.3 Stirnseiten

Es wird überprüft, wieviele der Stirnseiten bearbeitet werden müssen. Dabei muß betrachtet werden, ob die Güte der Abstechbearbeitung ausreicht, ohne daß eine Nachbearbeitung notwendig

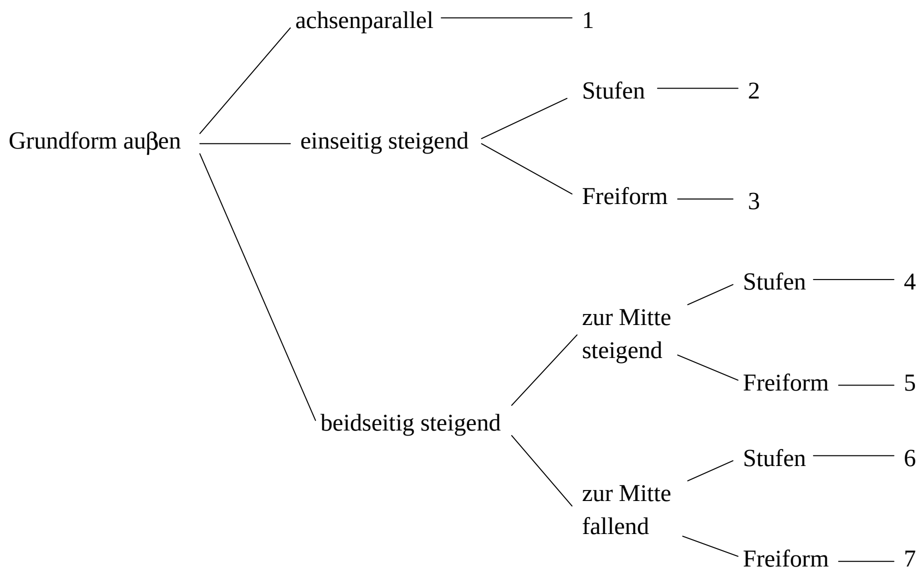


Abbildung 5.3: Struktur der Merkmalsausprägungen der äußeren Grundform

wird. Pro Stirnseite planbearbeiten wird eine Aufspannung nötig, wobei bei der Strategie des Abstechens als letztem Arbeitsschritt unter Umständen die Planbearbeitung einer Stirnseite und somit eine Aufspannung erspart werden kann.

5.1.4 Innenbearbeitungen

Hierbei werden wiederum die Stirnseiten betrachtet, von denen aus eine Innenbearbeitung erfolgen muß. Beim Klassifizierungssystem des Werkstücks wurde im Code bereits festgelegt, ob Innenbearbeitungen erfolgen und ob sie einseitig oder beidseitig sind; hier muß nun weiterhin entschieden werden, ob eine durchgehende Innenbearbeitung gegeben ist und ob diese von einer Stirnseite aus hergestellt werden kann oder ob sie in zwei Aufspannungen von beiden Stirnseiten aus gefertigt werden muß.

5.1.5 Toleranzen

Einfluß der Toleranz auf die Ausspannlänge

Unter der zulässigen Ausspannlänge l zur Einhaltung von Fertigungstoleranzen wird die frei zugängliche Werkstücklänge bei einseitiger Einspannung ohne sonstige Unterstützung bezeichnet. Sie berechnet sich nach:

$$l = 0,45 \cdot d^{\frac{2}{3}} \cdot \sqrt{IT} \qquad (5.1)$$

l	Ausspannlänge
d	mittlerer äußerer Durchmesser
IT	Toleranzfeldhöhe nach DIN 7151

Überschreitet die vorhandene Ausspannlänge die oben errechnete maximal zulässige Ausspannlänge, so sind die geforderten Toleranzen durch die Fertigung in fliegender Einspannung nicht einzuhalten. Erforderlich sind zusätzliche Einspannungen oder Abstützungen. Die Ausspannlänge wird als zusätzliches Prüfkriterium vorwiegend für Länge–Durchmesser–Verhältnisse zwischen 1,2 und 6 angewendet.

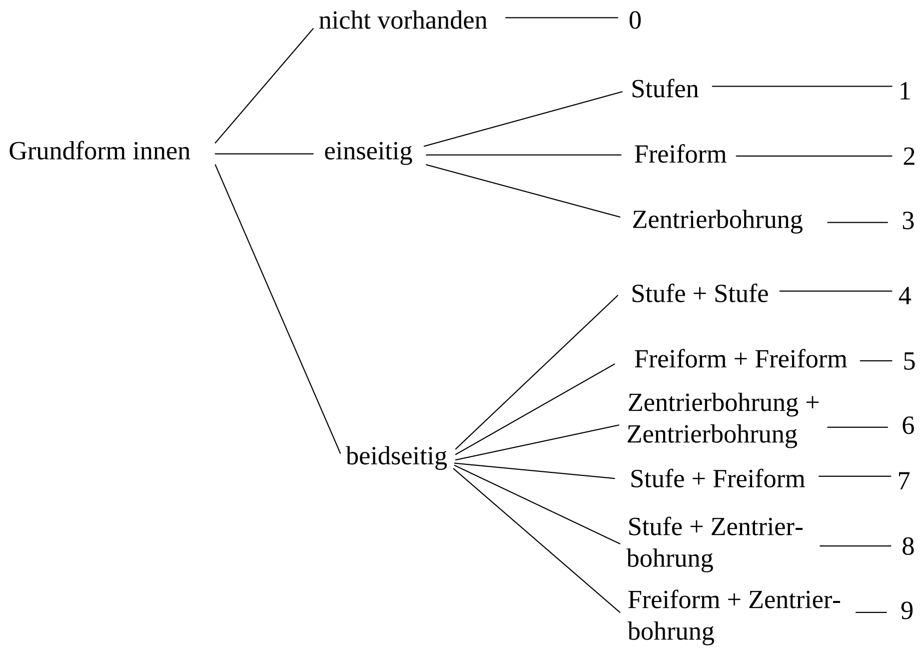


Abbildung 5.4: Struktur der Merkmalsausprägungen der inneren Grundform

Um für gegebene Ausspannlänge l die maximal erreichbare Toleranz IT_{max} zu berechnen, stellen wir die Gleichung um:

$$IT_{max} = 5 \cdot \left(\frac{l}{d^{\frac{2}{3}}} \right)^2 \tag{5.2}$$

Die mit fliegender Einspannung erreichbare Toleranz IT_{max} wird mit der schärfsten Werkstücktoleranz verglichen.

5.1.6 Mögliche Spannflächen

Die primitiven Bearbeitungselemente an den Enden des Werkstücks werden untersucht, ob sie für eine Einspannung mit Backen mögliche Spannflächen darstellen. Ziel ist die Bestimmung potentieller Spannstellen am Rohteil, Fertigteil oder an einem Zwischenzustand. Das Spannmittel ist zu diesem Zeitpunkt noch unbekannt, so daß gefundene Spannflächen einer späteren, genaueren

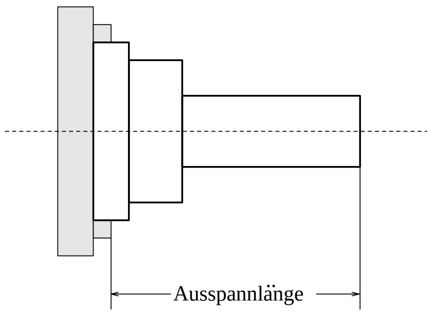


Abbildung 5.5: Ausspannlänge des Werkstücks

Prüfung unterzogen werden müssen. Eine Auswahl erfolgt nach folgenden Kriterien:

Form der Spannfläche

Spannflächen müssen zylindrisch sein bzw. eine „große“ Abweichung von der Zylindrizität in Richtung Kegelfläche ist nicht erlaubt (maximal 3^o).

Lage der Spannfläche

Die Spannfläche kann eine Außenfläche oder eine Innenfläche sein. Sonderfall ist das Spannen mit Spitzen. Der Abstand zu den Stirnflächen des Werkstücks sollte gering sein (möglichst nahe an den Stirnflächen).

Größe der Spannfläche

Die Länge der Spannfläche muß ausreichend groß sein. Einflußparameter sind hier Kraftübertragung, Flächenpressung und Geometrie des Spannmittels. Die Spannlänge, d.h. die Länge, die eine zylindrische Fläche mindestens haben muß, um Spannfläche zu sein, ermittelt sich nach folgender Formel:

$$L_{SP} = \frac{L}{\sqrt{D}} \qquad L_{SP}: \text{Spannlänge}, L: \text{Werkstücklänge}, D: \text{maximaler Durchmesser}$$

Weiterhin sollte die Spannfläche mindestens 70% des Durchmessers der aktuellen Bearbeitungsfläche besitzen.

Nachbarelemente der Spannfläche

Haben die Nachbarelemente der Spannfläche einen kleineren Durchmesser, dann kann die ausgewählte Fläche zum Spannen verwendet werden. Falls dies nicht zutrifft, so müssen weitere Untersuchungen erfolgen, die die Geometrie von Spannbacken (Spann- oder Backentiefe) mitberücksichtigen. Ist die Steigung der Außenkontur kleiner als 55^o, so kann angenommen werden, daß die Spannfläche angrenzend an eine Stirnfläche geeignet ist.

5.2 Die Speichervariable *slots*

Weil die Variable **slots** als Übergabeparameter für das semantische Netzwerk genutzt wird und in ihr damit alle Informationen enthalten sein müssen, um die Plansammlung zu indizieren, wird sie an dieser Stelle etwas ausführlicher vorgestellt; werden bei der Weiterentwicklung dem Planungssystem weitere „Spezialisten“ zugefügt (etwa für Werkzeugwahl oder für Reihenfolgestrategien bei der Bearbeitung), so muß die Variable **slots** um entsprechende Ziele erweitert werden, die von den neuen Spezialisten zur Entscheidungsfindung benötigt werden.

```
slots
    ( class Code:               <Integer>
    clambingLengthReached:      <OrderedCollection (<Boolean> <Boolean>)>
    clambingRadius:             <OrderedCollection ({less10,less360,bigger,nil}
                                {less10,less360,bigger,nil})>
    centeringBorePossible:      <OrderedCollection (<Boolean> <Boolean>)>
    boreLess10:                 <OrderedCollection (<Boolean> <Boolean>)>
    tolerance1:                 <Boolean>
    tolerance2:                 <Boolean>
    face:                       <OrderedCollection (<Boolean> <Boolean>)>
    turning:                    <OrderedCollection (<Boolean> <Boolean>)> )
```

In den Fällen, bei denen der Eintrag aus einer *OrderedCollection* besteht, bezieht sich stets der erste Eintrag auf das linke Ende des Werkstücks, und der zweite auf das rechte.

Bedeutung der einzelnen Einträge:

- classCode
  Der Slot *classCode* beinhaltet den in ?? beschriebenen Code zur Klassifizierung des Werkstücks.

- clambingLengthReached
  Die beiden Goals *neededClambingLengths* und *possibleClambingLengths* werden für die linke und rechte Seite des Werkstücks miteinander verglichen; wenn die gegebene Spannlänge größer oder gleich der benötigten ist, so wird der entsprechende Eintrag auf *true* gesetzt, ansonsten auf *false*.

- clambingRadius
  Für beide Seiten des Werkstücks wird eine Aussage getroffen über den Radius des ersten Bearbeitungselements; ist dieses Element von der Länge her als Spannstelle geeignet (vgl. Slot *clambingLengthReached*), so wird dem Slot in Abhängigkeit vom Maximalradius einer der drei Werte

  - #less10
  - #less360
  - #bigger

  zugewiesen. Ansonsten bleibt er undefiniert.

- centeringBorePossible
  Für die linke und rechte Seite des Werkstücks wird überprüft, ob eine Zentrierbohrung erlaubt ist; eine Zentrierbohrung ist möglich, falls die jeweilige Stirnseite kein Feature enthält und keine Aussagen über eine zu erfüllende Oberflächengüte gemacht werden.

- boreLess10
  Dieser Slot wird für die jeweilige Seite des Werkstücks auf *true* gesetzt, falls eine Bohrung mit Bohrdurchmesser < 10 mm vorliegt oder eine Zentrierbohrung erlaubt ist; ansonsten wird der Eintrag auf *false* gesetzt.

- tolerance1
  Sie wird auf *true* gesetzt, wenn die maximal zu erfüllende Toleranz des Werkstücks größer ist als die bei gegebener Ausspannlänge maximal erreichbare Fertigungstoleranz ist.

- tolerance2

- face
  Der Slot *face* wird für die jeweilige Seite des Werkstücks auf *true* gesetzt, falls eine Bearbeitung der Stirnseite notwendig ist, ansonsten auf *false*. Eine Bearbeitung der Stirnseite ist notwendig, wenn entweder die Einhaltung einer Oberflächengüte gefordert ist oder die Stirnseite ein Feature (außer einer Zentrierbohrung) enthält.

- turning
  Der Eintrag *turning* macht eine Aussage darüber, ob das Anfangs- bzw. Endelement des Werkstücks (als potentielles Spannelement) bearbeitet werden muß; ist dies nicht der Fall, so kommt man möglicherweise mit einer Aufspannung weniger aus. Dabei kann auf Bearbeitung verzichtet werden, wenn die folgenden Punkte erfüllt sind:

  - Das Bearbeitungselement ist ein Zylinderbereich
  - Es werden keine Angaben über zu erfüllende Oberflächengüten gemacht
  - Das Bearbeitungselement enthält keine Features
  - Der Radius des Element ist gleich dem maximalen Werkstückdurchmesser

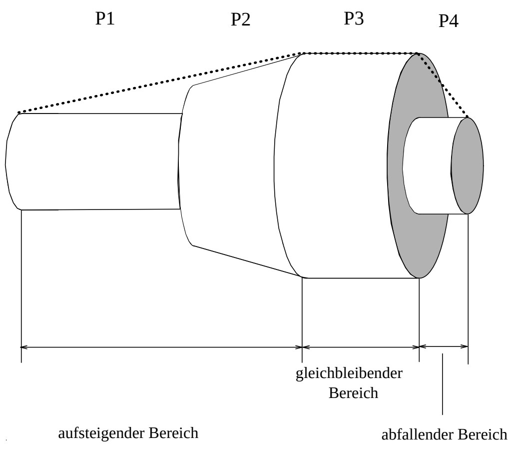


Abbildung 5.6: Bearbeitungsstrukturen eines Werkstücks

5.3 Erzeugung der Baumstruktur

Ein Werkstück kann man allgemein einteilen in eine aufsteigende, eine gleichbleibende und eine abfallende Struktur. Sinn dieser Einteilung ist die Annahme, daß zusammenhängende Strukturelemente in einem Bearbeitungsdurchgang gefertigt werden können. Dabei bezeichnet die aufsteigende Struktur den Bereich vom Beginn bis zum maximalen Radius, die gleichbleibende den Maximalbereich und die abfallende entsprechend den Bereich vom Maximalradius bis zum Ende. Abbildung ?? verdeutlicht diese Einteilung.

Diese Einteilung läßt sich auch rekursiv fortführen: eine Struktur besitzt weitere Unterstrukturen, wenn ihre Komponenten sich nicht monoton verhalten (d.h. ansteigende Strukturen nicht monoton wachsen, abfallende nicht monoton fallen und gleichbleibende nicht konstant bleiben). Dabei versteht man unter der gleichbleibenden (Unter-) Struktur je Schachtelungstiefe abwechselnd das Strukturmaximum bzw. -minimum, Abbildung ?? zeigt ein geschachteltes Strukturelement.

In der Phase der Überprüfung der strukturellen Ähnlichkeit zweier Werkstücke durch ein *Tree–Mapping*–Verfahren werden die so erzeugten Strukturen eingesetzt, wobei ihre Darstellung noch durch Hinzunehmen von Features oder Oberflächengüten bzw. Toleranzen erweitert werden kann; diese werden stets als Unterstrukturen der Komponenten aufgefaßt, zu denen sie gehören.

Um die Dreiteilung einer komplexen Komponente in aufsteigenden, gleichbleibenden und abfallenden Bereich in die Werkstückrepräsentation aufnehmen zu können, werden hierfür die Unterklassen **AscendingComponent**, **FlatComponent** und **DescendingComponent** der Klasse **ComplexComponent** eingeführt.

5.4 Indizierung der Fallbasis

5.4.1 Entscheidungsnetze als Domänenspezialisten

Eine Plansammlung ist nutzlos, wenn sie nicht so organisiert ist, daß in angemessener Zeit darauf zugegriffen werden kann. Deshalb braucht man neben den Plänen eine geeignete Speicherorganisation. Bei der Wahl einer geeigneten Organisationsform muß von den gewünschten Funktionen ausgegangen werden. Wenn wir davon ausgehen, daß daß die Grundaufgabe eines Planers die ist, Pläne zu erzeugen, die ihre Ziele erfüllen, erscheint es als ausreichend, wenn wir die erzeugten Pläne zusammen mit den Zielen in der Plansammlung abspeichern. In der ersten Prototypversion des CAbPlan–Systems wurde genauso vorgegangen, als Speicherstruktur der Plansammlung wurde eine einfache lineare Liste gewählt, wobei das Retrieval sich darauf beschränkte, den Plan zu finden, der die meisten Goals erfüllt. Dabei werden alle gespeicherten Pläne betrachtet. Es geht

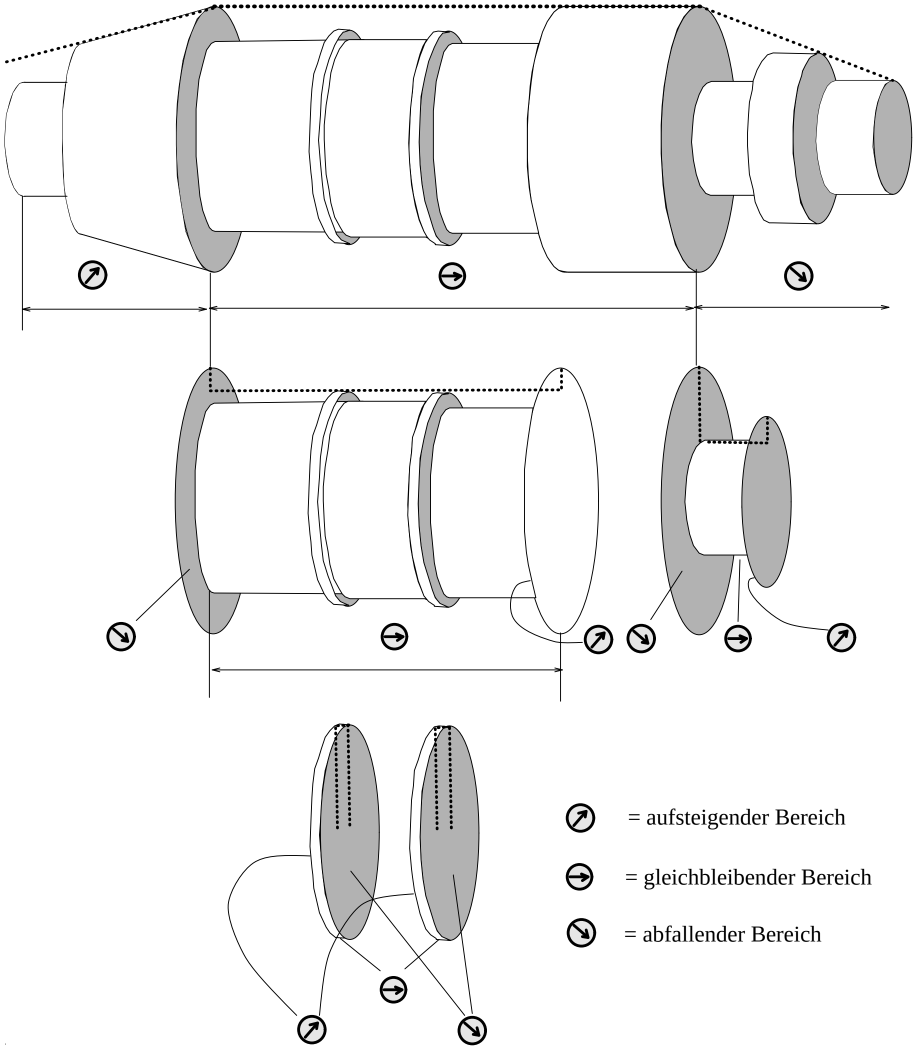


Abbildung 5.7: Werkstück mit geschachtelten Strukturen

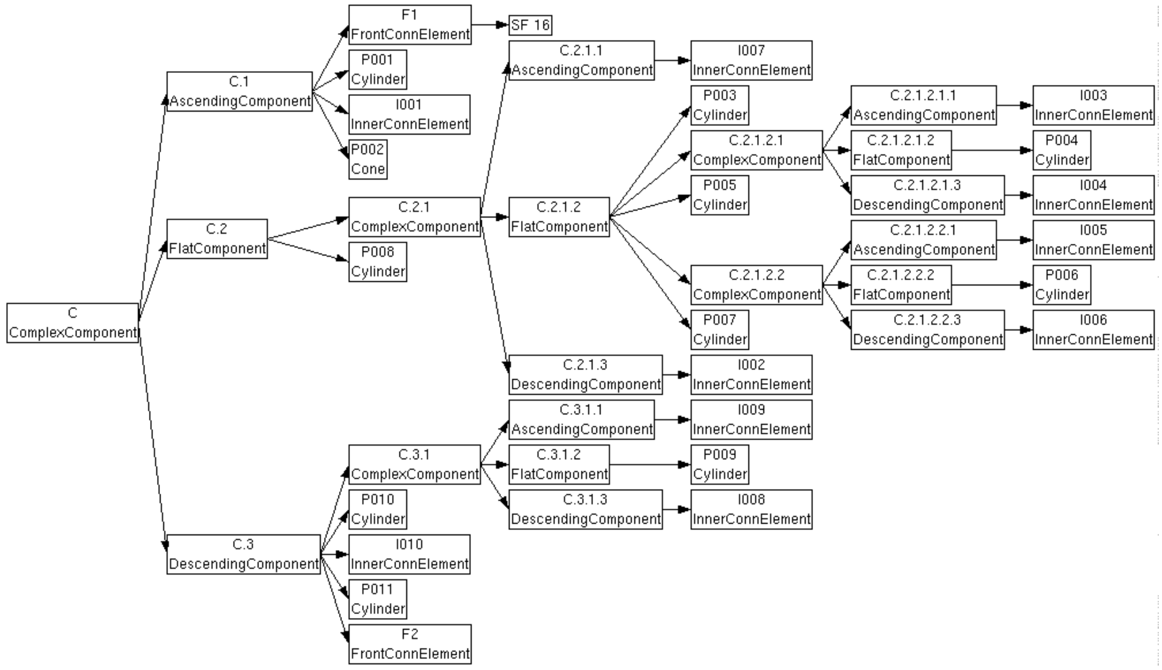


Abbildung 5.8: Darstellung des Werkstücks aus Abb. 5.7 als Baumstruktur

also nicht darum, daß jedes gegebene Ziel einen Plan findet, der es erfüllt, sondern darum, den Plan zu finden, der die meisten Ziele erfüllt.

Demgegenüber erscheint es zweckmäßig, zwischen den gegebenen Zielen und den Plänen, die sie erfüllen, einen Link zu erzeugen. Ein solcher Link würde es dem Planungssystem erlauben, leicht die Pläne zu finden, die alle Ziele erfüllen, ihm aber auch eine Möglichkeit geben, neu geschaffene Pläne an der richtigen Stelle in der Plansammlung abzulegen.

Es ist so, daß ein fallbasierter Planer für viele Ziele gleichzeitig planen muß. Um dies zu erfüllen, muß die Organisation der Fallbasis mehr enthalten als einfache Links zwischen einzelnen Zielen und den Plänen, die sie erfüllen. Er muß vielmehr in der Lage sein, mit einer Menge von Goals umgehen zu können und einen Plan zurückzugeben, der sie alle erfüllt (falls ein solcher existiert). Eine Möglichkeit hierzu ist ein Entscheidungsnetz (*Discrimination net*), bei dem auf Pläne über die Ziele zugegriffen wird, die er erfüllt; die Entscheidungsknoten enthalten Zugriffe auf generalisierte Ziele. Indem das Netz nun durchlaufen wird, wird ein Pfad gefunden, der alle geforderten Ziele erfüllt und an dessen Ende eine Menge von Plänen liegt, die alle Ziele erfüllen.

Wurden die Entscheidungsnetze durchlaufen, so liefern sie neben einem Index für die Plansammlung auch zwei Listen mit vorgeschlagenen Spannmitteln, jeweils für eine Aufspannung von links und eine Aufspannung von rechts[1].

5.4.2 Der Aufbau des semantischen Netzwerks

Das semantische Netzwerk wird in der Klassenvariable **Network** der Klasse **System** abgelegt. Sie ist eine geordnete Liste aus Instanzen der Klasse **SemanticalNet**. Der Aufbau der einzelnen Netze sowie den in ihnen enthaltenen Entscheidungsknoten läßt sich aus Abb. ?? erkennen. Dabei haben die einzelnen Variablen die folgende Bedeutungen (Die Komponenten des Anticipators werden in

[1] Die Spannmittel werden für eine Aufspannung des fertig bearbeiteten Werkstücks ausgewählt; man kann davon ausgehen, daß sich das Rohteil in nahezu allen Fällen mit dem gleichen Spannmittel spannen läßt

Kapitel ?? beschrieben):

SemanticalNet

name:	Name des Entscheidungsnetzes
entry:	Eintrittsknoten in das Netz
anticipatingComponents:	dem Knoten zugeordnete Anticipator–Komponenten (→ ??)
exits:	In dieser geordneten Liste werden die Exits des Netzes abgespeichert
lastNode:	letzter Entscheidungsknoten des Netzes

DecisionNode

net:	das semantische Netz, das den Entscheidungsknoten enthält
number:	Nummer des Knotens im Netz
decisionBlock:	Entscheidungsblock des Knotens; muß einen Übergabeparameter für die Relation **slots** enthalten sowie einen booleschen Wert als Ergebnis liefern
description:	eine syntaktische Beschreibung des Knotens zur Orientierung des Benutzers
truePath:	linker Nachfolger des Knotens, der im Falle einer wahren Auswertung des Entscheidungsblocks eingeschlagen wird; er kann entweder ein weiterer Knoten oder ein Exit sein
falsePath:	rechter Nachfolger des Knotens, der im Falle einer falschen Auswertung des Entscheidungsblocks eingeschlagen wird

Exit

number:	die Nummer des Exits im Netz
contents:	syntaktische Beschreibung des Exits zur Orientierung des Benutzers
nextEntry:	Pointer auf die Wurzel des nächsten Netzes

In Anhang ?? werden die implementierten semantischen Netze, die den Aufspannspezialisten zur Fallauswahl darstellen, vorgestellt. Es handelt sich um vier Netze, die also eine vierdimensionale Fallbasis adressieren.

5.4.3 Der Algorithmus zur Indizierung der Fallbasis

Die Vorgehensweise des Retrievers zur Indizierung der Fallbasis mit Hilfe des semantischen Netzwerks läuft nach folgendem Algorithmus ab:

Eingabe: **slots**

1. $index \leftarrow \emptyset$
2. $net \leftarrow 1$
3. $node \leftarrow \mathrm{entry}(net)$
4. if $\mathrm{decisionBlock}(node)[\mathbf{slots}] = true$
   then $node \leftarrow \mathrm{truePath}(node)$
   else $node \leftarrow \mathrm{falsePath}(node)$
5. $node$ ist **DecisionNode** $\Rightarrow$ gehe zu 4.
6. $index \leftarrow index \cup \mathrm{number}(node)$ ($node$ ist ein Exit!)
7. $net \leftarrow net + 1$
8. $net \neq \emptyset \Rightarrow$ gehe zu 3.
   Ausgabe: $index$

Der so erzeugte Index wird nun zur Adressierung der Fallbasis benutzt, wobei dieser Teil der Retrievalphase als Ausgabe die Menge von Plänen liefert, die unter dem berechneten Index abgespeichert sind.

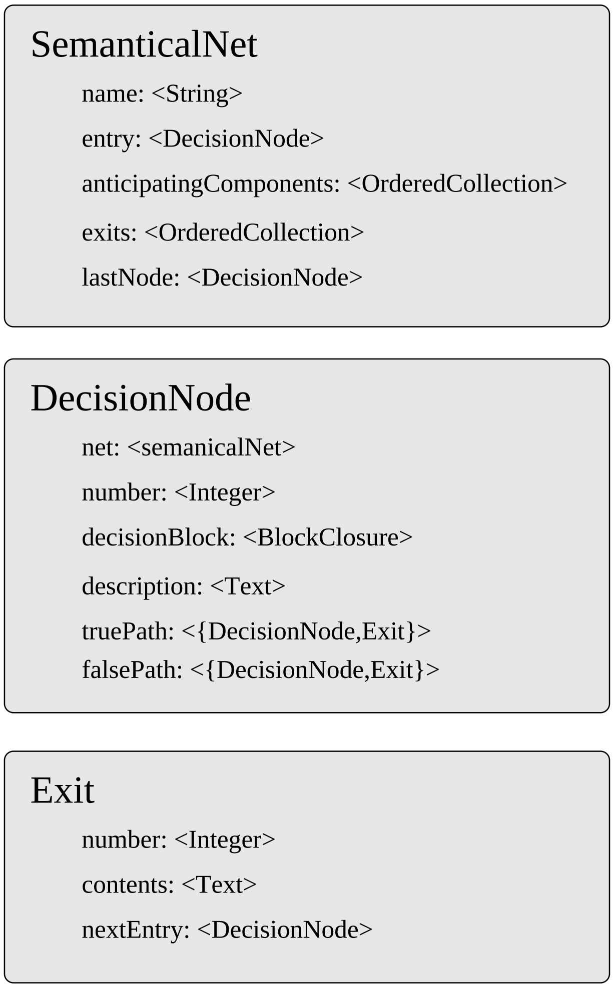


Abbildung 5.9: Die Objektklassen **SemanticalNet**, **DecisionNode** und **Exit**

5.5 Mapping von Baumstrukturen

Nach der Indizierung der Fallbasis führt der Planer für alle unter der gefundenen Adresse abgelegten Fälle ein Mapping durch, um denjenigen Fall auszuwählen, der die geringsten Kosten verursacht. Gibt es mehrere Mappings mit gleichen minimalen Kosten, so überläßt das System die Entscheidung zwischen diesen dem Benutzer. In den Abbildungen **??** und **??** werden die Werkstücke aus Abbildung **??** zum einen mit dem *Tai–Mapping*–Algorithmus und zum anderen mit dem *Strictly Structure Preserving Mapping*[2]–Algorithmus gemappt, wobei die beiden Methoden unterschiedliche Resultate liefern. In den Darstellung werden freie Knoten durch gestrichelte Umrandungen dargestellt; sie müssen also (im linken Baum) gelöscht bzw. (im rechten Baum) eingefügt werden. Unterschiedliche Abbildungen bei den angewendeten Mapping–Algorithmen wird durch inverse Darstellung der entsprechenden Knoten verdeutlicht.

Für die beiden betrachteten Werkstücke gilt, daß das rechte Werkstück Unterkomponente der gleichbleibenden Komponente des linken Werkstücks ist.

Das Tai–Mapping kommt mit sieben Lösch- und zwei Einfügeoperationen aus; hinzu kommt

[2] Der von E. und K. Tanaka angegebene Algorithmus zum *Structure Preserving Mapping* liefert nicht das in der Definition geforderte Ergebnis; der Vollständigkeit halber wurde er aber trotzdem implementiert. Das gewünschte Ergebnis wird mit einem Mapping-Algorithmus erziehlt, der hier *Strictly Structure Preserving Mapping* heißt [39]

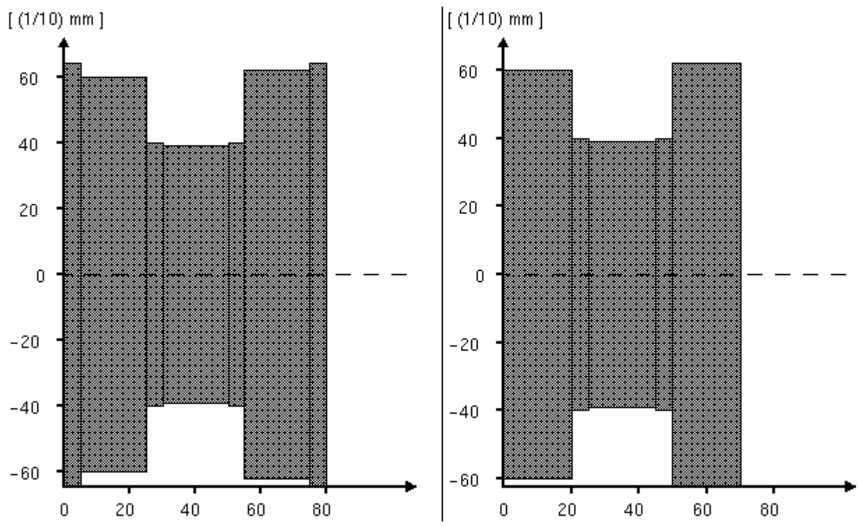


Abbildung 5.10: Gemappte Werkstücke

eine Ersetzungsoperation ($P007\ Cylinder \Rightarrow C.2\ FlatComponent$), die sich aus einer Lösch- und einer Einfügeoperation zusammensetzt. Die folgende Tabelle zeigt den berechneten Abstand zwischen den beiden Werkstücken nach dem Tai–Mapping:

Operator	Komponente	Name	Kosten
LÖSCHE	$AscendingComponent$	$C.1$	50
LÖSCHE	$FlatComponent$	$C.2$	50
LÖSCHE	$SurfaceFinish$	$SF160$	10
LÖSCHE	$Cylinder$	$P006$	4
LÖSCHE	$InnerConnElement$	$I006$	1
LÖSCHE	$Cylinder$	$P003$	4
LÖSCHE	$InnerConnElement$	$I003$	1
FÜGE EIN	$AscendingComponent$	$C.1$	500
FÜGE EIN	$Cylinder$	$P005$	30
LÖSCHE	$Cylinder$	$P007$	4
FÜGE EIN	$FlatComponent$	$C.2$	500
		$\sum$	1154

Betrachtet man die resultierenden Baumstrukturen nach Entfernen der freien Knoten (Abb. ??), so erkennt man, daß sich die Strukturen unterscheiden. Das Hauptkriterium für das Tai–Mapping liegt also in dem Finden der kostengünstigsten Operatorfolge zulasten von Strukturgleichheit.

Ein Mapping der gleichen Werkstücke mit dem Strictly Structure Preserving Mapping ergibt acht Löschoperationen, drei Einfügeoperationen und zwei Ersetzungsoperationen; für die Kosten ergibt sich folgendes Ergebnis:

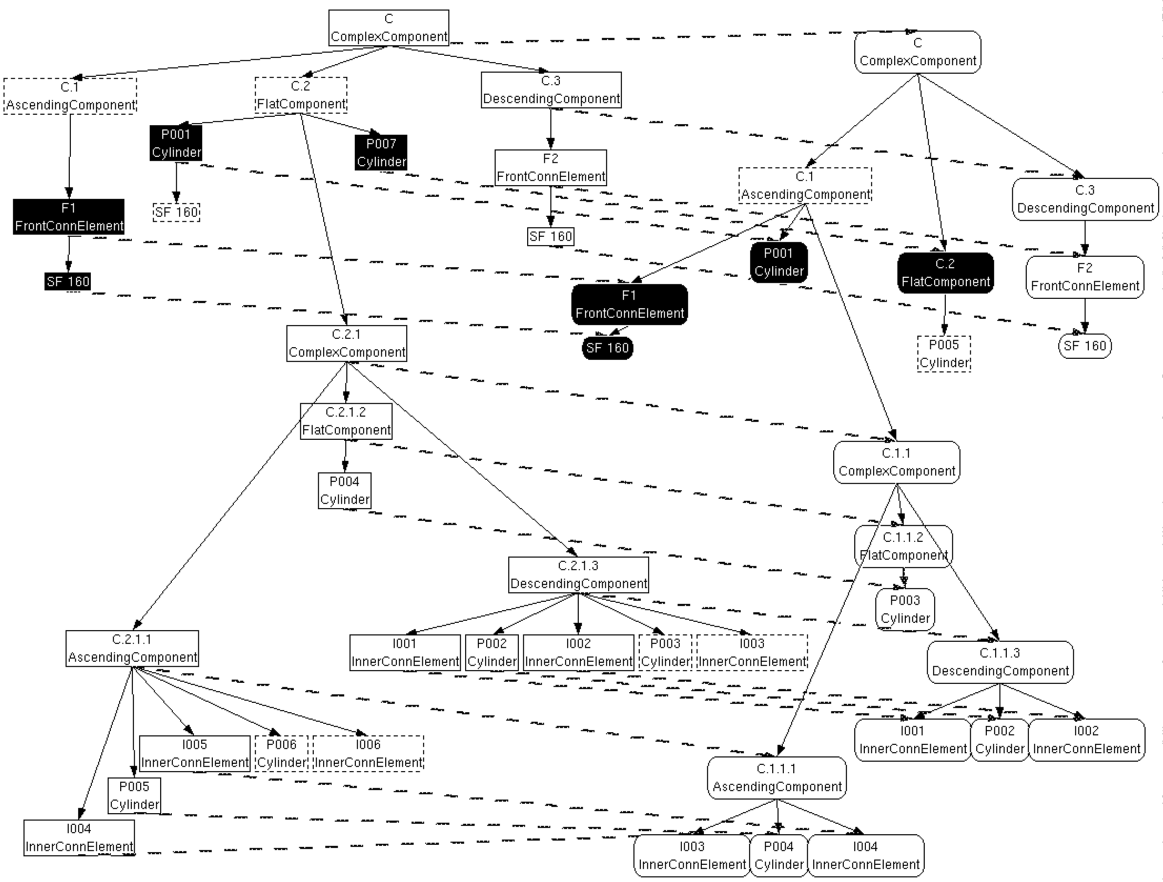


Abbildung 5.11: Mapping der Werkstücke durch Tai–Mapping

Operator	Komponente	Name	Kosten
LÖSCHE	$AscendingComponent$	$C.1$	50
LÖSCHE	$FrontConnElement$	$F1$	5
LÖSCHE	$SurfaceFinish$	$SF160$	10
LÖSCHE	$Cylinder$	$P006$	4
LÖSCHE	$InnerConnElement$	$I006$	1
LÖSCHE	$Cylinder$	$P003$	4
LÖSCHE	$InnerConnElement$	$I003$	1
LÖSCHE	$Cylinder$	$P007$	4
FÜGE EIN	$Cylinder$	$P001$	30
FÜGE EIN	$FlatComponent$	$C.2$	500
FÜGE EIN	$Cylinder$	$P005$	30
LÖSCHE	$FlatComponent$	$C.2$	50
FÜGE EIN	$AscendingComponent$	$C.1$	500
LÖSCHE	$Cylinder$	$P001$	4
FÜGE EIN	$FrontConnElement$	$F1$	10
		$\sum$	1183

Vergleicht man wiederum die nach dem Streichen der freien Knoten verbleibenden Baumstrukturen (Abb. ??), so erkennt man, daß sie identisch sind. Dies entspricht der Zielsetzung des

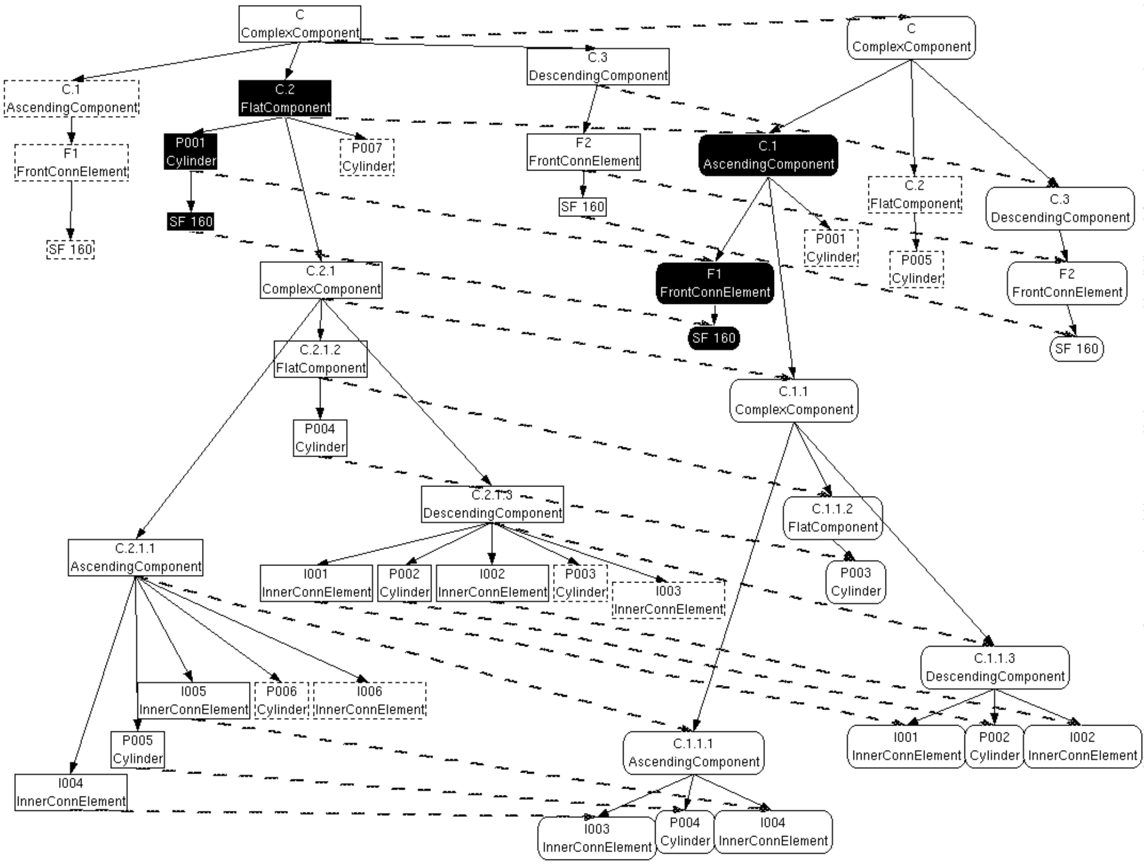


Abbildung 5.12: Mapping der Werkstücke durch Strictly Structure Preserving Mapping

Structure Preserving Mapping, gleiche Unterstrukturen aufeinander abzubilden. Hierfür wird ein erhöhter Modifikationsaufwand in Kauf genommen.

5.6 Der Anticipator

Nach [8] besteht die Aufgabe des Anticipators darin, anhand der Goals die Ausgangsituation der Planungsaufgabe zu untersuchen. Er entscheidet, ob **vor** der eigentlichen Planungsaufgabe eine Entscheidung getroffen werden muß und wählt als eigenständiges Modul einen Plan aus, der bekannte Fehler vermeidet. Diese strikte Trennung zwischen Anticipator und Retriever ist im vorliegenden System nicht gegeben. Jedem Knoten im semantischen Netzwerk kann eine *Anticipator–Komponente* hinzugefügt werden, die während des Netzdurchlaufs eine „Entscheidungsumlenkung“ an einer Stelle vornimmt, an der eine Fehlentscheidung vermutet wird (Abb. ??). Somit ist ein Link zu der Stelle, an der ein Fehler auftritt, durch diese Struktur impliziet gegeben. Wird eine Umlenkung ausgelöst, beendet das System den Durchlauf durch das jeweilige Netz durch den Exit, der die richtige Entscheidung darstellt. Im Gegensatz zum Anticipator im CHEF–System wird also kein einzelner Plan durch den Anticipator ausgewählt, vielmehr wird der Index zur Adressierung der Fallbasis verändert und damit eine Menge von Plänen ausgewählt, die den Fehler vermeiden. Aus dieser Menge wird mit Mappen der Baumstrukturen ein ähnlichster Fall ausgewählt.

Somit dient die Variable *slots* nicht nur der Repräsentation der zu erfüllenden Ziele in einer

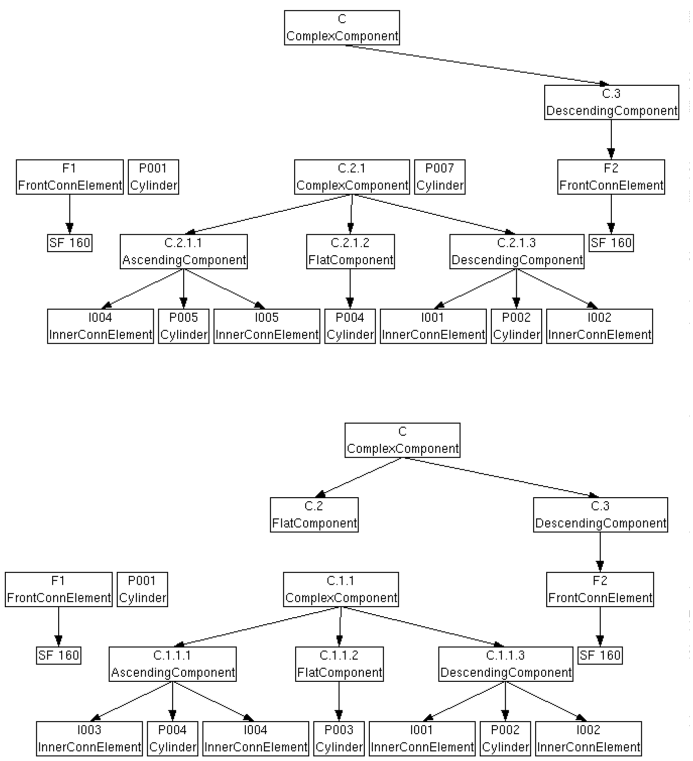


Abbildung 5.13: Resultierende Strukturen nach Entfernen der freien Knoten beim Tai–Mapping

generalisierten Form, sondern enthält gleichzeitig auch die zu vermeidenden Fehler.

Der Anticipator schließt beim CHEF–System die Kette der einzelnen Komponenten des fallbasierten Planungssystems (s. ??). Er erhält sein Wissen über Fehler vom *Assigner*, der das Ergebnis von Modifikations- und Reparaturphase untersucht. Hierzu ist Wissen über den Grund für das Auftreten des Fehlers nötig. In CHEF wird dieses Wissen in Form eines Vokabulars dargestellt, daß auch in der Simulationsphase zur Überprüfung des Ergebnisses der Modifikation eingesetzt wird.

Eine entsprechende Komponente steht im CAbPlan–System in der aktuellen Implementationsphase noch nicht zur Verfügung; die vorgestellten Aniticipator–Komponenten können daher nur aufgrund von Benutzereingaben an die entsprechende Stelle im semantischen Netzwerk gesetzt werden. Auch die Angabe des zu wählenden Exits geschieht nicht automatisch. Stehen die erforderlichen Komponenten des Planungssystems zur Verfügung, so wird die Variable *reason* zur Aufnahme einer Fehlerbeschreibung auf der Basis eines geeigneten Vokabulars dienen; momentan enthält sie lediglich eine vom Benutzer eingegebene Fehlerbeschreibung ohne syntaktische Bedeutung.

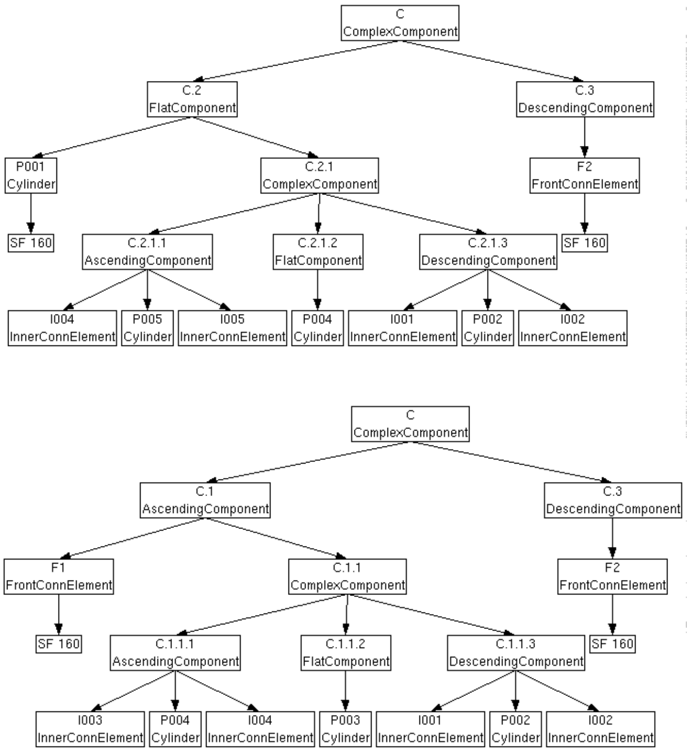


Abbildung 5.14: Gleiche Strukturen nach Entfernen der freien Knoten beim Strictly Structure Preserving Mapping

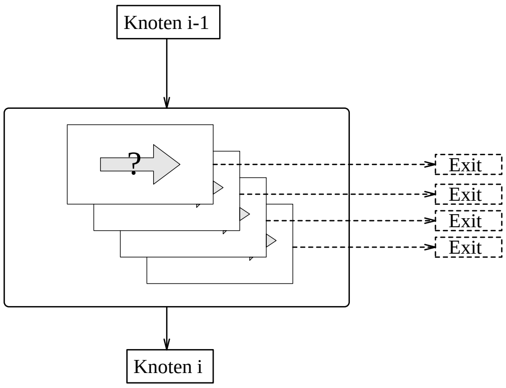


Abbildung 5.15: Anticipator–Komponente vor Entscheidungsknoten

Die Datenstruktur einer Anticipator–Komponente zeigt Abbildung ??. Hierbei repräsentiert die Variable *anticipatingValues* die generalisierten Goals, die der Variablen *slots* bei der Werkstückrepräsentation entspricht (s. ??). Die Einträge dieser beiden Variablen werden miteinander verglichen und bewirken bei Übereinstimmung eine Umlenkung nach dem in der Variablen *correctExit* gespeicherten Exit des Netzes.

AnticipatingComponent

anticipatingValues: <DescriptionDictionary>

correctExit: <Exit>

reason: <Text>

Abbildung 5.16: Struktur einer Anticipator-Komponente

Kapitel 6

Bewertung und Ausblick

Die vorliegende Implementierung findet aufgrund einer initialen Aufgabenstellung ein Werkstück in der Fallbasis, das dem Problem am ähnlichsten sein soll. Hierzu wird ein Entscheidungsnetz durchlaufen, um die Fallbasis aufgrund von generalisierten Goals zu indizieren. Eine auf diese Weise gewonnene Strukturierung der Fallbasis anhand eines Aufspannungsspezialisten ist sicherlich sinnvoller als eine unstrukturierte Fallbasis, in der linear nach dem Plan gesucht wird, der die meisten zu erfüllenden Ziele erreicht. Mit zunehmender Anzahl von Fällen verändert sich die Zeit zum Indizieren der Fallbasis nicht. Es wächst lediglich die Anzahl der jeweils zu mappenden Fallbeispiele, was aber bei einer einigermaßen guten Verteilung neu hinzukommender Fälle auf die verschiedenen Adressen der Fallbasis nicht ins Gewicht fällt. Gibt es nur geringe Unterschiede zwischen den einzelnen Plänen, kann dies dazu führen, daß viele Fälle unter dem gleichen Index abgespeichert werden, was zu einem fast linearen Anwachsen der zum Mapping benötigten Zeit führt.

Durch das gefundene Fallbeispiel soll die weitere Planungsstrategie von CAbPlan auf verschiedenen Hierarchieebenen gesteuert werden; die Planungsreihenfolge Anticipation – Retrieval – Modifikation des CHEF–Systems wird also nicht übernommen. Der gefundene Plan erfüllt lediglich die Ziele des Aufspannspezialisten[1]; man spricht von Ähnlichkeit auf der entsprechenden Abstraktionsebene. Aus dem vorliegenden Plan wird lediglich ein Skelettplan auf der Basis von Termen der Aufspannung direkt abgeleitet. Es kann nicht garantiert werden, daß auf der Basis dieses Skelettplanes auf allen Abstraktionsebenen ein korrekter Plan hergeleitet werden kann, auch wenn bereits auf der obersten Ebene mögliche Interaktionen zwischen Planungsschritten darunterliegender Ebenen erkannt werden. Es muß eine generative Planungskomponente vorhanden sein, die bei der Verfeinerung des Skelettplanes noch nicht erreichte Ziele bearbeitet, um so stets eine korrekte Lösung zu garantieren.

Das semantische Netzwerk ist nicht zwingend als Aufspannspezialist festgelegt; durch die n-dimensionale Struktur der Fallbasis (wobei n die Anzahl der Entscheidungsnetze ist) kann man ebensogut eine Indizierung der Fallbasis durch mehrere voneinander unabhängige Experten durchführen. Betrachtet man die Ergebnisse der einzelnen Netze getrennt voneinander, so läßt sich ein Fehler der Retrievalphase leichter lokalisieren. Die verschiedenen Abstraktionsebenen werden jeweils einem Spezialisten zugeordnet. Kommt es bei der Verfeinerung von Planungsschritten von einem Abstraktionsniveau zum nächsten zu Problemen, so wird die Entscheidung des entsprechenden Spezialisten als falsch angesehen, und eine entsprechende Anticipationskomponente wird aktiviert. Entscheidungsgraphen für Maschinenwahl, Werkzeugwahl, Spannmittelwahl und Bearbeitungsoptimierung, die in Form von semantischen Netzen implementiert werden können, werden in [25] angegeben.

Mit dem gegenwärtigen Wissensstand konnten das Entscheidungsnetzwerk als Aufspannungsspezialist und die Variable *slots*, die zur Aufnahme der Goals dient, nicht endgültig festgelegt werden. Es zeigt sich, daß die Variable *slots* auch von den Anticipator–Komponenten genutzt

[1] Durch die Anwendung von Mapping auf die strukturierte Werkstückbeschreibung erfüllt er eigentlich auch die Ziele eines Geometriespezialisten

wird, um zu entscheiden, ob für ein gegebenes Werkstück Fehler bei der Planung auftreten. Der in **??** beschriebene Aufbau kann dieser Aufgabe sicherlich nicht gerecht werden, weil hierin nur die für das momentan implementierte Entscheidungsnetzwerk relevanten Goals berücksichtigt werden. Neben Erweiterung der Variablen *slots* durch eventuelle Hinzunahme weiterer Entscheidungsnetze bzw. Entscheidungsknoten innerhalb der Netze muß sie also auch eine Art „negativer Goals“ aufnehmen, die Fehlerursachen darstellen und zu einer Beeinflussung der Fallauswahl führen.

Anhang A

Implementierungen

A.1 Datenstrukturen zur Repräsentation der Werkstücke

Bei der Beschreibung der einzelnen Datenstrukturen bzw. der Variablen wird der in Smalltalk übliche Sprachgebrauch verwendet; insbesondere wird der Begriff der **Klasse** (*class*) mit den zugehörigen **Klassenvariablen** (*class variables*) zur Bezeichnung der (hierarchischen) Objektmengen benutzt. Klassen- und Klassenvariablennamen beginnen mit einem Großbuchstaben. Im Falle von Entities einer Klasse gebraucht man den Begriff der **Instanz** (*instance*); die eine Instanz repräsentierenden Variablen werden entsprechend als **Instanzenvariablen** (*instance variables*) bezeichnet; sie beginnen stets mit einem kleinen Buchstaben. Der Einfachheit halber werden Instanzenvariablen oft als **Variablen** bezeichnet. Die Funktionen und Prozeduren auf Instanzen bzw. Klassen werden im Smalltalk–Sprachgebrauch **Methoden** genannt.

Ein wesentlicher Gesichtspunkt der Objektorientierten Programmierung (und somit von Smalltalk) ist die Hierarchiebildung von Klassen; dabei gilt das Prinzip der Vererbung von Eigenschaften von oben nach unten, das heißt, daß Unterklassen Eigenschaften und Variablendeklarationen von Klassen übernehmen, die in der Klassenhierarchie weiter oben stehen.

A.1.1 Attribute

Die Merkmale von Entities werden mit Hilfe von Attributen beschrieben. Hierzu wird eine Klasse **Attribute** definiert:

Attribute (	**name:**	**<String>**
	type:	**<String>**
	default:	**type**
	value:	**type)**

Bedeutung der Instanzenvariablen:

name:	Name des Attributs
type:	Datentyp des Attributs
default:	Defaultwert des Attributs
value:	Wert des Attributs

Bemerkungen:

- Die in den Variablen **default** und **value** gespeicherten Werte müssen dem Datentyp entsprechen, der in der Variablen **type** gespeichert ist
- Der Defaultwert wird angenommen, wenn (bei der Instanzenerzeugung bzw. bei der Benutzereingabe) kein Wert angegeben wird

A.1.2 Relationen

Die Verknüpfung der einzelnen Entities untereinander erfolgt durch Relationen; diese Relationen stellen im Prinzip Pointer zwischen zwei Objekten dar. Durch Relationen lassen sich *besteht–aus–* und *ist–Oberteil–von–*Beziehungen sowie Baumstrukturen darstellen.

Relation (	**name:**	**<String>**
	fromObject:	**<ApplicationObject>**
	toObject:	**<ApplicationObject>)**

Bedeutung der Instanzenvariablen:

name:	Name der Relation
fromObject:	Entity, von dem die Relation ausgeht
toObject:	Entity, auf das die Relation zeigt

A.1.3 Aufbau der Entities

Grundsätzlich werden im Planungssystem alle Entities des Werkstücks, d.h. das Werkstück selbst, seine primitiven sowie komplexen Komponenten, seine Features und Toleranzen mit Hilfe von Unterklassen der Klasse **ApplicationObject** repräsentiert.

ApplicationObject (	**attributes:**	**<DescriptionDictionary>**
	relations:	**<DescriptionDictionary>**
	searchKeys:	**<DescriptionDictionary>)**

Bedeutung der Instanzenvariablen:

attributes:	Verzeichnis der Attribute eines Entities
relations:	Verzeichnis der Relationen eines Entities
searchKeys:	Verzeichnis der Suchschlüssel eines Entities

Bemerkungen:

- Die Klasse **DescriptionDictionary** ist eine Unterklasse der Standard–Smalltalk–Klasse **Dictionary**, die die Werte von Variablen unter einem bestimmten Schlüssel abspeichert:

  Dictionary (**key → value,**
  key → value,
  ...)

- Alle Instanzenvariablen vom Typ **ApplicationObject** besitzen die folgenden Attribute:
  - **name**
    Ein eindeutiger Name des Objekts, der vom System vergeben wird.
  - **type**
    Die Klasse des Objekts
- Die Variable **searchKeys** wird im vorliegenden Planungssystem nicht verwendet
- Für die Klasse **ApplicationObject** sowie für ihre Unterklassen werden eine Reihe weiterer Attribute und Relationen definiert, die hier nicht erwähnt werden; diese sind Bestandteile anderer Projekte, die für das hier beschriebene Planungssystem keine Rolle spielen

Die Bearbeitungsbereiche (Komponenten) eines Werkstücks sind Teilbereiche, für die getrennte geometrische Repräsentationen nötig sind. Man unterscheidet zwischen atomaren (oder primitiven) und zusammengesetzten (komplexen) Bearbeitungsbereichen. Sie werden in der Klasse **Component** zusammengefaßt.

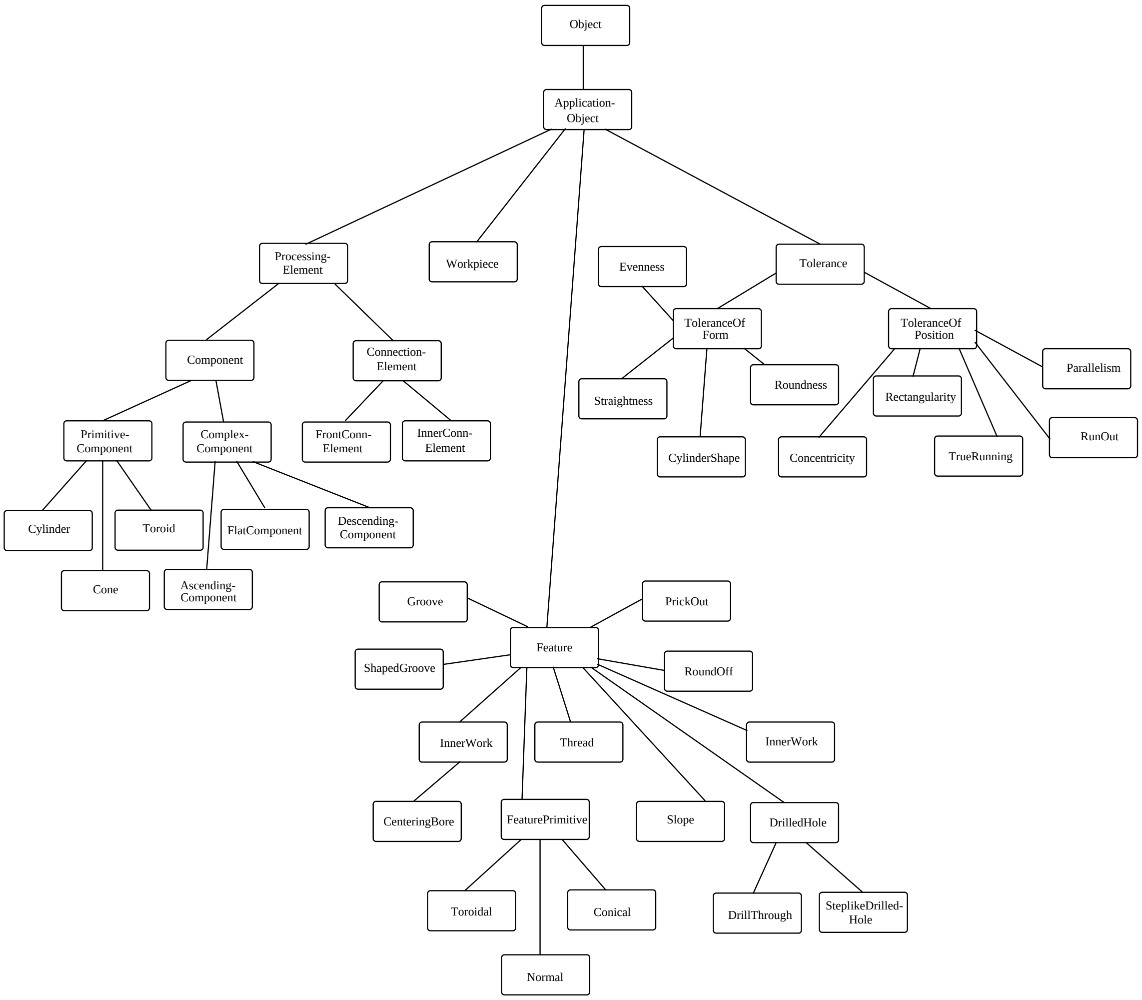


Abbildung A.1: Klassenhierarchie zur Darstellung eines Werkstücks

A.1.4 Primitive Komponenten

Die primitiven Komponenten sind die rotationssymmetrischen Grundkörper, die zur Bildung von komplexen Komponenten aneinandergereiht werden. Es gibt drei Arten von primitiven Komponenten:

- Zylinder
- Kegelstumpfbereich
- Tonnenbereich

Sie unterscheiden sich in der geometrischen Beschreibung (also den Attributen), die Beziehungen zu anderen Entities (Relationen) sind aber gleich. Die Klasse **PrimitiveComponent** ist eine Unterklasse von **ApplicationObject**.

PrimitiveComponent

relations: (	**leftConnEl:**	**<ConnectionElement>**
	rightConnEl:	**<ConnectionElement>**
	parent:	**<ComplexComponent>**

features:	**<OrderedCollection>**
tolerances:	**<OrderedCollection>)**

Bedeutung der Relationen:

leftConnEl:	linkes Verbindungselement
rightConnEl:	rechtes Verbindungselement
parent:	komplexe Oberkomponente
features:	Liste der für die Komponente spezifizierten Features
tolerances:	Liste der spezifizierten Toleranzangaben

A.1.5 Zylinderbereiche

Ein Zylinderbereich (*cylinder*) ist eine primitive Komponente mit konstantem Radius. Durch spezielle Features wie z.B. Rundungen kann es vorkommen, daß der Radius am rechten bzw. linken Rand des Zylinders vom angegebenen Radius abweicht; aus diesem Grunde werden die zusätzlichen Attribute **leftRadius** und **rightRadius** in die Beschreibung aufgenommen.

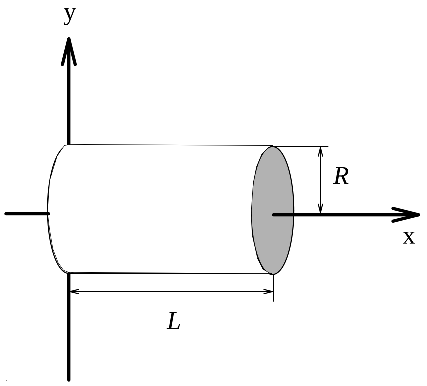


Abbildung A.2: Zylinderbereich

Cylinder

attributes: (	**length:**	**<Number>**
	radius:	**<Number>**
	leftRadius:	**<Number>**
	rightRadius:	**<Number>**
	surfaceFinish:	**<Number>)**

Bedeutung der Attribute:

length:	Länge L des Zylinders
radius:	Radius R
leftRadius:	Radius am linken Ende
rightRadius:	Radius am rechten Ende
surfaceFinish:	Oberflächengüte der Zylindermantelfläche

A.1.6 Kegelstumpfbereiche

Unter einem Kegelstumpfbereich (*cone*) versteht man einen Bereich, in dem sich der Radius konstant ändert. Seine Geometrie läßt sich durch Angabe des rechten und linken Radius sowie der Länge spezifizieren.

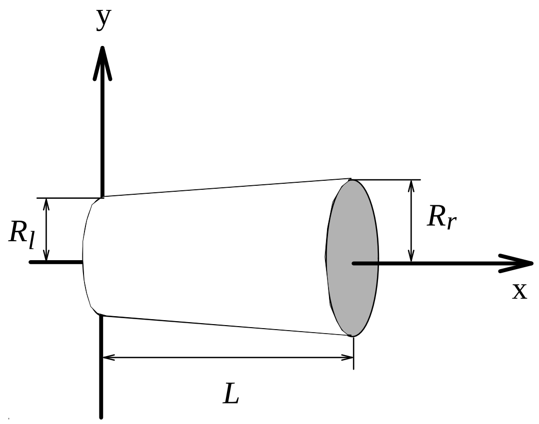


Abbildung A.3: Kegelstumpfbereich

Cone

attributes: (	**length:**	**<Number>**
	leftRadius:	**<Number>**
	rightRadius:	**<Number>**
	surfaceFinish:	**<Number>)**

Die Attribute **leftRadius** und **rightRadius** bezeichnen die Radien am linken bzw. rechten Ende des Werkstücks; die übrigen Attribute haben die gleichen Bedeutungen wie beim Zylinderbereich.

A.1.7 Tonnenbereiche

Ein Tonnenbereich (*toroid*) ist eine primitive Bearbeitungskomponente, deren Kontur eine Krümmung vom Radius R_C besitzt. Diese kann entweder konvex oder konkav sein. Zur geometrischen Spezifizierung genügen neben dem Konturradius und der Art der Krümmung der linke und der rechte Radius des Tonnenbereichs sowie seine Länge; aus diesen Angaben wird der Mittelpunkt des Konturkreises berechnet. Es muß des weiteren folgende Integritätsbedingung erfüllt sein, damit eine sinnvolle geometrische Form erreicht werden kann:

$$R_C \geq \frac{\Delta R^2 + L^2}{2 \cdot L} \qquad , L \leq |\Delta R| \tag{A.1}$$

$$R_C \geq \frac{\Delta R^2 + L^2}{2 \cdot |\Delta R|} \qquad , L > |\Delta R| \tag{A.2}$$

$\Delta R = R_r - R_l$

Toroid

attributes: (	**length:**	**<Number>**
	leftRadius:	**<Number>**
	rightRadius:	**<Number>**
	contRadius:	**<Number>**
	curvature:	**{konvex,koncave}**
	centre:	**<Point>**
	surfaceFinish:	**<Number>)**

Bedeutung der Attribute:

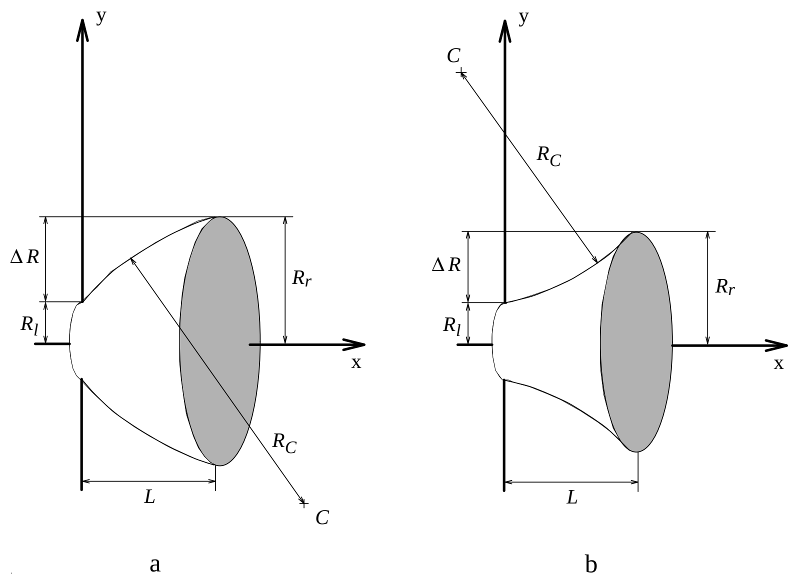


Abbildung A.4: a konvexer Tonnenbereich, b konkaver Tonnenbereich

contRadius:	Radius des Konturkreises
curvature:	Art der Krümmung
centre:	Mittelpunkt C des (oberen) Konturkreises

A.1.8 Komplexe Komponenten

Komplexe Komponenten sind Komponenten, die sich aus Unterkomponenten zusammensetzen. Diese Unterkomponenten können sowohl primitive Komponenten als auch wieder komplexe Komponenten sein. Dabei gilt, daß komplexe Komponenten gleiche Nachbarschaftsbeziehungen wie primitive Komponenten haben, so daß sie sich prinzipiell genau wie diese in die Struktur eines Werkstücks einbauen lassen. Grundsätzlich muß jede komplexe Komponente in der Liste ihrer Komponenten mindestens eine Unterkomponente besitzen.

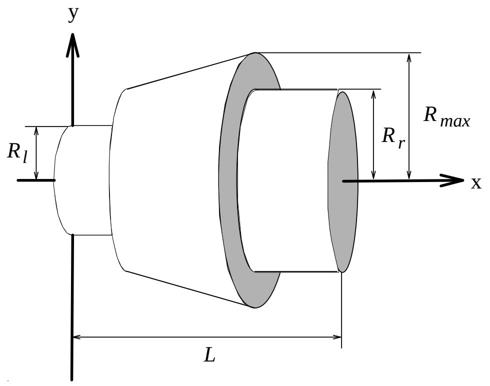


Abbildung A.5: Komplexe Komponente

ComplexComponent

relations: (	**leftConnEl:**	**<ConnectionElement>**
	rightConnEl:	**<ConnectionElement>**
	parent:	**<ComplexComponent>**
	subComponents:	**<OrderedCollection>)**
attributes: (	**length:**	**<Number>**
	leftRadius:	**<Number>**
	rightRadius:	**<Number>**
	maxRadius:	**<Number>)**

Bedeutung der Relationen:

leftConnEl:	Linkes Verbindungselement der komplexen Komponente
rightConnEl:	Rechtes Verbindungselement
parent:	Oberkomponente
subComponents:	Liste der Unterkomponenten (geordnet von links nach rechts)

Bedeutung der Attribute:

length:	Gesamtlänge der komplexen Komponente
leftRadius:	Radius am linken Ende der komplexen Komponente
rightRadius:	Radius am rechten Ende der komplexen Komponente
maxRadius:	Maximaler Radius der komplexen Komponente

A.1.9 Verbindungselemente

Als Verbindungselemente werden die Kreisflächen bezeichnet, die den Abschluß von primitiven Komponenten bilden. Sie stehen senkrecht zur x–Achse und parallel zur y–Achse. Man unterscheidet Planflächen, die zwischen zwei primitiven bzw. komplexen Komponenten liegen, und Stirnseiten, die den Abschluß des Werkstücks bilden. Die Verbindungselemente werden in der Klasse **ConnectionElement** zusammengefaßt.

A.1.10 Planflächen

Planflächen liegen stets zwischen zwei Bearbeitungselementen. Sind der rechte Radius des linken Elementes und der linke Radius des rechten Elementes gleich, so liegt in der Praxis zwar keine Planfläche vor, in der Repräsentation des Werkstücks wird aber dennoch eine eingeführt.

Die Darstellung von Planflächen erfolgt mit Hilfe von Instanzen der Klasse **InnerConnElement**:

InnerConnElement

relations: (	**leftComp:**	**<PrimitiveComponent>**
	rightComp:	**<PrimitiveComponent>**
	features:	**<OrderedCollection>)**
attributes: (	**minRadius:**	**<Number>**
	maxRadius:	**<Number>**
	dirOfMat:	**{left,right,nil}**
	surfaceFinish:	**<Number>)**

Bedeutung der Relationen:

leftComp:	Komponente, die links an die Planfläche anschließt
rightComp:	Komponente, die rechts an die Planfläche anschließt
features:	Liste der Features der Planfläche

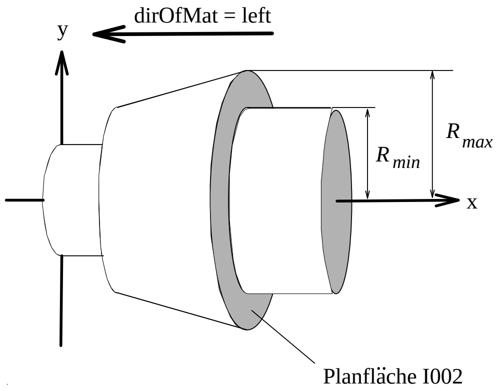


Abbildung A.6: Eine Planfläche des Werkstücks

Bedeutung der Attribute:

minRadius:	Minimaler Radius R_{min} der Planfläche
maxRadius:	Maximaler Radius R_{max} der Planfläche
dirOfMat:	Die Richtung, in der das Werkstück von der Planfläche aus „weitergeht“
surfaceFinish:	Oberflächengüte der Planfläche

Sei LC die linke Komponente der Planfläche und RC seine rechte. Dann gelten die folgenden Beziehungen:

- $R_{min} = min\{rightRadius(LC), leftRadius(RC)\}$
- $R_{max} = max\{rightRadius(LC), leftRadius(RC)\}$
-
  - $rightRadius(LC) > leftRadius(RC) \quad \Rightarrow \quad dirOfMat = left$
  - $rightRadius(LC) < leftRadius(RC) \quad \Rightarrow \quad dirOfMat = right$
  - $rightRadius(LC) = leftRadius(RC) \quad \Rightarrow \quad dirOfMat = nil$

A.1.11 Stirnseiten

Als Stirnseiten bezeichnet man die Kreisflächen an den beiden Enden des Werkstücks. Weil die Repräsentation des Werkstücks auf oberster Stufe aus einer komplexen Komponente mit ihren beiden Stirnseiten besteht, erhalten die Relationen der Stirnseiten Zeiger auf das Werkstück sowie auf die Komponente.

Stirnseiten werden von der Klasse **FrontConnElement** definiert:

FrontConnElement

relations: (	**workpiece:**	**<Workpiece>**
	component:	**<ComplexComponent>**
	features:	**<OrderedCollection>)**
attributes: (	**radius:**	**<Number>**
	dirOfMat:	**{left,right}**
	surfaceFinish:	**<Number>)**

Bedeutung der Relationen:

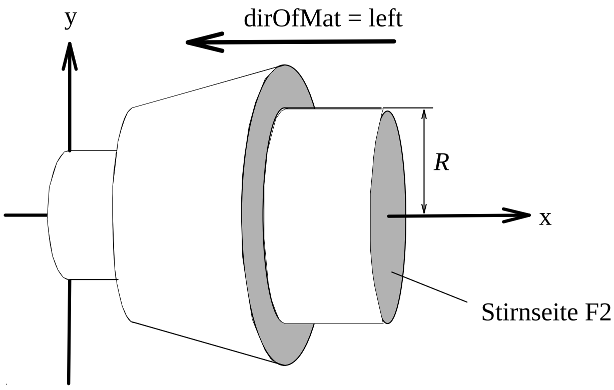


Abbildung A.7: Rechte Stirnseite des Werkstücks

workpiece:	Zeiger auf das gesamte Werkstück
component:	Komplexe Komponente, die alle Elemente des Werkstücks umschließt
features:	Liste der Features der Planfläche

Bedeutung der Attribute:

radius:	Radius R der Planfläche
dirOfMat:	Die Richtung, in der sich das Werkstück von der Stirnseite aus befindet
surfaceFinish:	Oberflächengüte der Planfläche

Für die linke Stirnseite gilt:

- $R =$ linker Radius der ersten primitiven Komponente
- $dirOfMat = right$

Für die rechte Stirnseite gilt entsprechend:

- $R =$ rechter Radius der letzten primitiven Komponente
- $dirOfMat = left$

A.1.12 Features

Neben den primitiven und komplexen Komponenten sowie den Verbindungselementen werden einem Werkstück eine Menge von zusätzlichen Bearbeitungsbereichen zugewiesen, die auf der Oberfläche der Grundkörper ansetzen. Diese Bereiche bezeichnet man als **Features**.

Die folgenden Features sind bei der Repräsentation des Werkstücks zugelassen:

- Nuten
- Gewinde
- Rundungen
- Fasen
- Freistiche
- Bohrungen

• Innenbearbeitungen

Features

relations: (parent: <ApplicationObject>)

Bedeutung der Relation:

parent: Komponente, auf der das Feature ansetzt

Nuten

Nuten (*grooves*) dürfen auf Zylindern, Kegelstumpfbereichen und Tonnenbereichen eingesetzt werden. Es gibt zwei Objektklassen, um Nuten darzustellen: Die Klassen **Groove** und *ShapedGroove*. Die Klasse **Groove**, die einfache Einstiche mit einem flachen Stechdrehmeißel repräsentiert, ist folgendermaßen definiert:

Groove

attributes: (	**extent:**	**<Number>**
	depth:	**<Number>**
	position:	**<Number>)**

Bedeutung der Attribute:

extent:	Breite e der Nut
depth:	Einstechtiefe d
position:	Abstand P vom Mittelpunkt der Nut zum linken Rand des Werkstücks

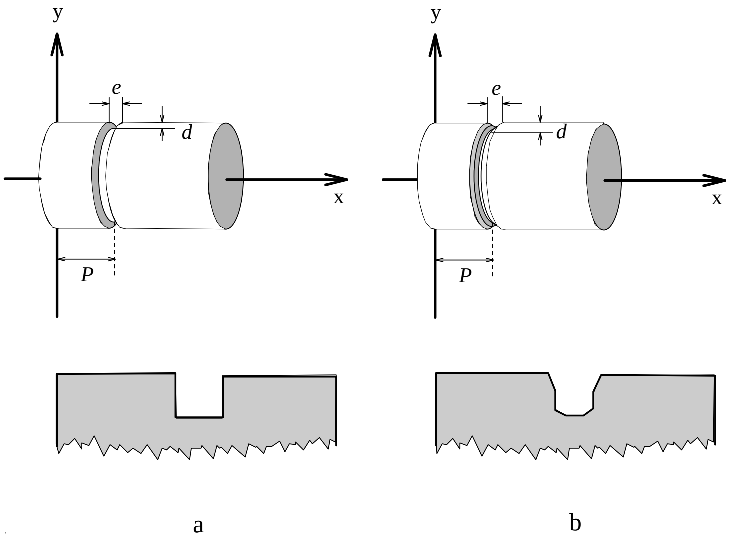


Abbildung A.8: a einfache Nut, b Nut, die aus **FeaturePrimitives** zusammengesetzt ist

Es wurde eine zweite Klasse definiert, um auch kompliziertere Einstiche darstellen zu können; eine solche Nut läßt sich nicht mehr allein mit den Werten für Breite, Tiefe und Position festlegen; es müssen zusätzlich Aussagen über die Form gemacht werden. Diese Nut ist aus einer Kombination von Grundformen zusammengesetzt. In der Praxis werden sie von Formdrehmeißeln durchgeführt, wobei keine klare Abgrenzung gegeben ist, ob noch von einer Nut gesprochen werden kann oder ob es sich um eine Freiform handelt und sie somit als eigene Bearbeitungskomponente angesehen werden muß.

Zum Entwerfen von solchen speziellen Nutformen stellt die CAD–Komponente des Planungssystems ein Hilfswerkzeug bereit, den sog. *Groove-Designer* (s. Kapitel ??).

ShapedGroove

relations: (	**featPrims:**	**<Array>**	)
attributes: (	**extent:**	**<Number>**	
	depth:	**<Number>**	
	position:	**<Number>**	)

Bedeutung der Relation:

featPrims: Liste mit speziellen primitiven Komponenten, aus denen eine selbstdefinierte Nut besteht

Gewinde

Gewinde (*threats*) sind nur auf Zylinderbereichen definiert; sie befinden sich am Ende des Werkstücks.

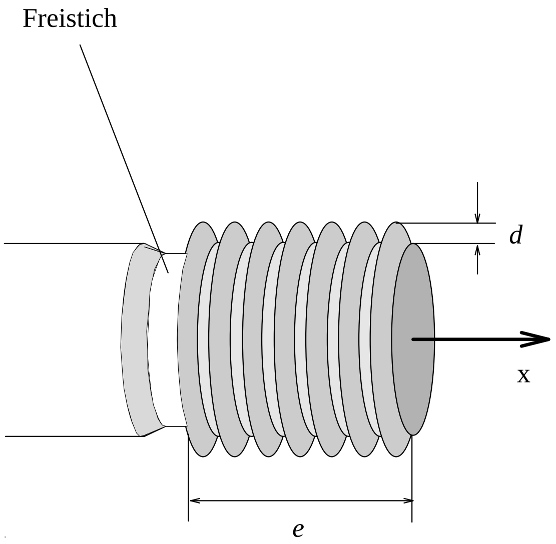


Abbildung A.9: Gewinde mit Freistich

Threat

attributes: (	**extent:**	**<Number>**	
	depth:	**<Number>**	
	side:	{ **left,right** }	)

Bedeutung der Attribute:

extent: Gesamtbreite e des Gewindes
depth: Gewindetiefe d
side: Die Seite der primitiven Komponente, auf der sich das Gewinde befindet

Fasen und Rundungen

Fasen (*slopes*) und Rundungen (*rounds off*) können nur auf Zylinder aufgesetzt werden. Sie dienen dazu, den Kantenwinkel von Bereichen zu verkleinern, um so eine Verletzungsgefahr zu reduzieren. Bei Fasen handelt es sich um Abschrägungen um einen Winkel, während Rundungen einen Kreisbogen mit einem bestimmten Radius darstellen. Wenn bei der Eingabe der Position einer Fase bzw. Rundung *both sides* angegeben wird, so erstellt das CAD–Programm für jede Seite ein seperates Featureobjekt. Fasen und Rundungen verkleinern den Endradius des Zylinderbereichs an der Seite, an der sie sich befinden.

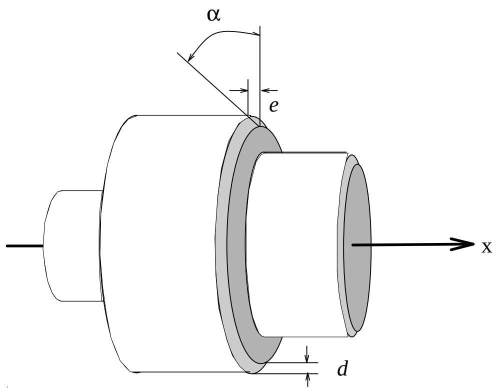


Abbildung A.10: Fasen auf Zylinderbereichen

Slope

attributes: (	**extent:**	**<Number>**
	depth:	**<Number>**
	angle:	**<Number>**
	side:	**{left,right})**

Bedeutung der Attribute:

extent:	Breite e der Fase
depth:	Tiefe d der Fase
angle:	Fasenwinkel α
side:	Seite des Zylinderbereichs, auf dem sich die Fase befindet

RoundOff

attributes: (	**extent:**	**<Number>**
	side:	**{left,right})**

Bedeutung der Attribute:

extent:	Krümmungsradius
side:	Seite des Zylinderbereichs, auf dem sich die Rundung befindet

Freistiche

Freistiche (*prickouts*) werden üblicherweise hinter Gewinden verwendet, wo sie eine Vertiefung darstellen, um Schmutz oder Spänen, die beim Aufschrauben einer Mutter auf das Gewinde anfallen, eine Ablaufmöglichkeit zu gewährleisten (Abb. ??.a). Freistiche werden Planflächen zugeordnet, wobei die Radien der Bereiche, die sich an die Planfläche anschließen, unterschiedlich sein müssen.

Es besteht die Möglichkeit, daß ein Freistich beide Bearbeitungsbereiche betrifft (Abb. ??.b). Aus diesem Grund wird unterschieden einmal zwischen horizontalen Ausdehnungen und Tiefen auf der ersten Komponente, und zum anderen zwischen vertikalen Ausdehnungen und Tiefen in der Planfläche, die zwischen den beiden Bereichen liegt. Freistiche werden mit einem speziellen Hilfsmittel, dem sog. *Prickout Designer* (s. ??), konstruiert, um so anwendungsspezifische Formen zu ermöglichen.

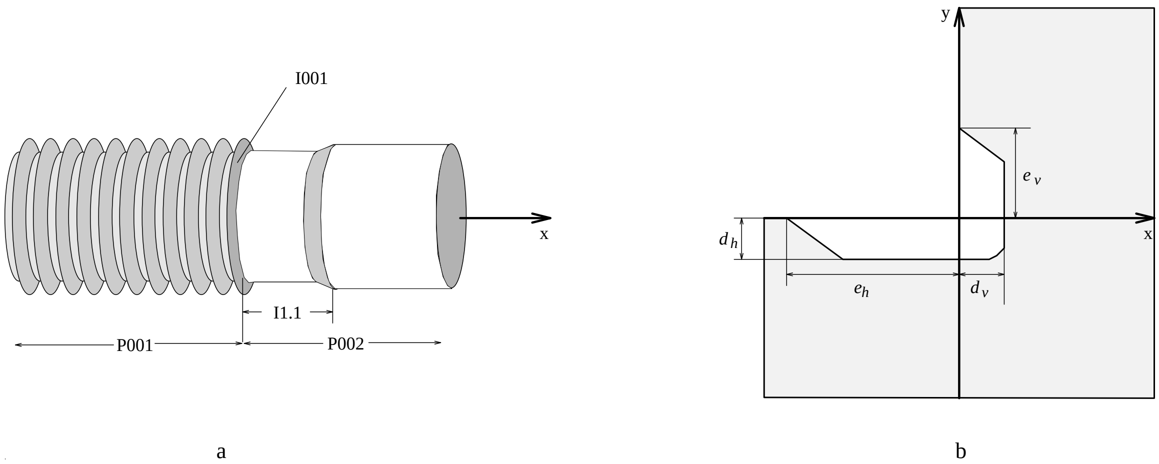


Abbildung A.11: a Freistich **I1.1** hinter Gewinde als Feature der Verbindungskomponente **I001**, b Freistich im Koordinatensystem

PrickOut

relations: (	**featPrims:**	**<Array>)**
attributes: (	**horDepth:**	**<Number>**
	vertDepth:	**<Number>**
	horExtent:	**<Number>**
	vertExtent:	**<Number>)**

Bedeutung der Relation:

featPrimitives: Liste mit speziellen primitiven Komponenten, aus denen ein Freistich zusammengesetzt ist (s. ??)

Bedeutung der Attribute:

horDepth:	Tiefe d_h des Freistichs auf dem Zylinder
vertDepth:	Tiefe d_v des Freistichs in der Planfläche
horExtent:	Breite e_h des Freistichs auf dem Zylinder
vertExtent:	Höhe e_v des Freistichs in der Planfläche

Bohrungen

Bohrungen (*bores*) befinden sich stets in der Symmetrieachse eines Werkstücks und werden den Stirnseiten zugeordnet. Sie werden mit einem Bohrer als Spezialwerkzeug ausgeführt. Es werden drei Arten von Bohrungen unterschieden:

- „Einfache“ Bohrungen

- Bohrungen mit mehreren Bohrdurchmessern (stufenweise Bohrung)
- Durchgängige Bohrungen

Im Falle einer durchgängigen Bohrung wird das Feature sowohl der linken als auch der rechten Stirnseite zugeordnet. Bei einer schrittweisen Bohrung muß der Bohrdurchmesser von außen nach innen hin abnehmen.

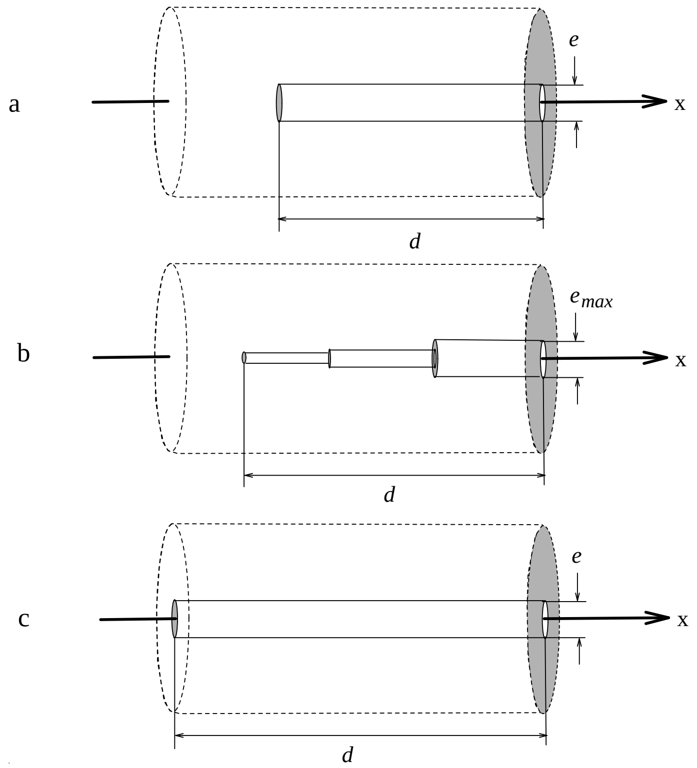


Abbildung A.12: a einfache Bohrung, b Bohrung mit Stufen, c durchgängige Bohrung

DrilledHole

attributes: (	**extent:**	**<Number>**
	depth:	**<Number>)**

Bedeutung der Attribute:

extent: Ausdehnung (Durchmesser) e der Bohrung
depth: Tiefe d der Bohrung

SteplikeDrilledHole

relations: (	**drilledHoles:**	**<OrderedCollection>)**
attributes: (	**extent:**	**<Number>**
	depth:	**<Number>)**

Bedeutung der Relation:

drilledHoles: Liste der Einzelbohrungen, die das Feature bilden.

Bedeutung der Attribute:

extent:	Maximale Ausdehnung e_{max} der Bohrung
depth:	Gesamttiefe d der Bohrung

DrilledThrough

attributes: (	**extent:**	**<Number>**
	depth:	**<Number>)**

Bedeutung der Attribute:

extent:	Durchmesser e der Bohrung
depth:	Tiefe d der Bohrung (= Länge des Werkstücks)

Innenbearbeitungen

Innenbearbeitungen setzen, ebenso wie Bohrungen, auf den Stirnseiten des Werkstücks an. Sie werden in der Praxis mit sog. Innendrehmeißeln durchgeführt. Innenbearbeitungen setzen sich aus den primitiven Grundkomponenten zusammen, aus denen auch ein Werkstück besteht, also Zylinder-, Kegelstumpf- und Tonnenbereichen; konstruiert werden sie mit einem speziellen Hilfswerkzeug, dem *Innerwork-Designer* (s. ??). Durch Angabe der Position einer Innenbearbeitung besteht die Möglichkeit, ihren Ursprung auf der y–Achse der Stirnseite nach außen zu verschieben, um sie so um einen Kern herum anzusetzen, der bei der Bearbeitung ausgespart bleibt (Abb. ??.b).

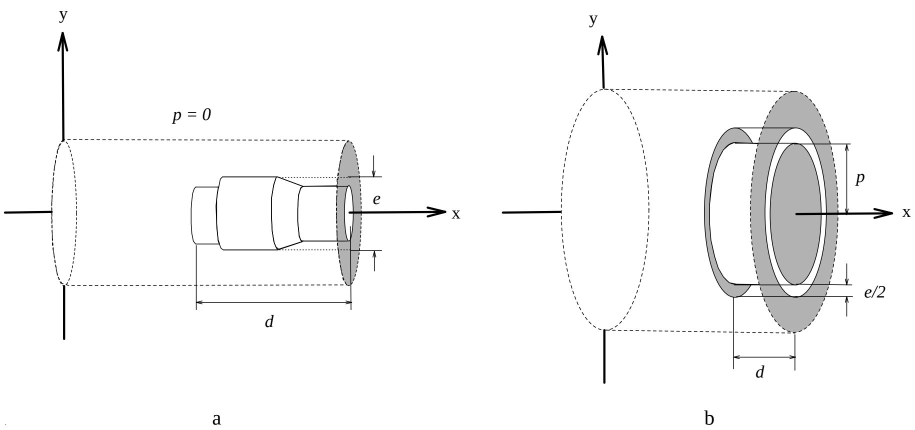


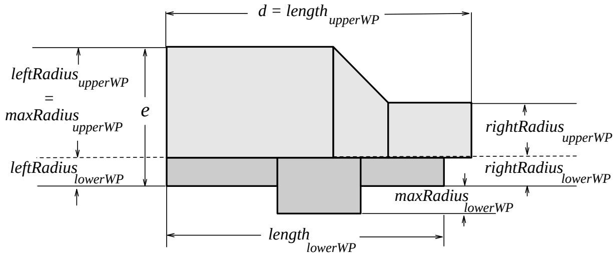
Abbildung A.13: a Innenbearbeitung aus primitiven Komponenten, b Ring als Innenbearbeitung

DrilledHole

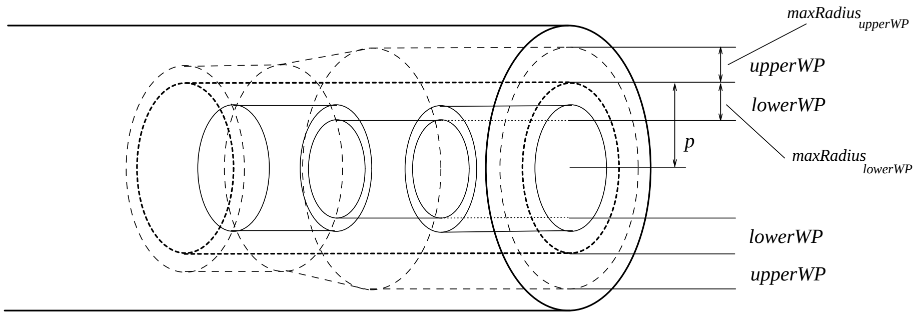
relations: (	**upperWP:**	**<Workpiece>**
	lowerWP:	**<Workpiece>)**
attributes: (	**extent:**	**<Number>**
	depth:	**<Number>**
	position:	**<Number>)**

Bedeutung der Relationen:

upperWP:	Obere Hälfte der Innenbearbeitung
lowerWP:	Untere Hälfte der Innenbearbeitung



a



b

Abbildung A.14: a Innenbearbeitung aus *upperWP* und *lowerWP*, b Abbildung von a auf dem Werkstück

Bedeutung der Attribute:

extent:	Maximaler Durchmesser e der Innenbearbeitung
depth:	Gesamttiefe d der Innenbearbeitung
position:	Ursprung der Innenbearbeitung auf der y–Achse

Bemerkung:

Die Unterteilung der Innenbearbeitung in **upperWP** und **lowerWP** macht nur dort Sinn, wo der maximale Radius des **lowerWP** kleiner ist als die y–Verschiebung **position**; im Falle **position** = 0 gilt automatisch **upperWP** = (gespiegeltes) **lowerWP**; im Falle, daß der Radius des **lowerWP** kleiner ist als die y–Verschiebung, kommt es beim Drehen zu Konturüberschneidungen, so daß dieser Fall ausgeschlossen werden muß.

A.1.13 Toleranzen

Die Toleranzangaben des Werkstücks setzen jeweils auf einem Bearbeitungselement an. Die der jeweiligen Toleranz zugeordnete Toleranzzone läßt sich aus Anhang ?? ablesen, und die Zuordnung der Toleranzwerte und Nennmaßbereiche zu den ISO–Grundtoleranzen ist in Anhang ?? gegeben. Insgesamt sind für achsensymmetrische Drehteile die folgenden Toleranzen definiert:

- Formtoleranzen
  - Geradheit

- Ebenheit
- Rundheit
- Zylinderform

- Lagetoleranzen
  - Parallelität
  - Rechtwinkligkeit
  - Konzentrizität
  - Planlauf
  - Rundlauf

Der Nennmaßbereich wird im Falle der Ebenheit und des Planlaufs durch die Länge der Komponente gebildet, in den anderen Fällen durch ihren Durchmesser. Die Lagetoleranzen benötigen (mindestens) einen Bearbeitungsbereich, auf den sich ihre Zahlangaben beziehen; im Falle der Rundlaufabweichung wird auf zwei Komponenten Bezug genommen.

Tolerance

relations: (	**parent:**	**<PrimitiveComponent>)**
attributes: (	**value:**	**<Number>**
	nominalSize:	**<Number>**
	it:	**<Number>)**

Bedeutung der Relation:

parent: Element, für das die Toleranz gilt

Bedeutung der Attribute:

value:	Zahlwert der tolerierten Eigenschaft (in mm)
nominalSize:	Nennmaß des tolerierten Elements
it:	ISO–Grundtoleranzwert

Es werden weiterhin zwei Klassen definiert, um Form– und Lagetoleranzen unterscheiden zu können:

ToleranceOfForm

ToleranceOfPositon
relations: (ReferenceElement: <ProcessingElement>)

Bedeutung der Relation:

ReferenceElement: Das Bearbeitungselement, relativ zu dem die Toleranz definiert wird

Für jede Toleranzart wird eine eigene Klasse definiert, die Unterklasse der jeweiligen Toleranzeinteilung **ToleranceOfForm** bzw. **ToleranceOfPosition** ist.

Straightness
Evenness
Roundness
CylinderShape
Parrallelism
Rectangularity
Concentricity
RunOut
TrueRunning
relations: (ReferenceElement2: <ProcessingElement>)

A.1.14 Das Werkstück

Die Klasse **Workpiece**, die das Werkstück repräsentiert, ist direkte Unterklasse der Klasse **ApplicationObject**. Werkstücke werden sowohl in der Fallbasis als auch bei der Speicherung durch das CAD–System durch Instanzen der Klasse **Workpiece** dargestellt.

Workpiece

relations: (	**component:**	**<ComplexComponent>)**
	leftFront:	**<FrontConnElement>**
	rightFront:	**<FrontConnElement>**
	clambingDevices:	**<OrderedCollection>**
	clambingDevicesTurned:	**<OrderedCollection>**
	slots:	**<DescriptionDictionary>**
	neededClambingLength:	**<OrderedCollection (<Boolean><Boolean>)>**
	possibleClambingLength:	**<OrderedCollection (<Boolean><Boolean>)>)**
attributes: (	**length:**	**<Number>**
	leftRadius:	**<Number>**
	rightRadius:	**<Number>**
	maxRadius:	**<Number>**
	surfaceFinishRow:	**<Number>**
	isAnalysed:	**<Boolean>**
	isStructured:	**<Boolean>)**

Bedeutung der Relationen:

component:	Die komplexe Komponente, die das gesamte Werkstück umfaßt
leftFront:	Die linke Stirnseite des Werkstücks
rightFront:	Die rechte Stirnseite des Werkstücks
clambingDevices:	Liste der Spannmittel, die das Werkstück in Normallage spannen (→ ??)
clambingDevicesTurned:	Entsprechende Liste der Spannmittel für das umgedrehte Werkstück
slots:	Slots zur Aufnahme der zu erfüllenden Ziele; diese Relation wird als Parameter dem semantischen Netzwerk übergeben (→ ??)
neededClambingLength:	Benötigte Einspannlängen, um das Werkstück in fliegender Spannung mit Backen sicher einzuspannen (→ ??)
possibleClambingLength:	Mögliche Einspannlängen für Spannbacken (→ ??)

Bedeutung der Attribute:

length:	Gesamtlänge des Werkstücks
leftRadius:	Radius am linken Ende des Werkstücks
rightRadius:	Radius am rechten Ende des Werkstücks
maxRadius:	maximaler Radius des Werkstücks
surfaceFinishRow:	Die Reihe, auf die sich die Oberflächensymbole beziehen (→ ??)
isAnalysed:	Flag, das anzeigt, ob das Werkstück bereits analysiert wurde
isStructured:	Flag, das anzeigt, ob die Relation **component** als Baumstruktur vorliegt (→ ??)

A.2 Die Benutzeroberfläche

Zum Starten des Planungssystems gibt es eine spezielle Klasse **StartPlanner**; der Aufruf erfolgt durch die Klassenmethode **open**:

StartPlanner open

Es erscheint ein Fenster mit drei Schaltern (Abb. ??). Der oberste Schalter *Start Planner* öffnet das Hauptfenster des Planungssystems, der Schalter *Start CAD–System* aktiviert den Workpiece–Designer (s. ??), und der untere Schalter *Start Tree View* lädt eine *Tree View* (vgl. *Map-Viewer* → ??).

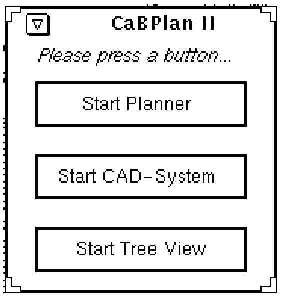


Abbildung A.15: Start-Window für den Programmaufruf

Im Folgenden werden die verschiedenen Komponenten der Benutzeroberfläche vorgestellt.

A.2.1 Das Auswählen von Elementen

Für eine Reihe von Anwendungen ist es notwendig, eine oder mehrere Komponenten des aktuellen Werkstücks auszuwählen. Dies geschieht auf folgende Arten:

- Öffnen eines Auswahlfensters
  Einzelne Komponenten (Bearbeitungskomponenten, Verbindungselemente) bzw. das ganze Werkstück können dadurch gewählt werden, daß ein Auswahlfenster geöffnet wird, welches die gewünschten Elemente umfaßt. Ein Auswahlfenster wird geöffnet, indem
  - der linke Mausknopf in der ersten Ecke des gewünschten Fensters gedrückt wird
  - er gedrückt bleibt, während das Fenster aufgezogen wird
  - er im zweiten Eckpunkt des Fensters losgelassen wird

  Es werden stets diejenigen Komponenten gewählt, die vollständig innerhalb des Bearbeitungsfenster liegen.

- Anklicken eines Bearbeitungselements
  Ein einzelnes Bearbeitungselement kann durch Anklicken mit der linken Maustaste gewählt werden.

- Über Pull-down-Menüs

  Es besteht die Möglichkeit, über die Option select ▷ des Pull-down-Menüs Elemente auszuwählen. Dabei ist zu unterscheiden, ob
  - noch kein Element gewählt ist
    In diesem Fall gibt es die Optionen, die linke bzw. rechte Stirnseite oder das gesamte Werkstück auswählen.

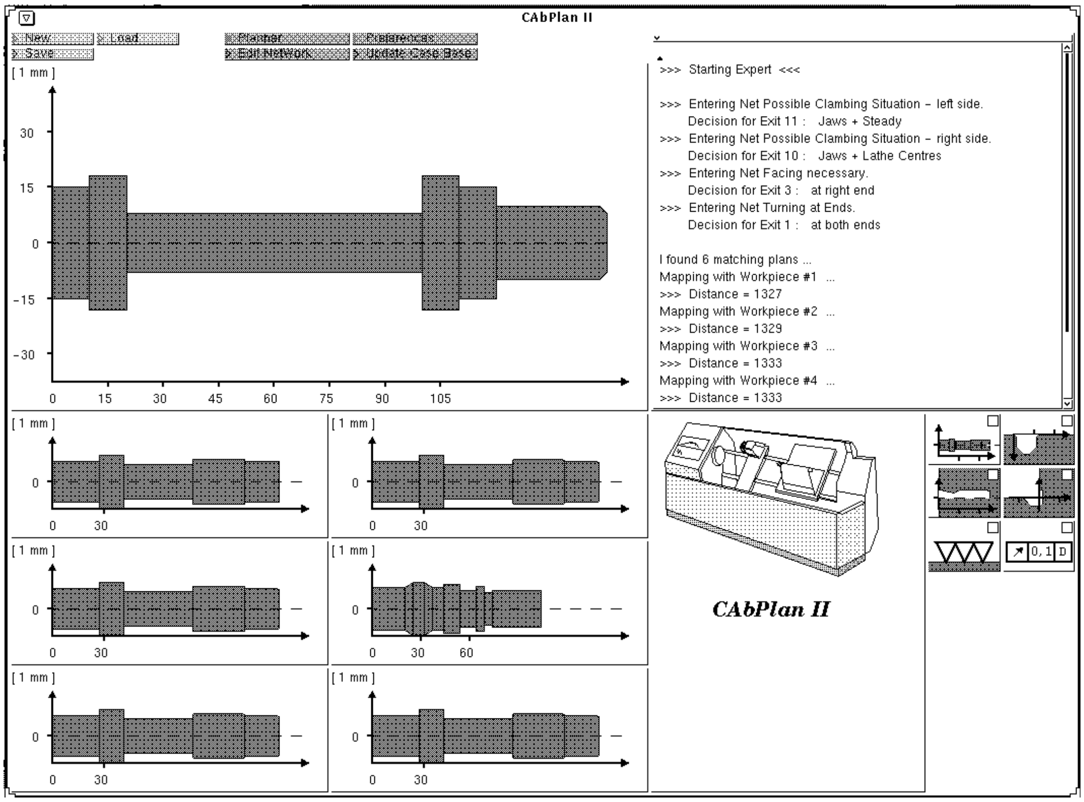


Abbildung A.16: Benutzeroberfläche des Planungssystems

ganz oben links:	Menüleiste	oben rechts:	Textdisplay des Planungs-systems
oben links:	CAD–Komponente		
unten links:	Displayfenster des Planungssystems	unten mitte rechts:	Multifunktionsfenster
		unten rechts:	Icons für Hilfswerkzeuge

– ein Bearbeitungselement gewählt ist
Falls schon ein Bearbeitungselement gewählt ist, werden Optionen zur Verfügung gestellt, das linke bzw. rechte Verbindungselement, die Komponente, die in der Relation **parent** gespeichert ist, oder ein oder mehrere Features auszuwählen (Abb. ??).

– ein Verbindungselement gewählt ist
In diesem Fall können über ein Pull-down-Menü ein oder mehrere Features des Verbindungselements gewählt werden

A.2.2 Die Menüleiste

- Schalter für Werkstückzugriffe
Diese Schalter sind Bestandteile des CAD–Systems zur Konstruktion von Werkstücken. Beim Speichern bzw. Laden von Werkstücken ist zu beachten, daß diese in der Klassenvariablen **Workpieces** (und nicht in externen Dateien) gespeichert werden.

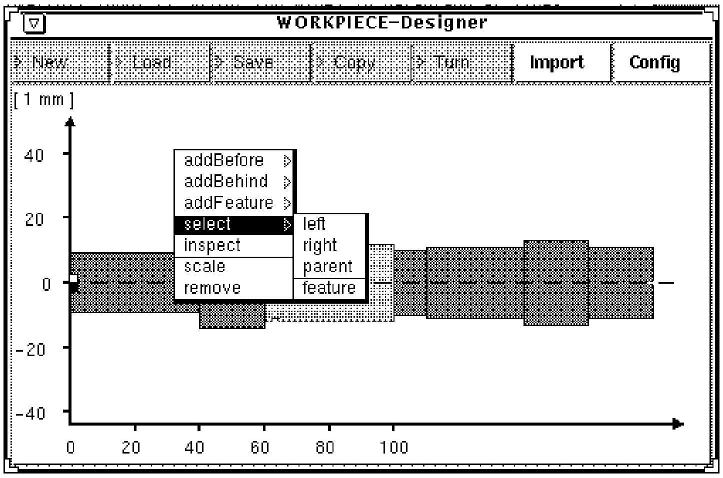


Abbildung A.17: Pull-down-Menü zum Auswählen von Elementen

- $\boxed{\triangleright\ \textit{New}}$
  Legt ein neues Werkstück an.
- $\boxed{\triangleright\ \textit{Load}}$
  Lädt ein Werkstück aus der Klassenvariablen **Workpieces**.
- $\boxed{\triangleright\ \textit{Save}}$
  Speichert das aktuelle Werkstück in der Klassenvariablen **Workpieces** ab. Es muß der Namen angegeben werden, unter dem das Werkstück abgespeichert werden soll. Dieser kann sich von dem aktuellen Namen des Werkstücks unterscheiden. Existiert ein Werkstück gleichen Namens in der Werkstücksammlung, so wird dieses überschrieben.

• Schalter für Systemaufrufe

- $\boxed{\triangleright\ \textit{Planner}}$
  Startet das Planungssystem (s. ??). Es muß ein aktuelles Werkstück in der CAD–Komponente vorhanden sein.
- $\boxed{\triangleright\ \textit{Edit Network}}$
  Hier werden Methoden zur Verfügung gestellt, um das Netzwerk zu editieren. Das Netzwerk wird in Abschnitt ?? beschrieben.
  * $\boxed{\text{Add Semantical Net}}$
    Es wird dem Netzwerk ein weiteres semantisches Netz hinzugefügt. Zunächst muß der Name des neuen Netzes angegeben werden. Dann wird dieses Netz Schritt für Schritt erzeugt:
    1. Der aktuelle Knoten wird auf die Wurzel des Netzes gesetzt.
    2. Die Bedingung des aktuellen Knotens wird als Instanz der Klasse **BlockClosure** eingegeben; Übergabeparameter ist die Relation **slots** des Werkstücks.
       Bei syntaktisch falscher Eingabe wird ein Hilfstext zur Verfügung gestellt.
    3. Eine Beschreibung für den aktuellen Knoten kann eingegeben werden.
    4. Die Art der Nachfolger des aktuellen Knotens wird bestimmt (Knoten oder Blätter)
    5. Der aktuelle Knoten wird auf den linken Nachfolger gesetzt, falls dieser ein Knoten ist. In diesem Fall wird rekursiv bei 2. weitergemacht.
       Ist der linke Nachfolger ein Blatt, dann wird nach dessen Inhalt gefragt.

6. Der aktuelle Knoten wird auf den rechten Nachfolger gesetzt, falls dieser ein Knoten ist. In diesem Fall wird rekursiv bei 2. weitergemacht.
   Ist der rechte Nachfolger ein Blatt, dann wird nach dessen Inhalt gefragt.

Das neue Netz wird an letzter Stelle im semantischen Netzwerk eingefügt; die Verbindungspointer innerhalb des Netzwerks werden aktualisiert.

* Remove Semantical Net

  Ein semantisches Netz wird aus dem Netzwerk entfernt, die Verbindungspointer des Netzwerks werden aktualisiert.

Beachte:

Wird das Netzwerk um ein semantisches Netz erweitert oder wird ein Netz gelöscht, so muß die Struktur der Fallbasis mit ▷ *Update Case Base* an diese Veränderung angepaßt werden; **dies hat zur Folge, daß alle bisher gespeicherten Fälle der Fallbasis gelöscht werden!** Diese Editieroperationen auf dem Netzwerk sollten daher nur mit größter Vorsicht eingesetzt werden.

* Add Node

  Es kann an beliebiger Stelle im Netzwerk ein Blatt durch einen weiteren Knoten ersetzt werden, d.h. der Baum kann nach unten erweitet werden. Zunächst muß man das semantische Netz angeben, in das ein neuer Knoten eingefügt werden soll. Dann wird das Blatt ausgewählt, daß durch den Knoten ersetzt werden soll. Anschließend wird nach dem Entscheidungsblock, einer Beschreibung des Entscheidungsblocks und nach den Inhalten der beiden Blättern gefragt, die unter dem neuen Knoten liegen.

* Replace Block

  Ein bestehender Knoten wird durch einen anderen ersetzt. Es muß das semantische Netz angegeben werden, in dem ein Knoten ersetzt werden soll. Anschließend wird der Knoten gewählt, der zu ersetzen ist. Nun kann der neue Entscheidungsblock und eine neue Beschreibung des Knotens eingegeben werden.

– ▷ *Preferences*

  Die Voreinstellungen werden in der Klassenvariablen **Preferences** der Klasse **System** gespeichert. Je nach ihrer aktuellen Belegung erscheint die entsprechende Option im Menü.

  * system ▷

    Die Belegungen unter dieser Option beziehen sich auf die Gestaltung der Benutzeroberfläche.

    · display clambing devices / do not display clambing devices

      Schalter, ob die vom Planungssystem vorgeschlagenen Spannmittel graphisch angezeigt werden.

    · fill clambing devices / do not fill clambing devices

      Werden die Spannmittel graphisch angezeigt, so gibt diese Voreinstellung an, ob sie als gefüllte Polylinie dargestellt werden, oder ob nur die Umrandungslinie gezeichnet wird

    · display surface finishes / do not display surface finishes

      Dieser Schalter entscheidet, ob die Oberflächengütensymbole in der Zeichnung erscheinen sollen. Ist dies der Fall, so wird auch die Reihe aus Tabelle ?? angegeben, auf die sich die Symbole beziehen.

    · filled background / white background

      Schaltet zwischen einem gefüllten Hintergrund[1] und einem weißen Hintergrund.

[1]In Abhängigkeit vom verwendeten Monitor werden als Hintergrundfarben entweder Grautöne bei einem Graustufenmonitor oder verschiedene Farben bei einem Farbmonitor verwendet

* mapping ▷

  Hier werden eine Reihe von Voreinstellung zum Mapping gesetzt.

  · mapping algorithm ▷

    Hiermit wird der Mapping–Algorithmus festgelegt. Man kann sich entscheiden zwischen *Structure Preserving Mapping*, *Strictly Structure Preserving Mapping* oder *Tai–Mapping*.

  · change deletion costs

    Die Kosten der Löschoperationen beim Mapping werden hier für die verschiedenen Objekte verändert.

  · change insertion costs

    Die Kosten der Einfügeoperationen beim Mapping werden hier verändert.

  · objects to be mapped ▷

    Hier werden die Objekte ausgewählt, die miteinander gemapt werden. Man kann sich entscheiden,

    - Komponenten
    - Features
    - Toleranzen
    - Oberflächengüten

    des Werkstücks (in beliebiger Kombination miteinander) in das Mapping mit eingehen zu lassen.

    Die folgenden drei Voreinstellungen beziehen sich auf den *Map Viewer* (s. ??), der eine graphische Darstellung des Mappings ermöglicht.

  · layout vertical / layout horizontal

    Schalter, ob im *Map Viewer* die beiden Baumstrukturen übereinander oder nebeneinander angeordnet werden.

  · displayed text ▷

    Hier wird angegeben, welche Attribute im *Map Viewer* eingeblendet werden:

    - Der Name des Entities
    - Der Typ des Entities
    - Name und Typ des Entities

  · mapping displayed / mapping not displayed

    Mit diesem Schalter wird der *Map Viewer* an- bzw. ausgeschaltet.

* reset all

  Mit dieser Option werden die Standard–Voreinstellungen eingeschaltet.

* inspect preferences

  Es wird ein **Inspector** für die Klassenvariable **Preferences** geöffnet.

– ▷ *Update Case Base*

  Die Struktur der Fallbasis wird aktualisiert; dies wird notwendig, wenn semantische Netze hinzukommen oder gelöscht werden. Eine Aktualisierung der Fallbasis löscht alle gespeicherten Fälle!

A.2.3 Die CAD–Komponente

Die ins Planungssystem integrierte CAD–Komponente ist ein *Workpiece Designer* [13]; die Beschreibung erfolgt in Abschnitt ??.

A.2.4 Menü des Fensters zum Anzeigen der indizierten Werkstücke

Die Werkstücke, die in der Fallbasis unter dem vom semantischen Netzwerk gewählten Index gespeichert sind, werden im unteren linken Bereich der Oberfläche dargestellt. Es werden folgende

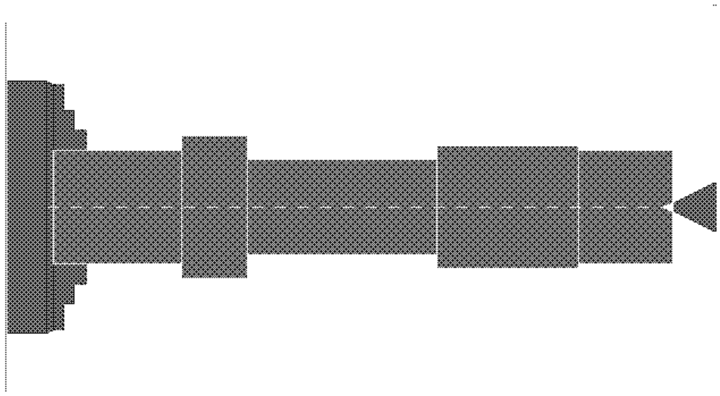


Abbildung A.18: Darstellung der Spannmittel (gefüllt)

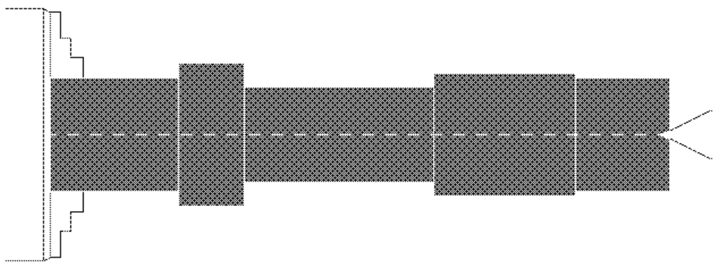


Abbildung A.19: Darstellung der Spannmittel (nicht gefüllt)

Optionen angeboten:

- select case
  Der Fall wird ausgewählt
- mapping
  Der aktuelle Fall wird mit dem Problem gemappt
- remove case
  Der Fall wird aus der Fallbasis gelöscht
- inspect case
  Es wird ein Inspector für den Fall geöffnet

A.2.5 Hilfswerkzeuge für den Werkstückentwurf

Folgende Hilfswerkzeuge können über Anklicken der entsprechenden Icons aufgerufen werden:

- *Workpiece Designer*
  zum Entwickeln von Werkstücken

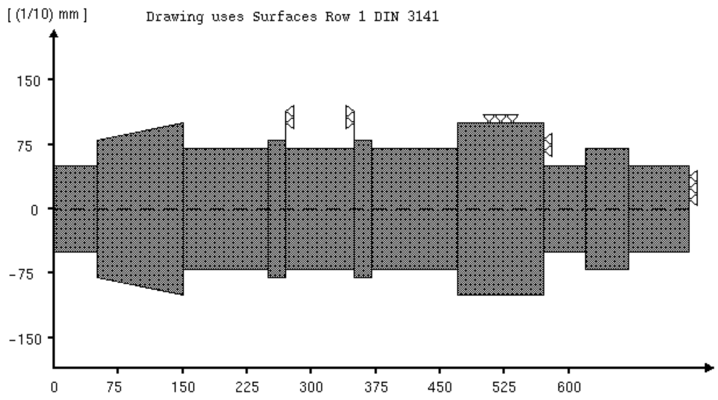


Abbildung A.20: Darstellung von Oberflächensymbolen in der Zeichnung

- *Groove Designer*
  zum Entwerfen von Nuten
- *Innerwork Designer*
  zum Entwerfen von Innenbearbeitungen
- *Prickout Designer*
  zum Entwerfen von Freistichen
- Oberflächengüten–Editor
- Toleranzen–Editor

Bei Aktivierung vom *Workpiece Designer*, *Groove Designer*, *Innerwork Designer* und *Prickout Designer* werden eigene Fenster geöffnet, Oberflächengüteneditor und Toleranzeneditor werden im Multifunktionsfenster des Planungssystems dargestellt.

Der *Workpiece Designer*

Der *Workpiece Designer* dient dazu, neue Werkstücke anzulegen und zu konstruieren bzw. existierende zu modifizieren. Dies geschieht über Editierfunktionen, die mit Hilfe von Pull–Down–Menüs aufgerufen werden. Je nachdem, welche Elemente des aktuellen Werkstücks gewählt sind, sind unterschiedliche Editierfunktionen möglich:

- Gesamtes Werkstück ist gewählt bzw. nichts ist gewählt
  - $\boxed{\text{addFirst} \triangleright}$
    Als erstes Element des Werkstücks wird eine Komponente eingefügt. Mögliche Komponenten sind:
    * Zylinder
      Ein Zylinderbereich wird als erste Komponente des Werkstücks eingefügt; er wird über Länge und Radius definiert.
    * Kegelstumpfbereich
      Als erste Komponente des Werkstücks wird ein Kegelstumpfbereich eingefügt; der Kegelstumpfbereich wird über linken Radius, rechten Radius und Länge definiert.
    * Tonnenbereich
      Es wird als erste Komponente ein Tonnenbereich eingefügt; der Tonnenbereich wird festgelegt über die Länge, den linken Radius, den rechten Radius, den Konturradius der Krümmung sowie über die Art der Krümmung (konvex oder konkav).

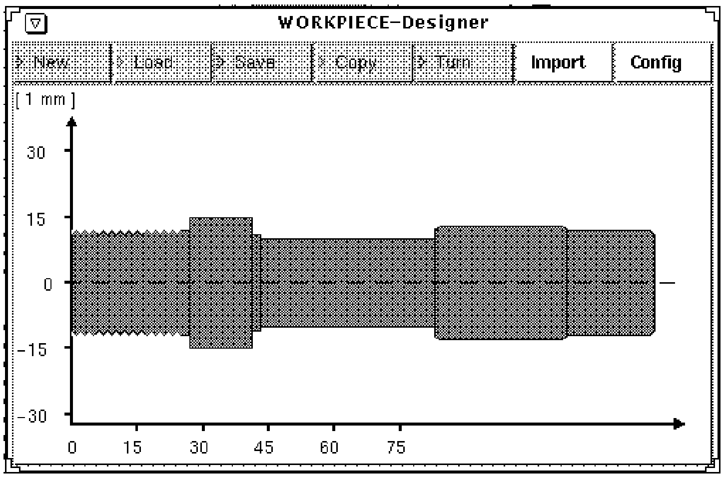


Abbildung A.21: Oberfläche des *Workpiece Designers*

  * Aktuelle Komponente
    Die aktuelle Komponenete (= eine Kopie des gesamten Werkstücks) wird am linken Ende eingefügt.
- addLast ▷
  Diese Funktion ist äquivalent zu der Funktion addFirst ▷ , außer daß hier die neue Komponente als letztes Element des Werkstücks eingefügt wird.
- select ▷
  Es werden können folgende Komponenten gewählt werden:
  * linke Stirnseite
  * rechte Stirnseite
  * gesamtes Werkstück
- inspect ▷
  Öffnet einen Inspektor für das Werkstück
- scale ▷
  Skaliert das Werkstück horizontal oder vertikal neu; die entsprechenden x– bzw. y–Werte werden mit dem spezifizierten Skalierfaktor multipliziert.
- WuD ▷
  Stellt die Dienstfunktionen zum Neuanlegen, Laden und Speichern des Werkstücks zur Verfügung.

• Ein Bearbeitungselement ist gewählt
  - addBefore ▷
    Ähnlich der Funktion addFirst ▷, wenn das Werkstück gewählt ist, außer daß das einzufügende Element vor dem gewählten Bearbeitungselement eingefügt wird
  - addBehind ▷
    Diese Funktion ist äquivalent zu der Funktion addBefore ▷ , außer daß hier die neue Komponente rechts des aktuellen Bearbeitungselementes eingefügt wird.
  - addFeature ▷
    Dem aktuellen Bearbeitungselement kann ein Feature hinzugefügt werden; die möglichen Features sind abhängig von der Art des Bearbeitungsbereichs: Einem Zylinderbereich können Nuten, Fasen, Rundungen und Gewinde, Kegelstumpfbereichen und Tonnenbereichen nur einfache Nuten zugefügt werden.

- $\boxed{\text{select} \triangleright}$
  Es werden können folgende Komponenten gewählt werden:
  * linkes Verbindungselement
  * rechtes Verbindungselement
  * *parent*–Komponente des Werkstücks
  * Features
  * Toleranzen
- $\boxed{\text{inspect}}$
- $\boxed{\text{scale}}$
- $\boxed{\text{remove}}$
  Löscht die aktuelle Komponente.

• Eine Planfläche ist gewählt

- $\boxed{\text{addFeature} \triangleright}$
  Fügt der Planfläche ein Feature hinzu. Erlaubte Features für Planflächen sind Nuten, Freistiche und Innenbearbeitungen.
- $\boxed{\text{selectFeature}}$
  Es können ein Feature bzw. alle Features der Planfläche gewählt werden.
- $\boxed{\text{selectTolerance}}$
  Die Toleranz der Planfläche wird gewählt.
- $\boxed{\text{show}}$
  Schalter, um die Planfläche durchgängig graphisch sichtbar zu machen.
- $\boxed{\text{hide}}$
  Schalter, um die Planfläche auszublenden.
- $\boxed{\text{inspect}}$

• Eine Stirnseite ist gewählt

- $\boxed{\text{addFeature} \triangleright}$
  Fügt der Stirnseite ein Feature hinzu. Erlaubte Features für Stirnseiten sind Nuten, Innenbearbeitungen, Bohrungen und Zentrierbohrungen.
- $\boxed{\text{selectFeature}}$
- $\boxed{\text{selectTolerance}}$
- $\boxed{\text{inspect}}$

• Mehrere Bearbeitungselemente sind gewählt

- $\boxed{\text{remove}}$

• Ein Feature bzw. eine Toleranz ist gewählt

- $\boxed{\text{inspect}}$
- $\boxed{\text{remove}}$

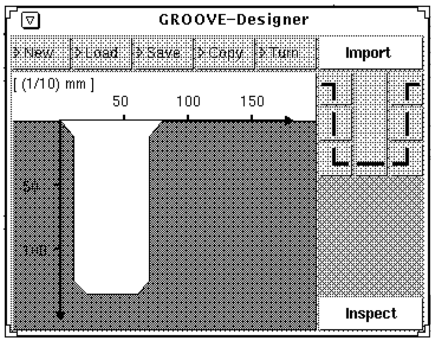


Abbildung A.22: Oberfläche des *Groove Designers*

Der *Groove Designer*

Der *Groove Designer* ist ein Hilfsmitel, um komplexere Nuten zu entwerfen. Eine Nut wird repräsentiert durch 7 primitive Bereiche,die alle einzeln editierbar sind. Im einzelnen sind dies 4 Eckenbereiche und 3 Längsbereiche (Abb. ?? rechts).

Für die Eckenbereiche gibt es drei Typen:

- „Normal" (d.h. Ecke mit Eckenwinkel 90^o)
- Konisch (Eckenwinkel $\neq 90^o$)
- Toroid (Tonnenförmig)

Ist der Eckenbereich konisch, so müssen Breite und Eckenwinkel spezifiziert werden; ist er toroid, benötigt man Breite, Tiefe und Rundungsradius.

Der horizontale Längsbereich wird durch Angabe seiner Breite spezifiziert; die beiden vertikalen Längsbereiche können als „normale" Bereiche mit senkrechten Seitenwänden oder als konische Bereiche mit nicht–senkrechten Seitenwänden definiert werden. Beim senkrechten Längsbereich genügt zur Spezifizierung die Angabe der Tiefe, im anderen Fall muß zusätzlich der Anstiegswinkel angegeben werden.

Es stehen im Pop–Up–Menü folgende Optionen zur Verfügung:

- Show Coordinate System / Hide Coordinate System
  Schaltet Koordinatensystem an bzw. aus.
- Show Raster / Hide Raster
  Schaltet Raster an bzw. aus.
- Change Distance between Tick Marks
  Spezifiziert die Dichte des Rasters

Der *Innerwork Designer*

Der *Innerwork Designer* ist ein Hilfsmittel zum Konstruieren von Innenbearbeitungen. Innenbearbeitungen sind zusammengesetzt aus primitiven Bearbeitungskomponenten ohne Verbindungselemente, die Editierfunktionen sind ähnlich zu denen des *Workpiece Designers*. Innenbearbeitungen setzen sich aus einem oberen und einem unteren Teil zusammen (s. ??). Folgende Pop–Up–Menüs stehen für die Bearbeitung zur Verfügung:

- kein Bearbeitungselement ist gewählt
  - addFirst ▷

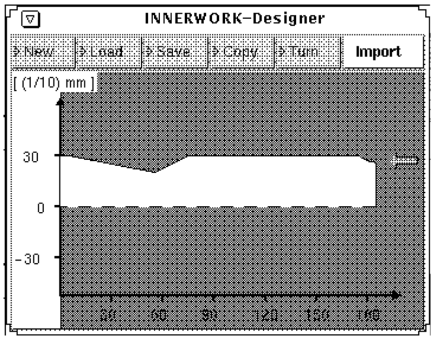


Abbildung A.23: Oberfläche des *Innerwork Designers*

- addLast ▷
- upper
  Wechselt auf den oberen Teil der Innenbearbeitung.
- lower
  Wechselt auf den unteren Teil der Innenbearbeitung.
- inspect
- CoordSyst
  * Show Coordinate System / Hide Coordinate System
  * Show Raster / Hide Raster
  * Change Distance between Tick Marks

• es ist ein Bearbeitungselement gewählt

- addBefore ▷
- addBehind ▷
- select
  * left
  * right
  * parent
- remove

Der *Prickout Designer*

Der *Prickout Designer* dient zum Konstruieren von Freistichen. Ein Freistich besteht aus 9 primitiven Bereichen (Abb. ?? rechts), die einzeln editiert werden können. Er erstreckt sich über zwei Bearbeitungsbereiche, wobei er als Feature der Planseite des Bearbeitungsbereichs mit dem größeren Durchmesser zugeordnet wird. Das Editieren der einzelnen Bereiche des Freistichs erfolgt entsprechend dem *Groove Designer*.

Der Oberflächengüten–Editor

Beim Aufruf des Oberflächengüteneditors muß mindestens eine Komponente des Werkstücks gewählt sein. Durch Anklicken des entsprechenden Symbols wird den gewählten Komponenten die

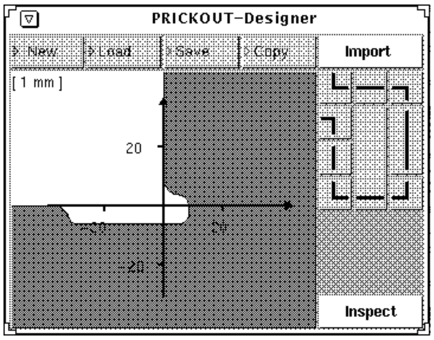


Abbildung A.24: Oberfläche des *Prickout Designers*

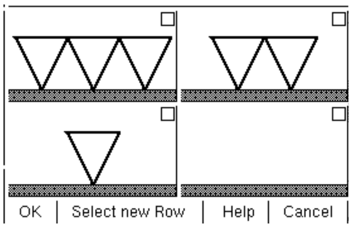


Abbildung A.25: Oberflächengüteneditor

gewünschte Oberflächengüte zugewiesen (vgl. ??). Wird über die Schaltfläche *Select New Row* eine neue Spalte in der Tabelle für die Oberflächengütensymbole gewählt, so gilt diese für die gesamte Zeichnung; bereits eingetragene Oberflächengüten ändern sich also.

Durch Anklicken der Schaltfläche *Help* wird die Tabelle für Oberflächengütensymbolen dargestellt.

Der Toleranzen–Editor

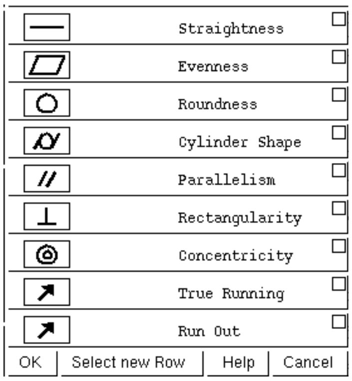


Abbildung A.26: Menü zur Toleranzeneingabe

Der Toleranzeneditor weist gewählten Komponenten spezielle Toleranzwerte zu. Die Art der Toleranz wird durch Anklicken des entsprechenden Icons bestimmt. Formtoleranzen werden durch ihren numerischen Wert spezifiziert, Lagetoleranzen benötigen zusätzlich die Komponente(n), auf die sie sich beziehen. Die ISO–Grundtoleranzen werden anschließend vom System berechnet.

A.2.6 Der Map–Viewer

Mit dem *Map–Viewer* steht ein Hilfswerkzeug zur Verfügung, mit dessen Hilfe u.a. Mappings zwischen Baumstrukturen graphisch dargestellt werden können. Zusätzlich stehen eine Reihe von Optionen zum Editieren von Baumstrukturen zur Verfügung.

Wählen von Knoten

Ein einzelner Knoten wird durch Anklicken mit der linken Maustaste gewählt. Sollen mehr als ein einzelner Knoten gewählt werden, so gibt es zwei Möglichkeiten:

- Drücken der < *Shift* >–Taste, während Knoten gewählt werden
- Öffnen eines Auswahlfensters mit der linken Maustaste

Pull–Down–Menüs

In Abhängigkeit von der Anzahl und Art der gewählten Knoten stehen unterschiedliche Pull–Down–Menüs zur Verfügung.

- Kein Knoten ist gewählt
  - add root
    Es wird eine neue Wurzel hinzugenommen. Ein Wurzelknoten enthält einen einzeiligen Text.
  - remove tempLinks
    Die Links zwischen Knoten werden gelöscht
  - display options ▷
    * show tempLinks / hide tempLinks
      Die Darstellung der Links zwischen Knoten wird an- oder ausgeschaltet (ohne diese zu löschen)
    * change grid
  - layout ▷
    * update
      Das Fenster des *Map–Viewers* wird regeneriert
    * set auto layout / no auto layout
      Aktiviert bzw. desaktiviert das Auto-Layout, d.h. die automatische Anordnung von Knoten auf dem Bildschirm. Gewählte Knoten können innerhalb des Fensters nur bewegt werden, wenn das Auto-Layout ausgeschaltet ist. Bei eingeschaltetem Auto-Layout kann man sich in einem zusätzlichen Menüpunkt entscheiden, ob die Baumstrukturen horizontal oder vertikal angeordnet dargestellt werden sollen.
  - inspect view ▷
    Offnet verschiedene Inspektoren
- Ein Knoten ist gewählt

- add node ▷
  Dem aktivierten Knoten wird ein weiterer Nachfolgeknoten zugeordnet. Man kann sich entscheiden zwischen
  * 1-line text node
    Der Knoten enthält einen einzeiligen Text
  * 2-line text node
    Der Knoten erhält einen zweizeiligen Text. Diese Knotenart wird bei der Darstellung der Werkstückstruktur benutzt.
  * image node
    Knoten erhält ein gespeichertes Bild als Inhalt. Hierzu muß in der Klassenmethode *defaultDirectory* der Klasse *ImageNodeObject* das Verzeichnis auf einen Pfad gesetzt werden, der Smalltalk–*Images* enthält.
  * text & image node
    Der Knoten enthält sowohl einen einzeiligen Text als auch eine Graphik.
  * image fromUser
    Der Knoten erhält eine Graphik, die der Benutzer durch Öffnen eines Auswahlfensters spezifiziert.
- remove node

  Der gewählte Knoten wird gelöscht. Unterstrukturen werden zu neuen Bäumen. Diese Option steht auch zur Verfügung, wenn mehr als ein Knoten gewählt ist.
- remove subtree
  Der Knoten und seine Unterstrukturen werden gelöscht. Auch diese Option läßt sich auf mehrere Knoten gleichzeitig anwenden.
- edit node text
  Der Inhalt des Knotens kann editiert werden.
- set style ▷
  Die Darstellungsart der Knoten kann verändert werden.

• Zwei Knoten sind gewählt

- add tempLink
  Zwischen den beiden Knoten wird ein Link hergestellt.
- mapping ▷

  * ssp-Distance
    Berechnet den Abstand bei Anwenden des *Strictly Structure Preserving Mapping*
  * ssp-Map
    Führt *Strictly Structure Preserving Mapping* für die Bäume mit den gewählten Knoten als Wurzel durch und zeigt das Ergebnis graphisch an.
  * ssp-Maps
    Zeigt nacheinander alle möglichen Mappings mit der sich ergebenden Distanz an.
  * sp-Distance
    entsprechend oben für *Structure Preserving Mapping*
  * sp-Map

  * sp-Maps

* $\boxed{\text{tai-Distance}}$
  entsprechend oben für *Tai–Mapping*
* $\boxed{\text{tai-Map}}$
* $\boxed{\text{tai-Maps}}$

A.3 Die implementierten Entscheidungsnetze

In den Abbildungen ?? - ?? werden die im Planungssystem implementierten Netze dargestellt; zusammen stellen sie den in ?? vorgestellten Aufspannspezialisten dar, der dazu dient, einen geeigneten Fall aus der Fallbasis zu finden, welcher dann die Steuerung des hierarchischen Planungsablaufes übernehmen soll. Ein Knoten enthält in den Darstellungen eine Beschreibung der zu erfüllenden Goals sowie den Quellcode des Entscheidungsblocks, die Blätter werden durch eine Beschreibung des entsprechenden Exits zusammen mit seiner Nummer gekennzeichnet. Der linke Pfad stellt jeweils den **true**-Pfad dar, der rechte den **false**-Pfad. Wie in ?? erwähnt, stellt die Relation **slots** des Werkstücks den Übergabeparameter des Entscheidungsblocks dar; in den Entscheidungsblöcken wird der Parameter als **s** deklariert.

Beispiel:

```
[:s | ((s at: #clambingRadius) first) = #less10]
```

Die Einträge in der Relation **slots** stellen jeweils Instanzen des Typs **Dictionary** dar; deshalb kann auf einen bestimmten Eintrag mit Hilfe des Schlüsselwortes zugegriffen werden; z.B. wird in obigem Beispiel auf den Wert des Eintrags unter **clambingRadius** durch

```
(s at: #clambingRadius)
```

zugegriffen.

Die Blätter der Entscheidungsnetze werden durch eine Beschreibung des jeweiligen Exits dargestellt; diese Beschreibungen dienen dem besseren Verständnis der getroffenen Entscheidung. Im System stellen die Nummern der Exits den jeweiligen Beitrag zur Indizierung der Fallbasis dar.

Im einzelnen sind 4 Entscheidungsnetze implementiert:

- *„possible clambing situation - left side“* - Spannmittel zum Aufspannen des Werkstücks von links (Abb. ??)
- *„possible clambing situation - right side“* - Spannmittel zum Aufspannen des Werkstücks von rechts (Abb. ??)
- *„Facing necessary“* - Bearbeitung der Stirnseiten nötig (Abb. ??)
- *„Turning at Ends“* - Bearbeitung auf dem ersten und letzten Element (als potentielle Spannstellen) nötig (Abb. ??)

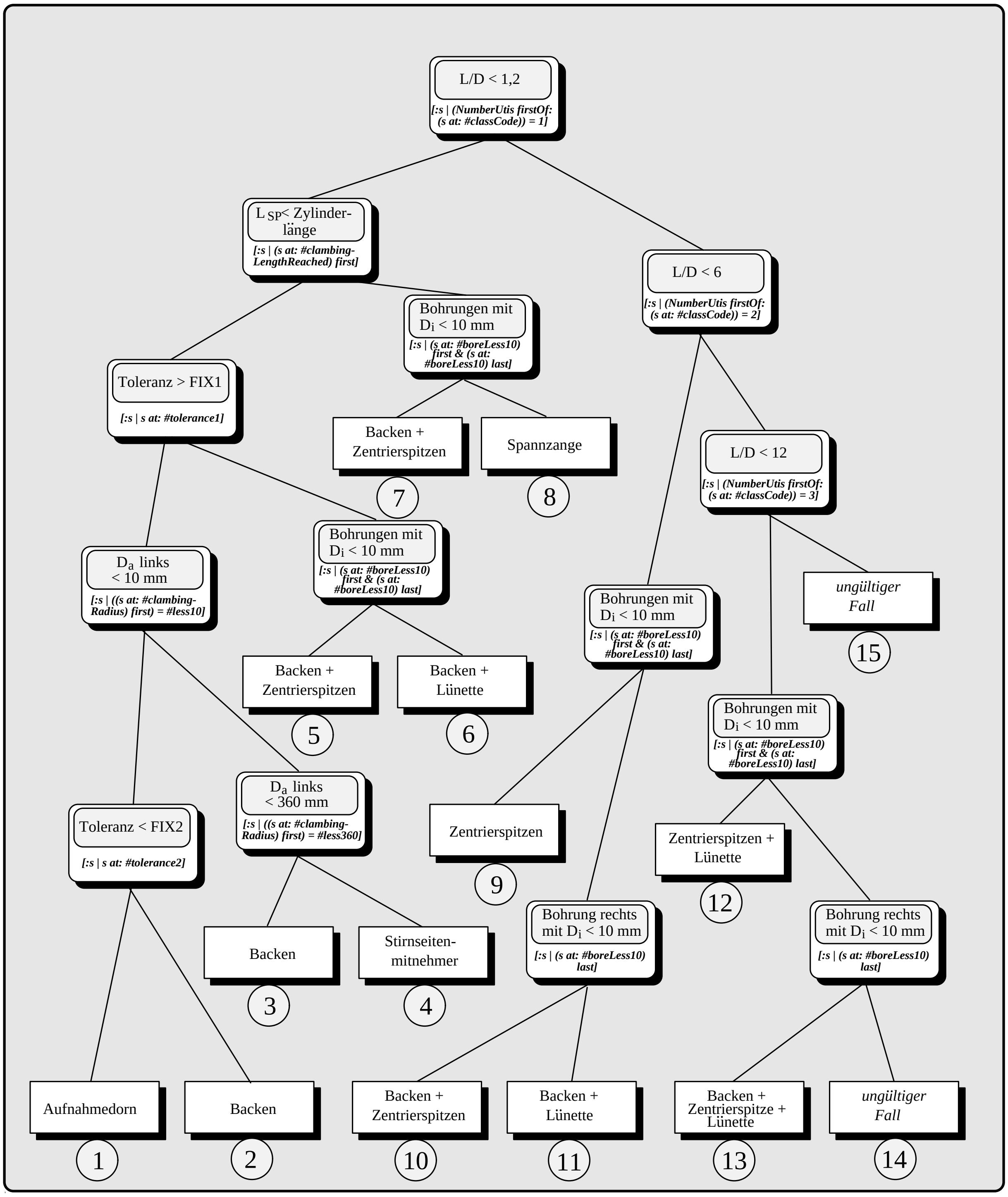


Abbildung A.27: Entscheidungsnetz *„possible clambing situation - left side“*

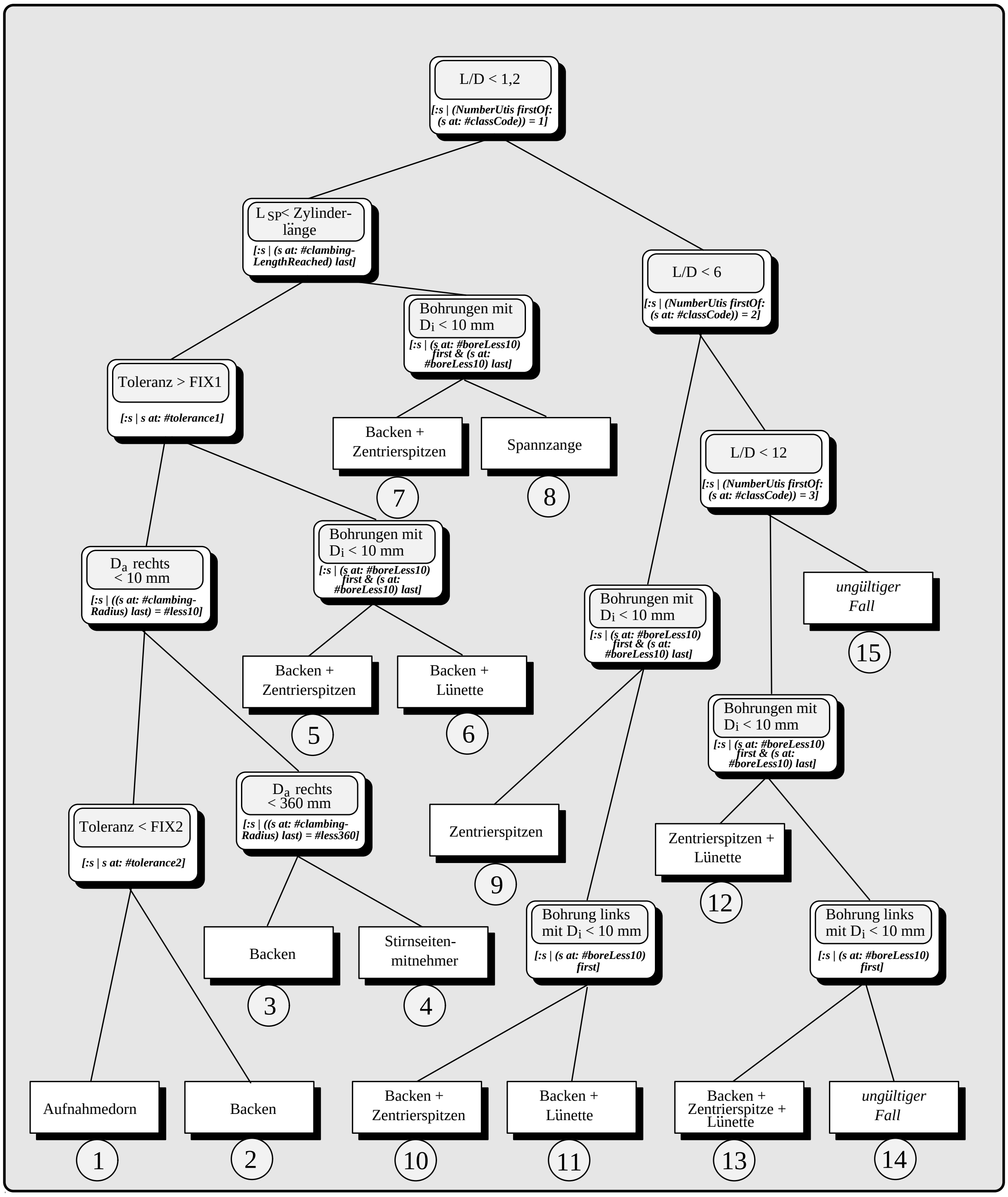


Abbildung A.28: Entscheidungsnetz *„possible clambing situation - right side"*

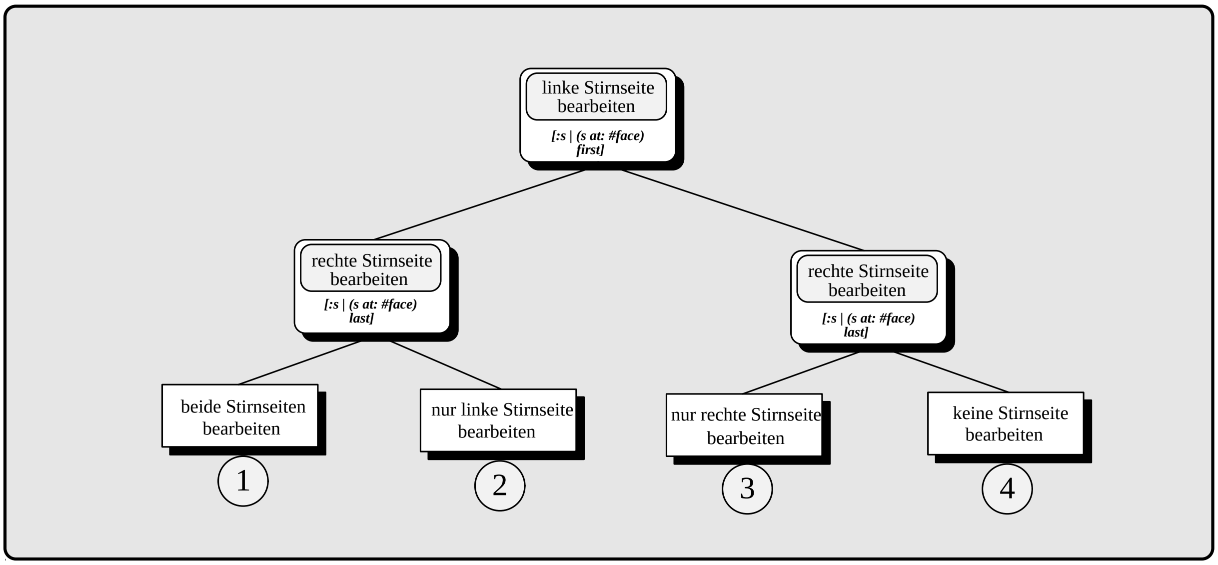


Abbildung A.29: Entscheidungsnetz *„Facing necessary"*

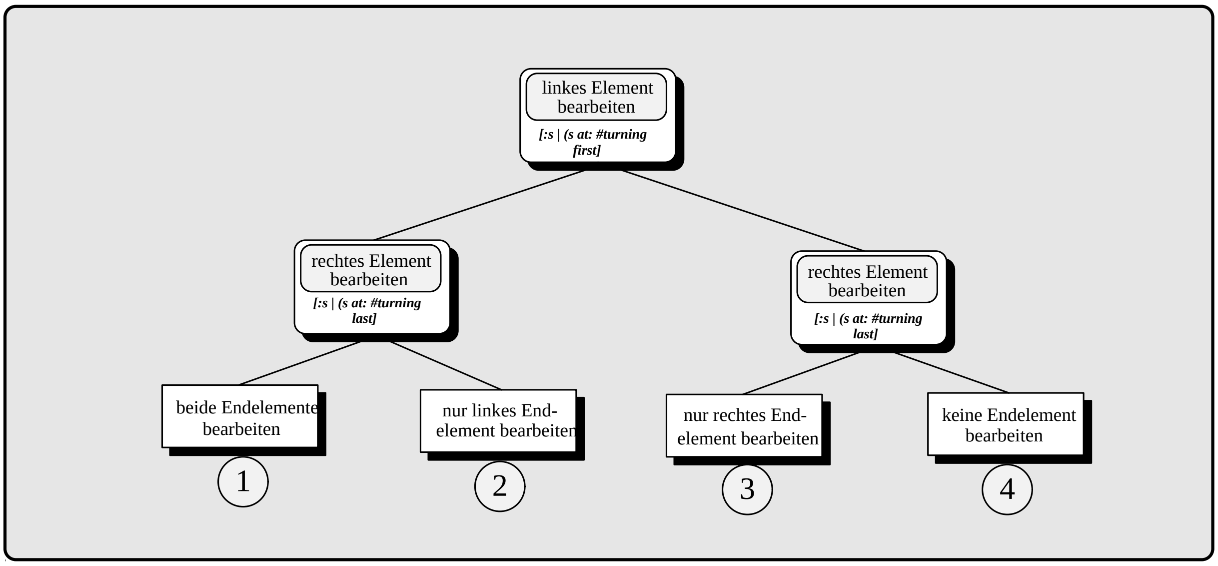


Abbildung A.30: Entscheidungsnetz *„Turning at Ends"*

Anhang B

Ausgabe des CHEF-Systems

Die folgenden Ausgaben des CHEF-Systems erfolgten bei der Planung eines Kochrezeptes für gebratenes Rindfleisch mit Broccoli.

```
Searching for plan that satisfies -
       Include beef in the dish.
       Include Broccoli in the dish.
       Make a stir-fried dish.

Found recipe -> REC2 BEEF-WITH-GREEN-BEANS

Recipe exactly satisfies goals ->
       Make a stir-fry dish.
       Include beef in the dish.

Recipe partially matches ->
       Include broccoli in the dish.
             in that the recipe satisfies:
                    Include vegetables in the dish.

Building new name for copy of BEEF-WITH-GREEN-BEANS
Calling recipe BEEF-AND-BROCCOLI

Modifying recipe: BEEF-AND-BROCCOLI
to satisfy: Include broccoli in the dish.

Placing some broccoli in recipe BEEF-AND-BROCCOLI

-Considering ingredient-critic:
       Before doing step: Stir fry the -Vegetable-
             do: Chop the broccoli into pieces the size of chunks.
-ingredient-critic applied.


                  BEEF-AND-BROCCOLI

               A half pound of beef
               Two tablespoons of soy sauce
               One teaspoon of rice wine
               A half tablespoon of corn starch
```

```
            One teaspoon of sugar
            A half pound of broccoli
            One teaspoon of salt
            One chunk of garlic

        Chop the garlic into pieces the size of matchheads.
        Shred the beef.
        Marinate the beef in the garlic, sugar, corn starch,
        rice wine and soy sauce.
        Chop the broccoli into pieces the size of chunks.
        Stir fry the spices, rice wine and beef for one minute.
        Add the broccoli to the spinices, rice wine and beef.
        Stir fry the spices, rice wine and beef for three minutes.
        Add the salt to the spices, rice wine, broccoli and beef.

    The beef is now tender.         The dish now tastes salty.
    The dish noew tastes savory.    The dish now tastes sweet.
    The broccoli is now crisp.      The dish now tastes like garlic.


Checking goals of recipe -> BEEF-AND-BROCCOLI

Recipe -> BEEF-AND-BROCCOLI has failed goals.

The goal: The broccoliu is now crisp.
is not satisfied.
It is instead the case that: The broccoli is now soggy.

Unfortunately: The broccoli is now a bad texture.
In that: The broccoli is now soggy.

Changing name of recipe BEEF-AND-BROCCOLI
to BAD-BEEF-AND-BROCCOLI



Explaining the following failures:
       It is not the case that: The broccoli is now crisp.
in that: The broccoli is now soggy.
       The broccoli is now a bad texture.
in that: The broccoli is now soggy.
In: BAD-BEEF-AND-BROCCOLI

ASKING THE QUESTION: 'What is the failure?'

ANSWER-> The failure is: It is not the case that: The broccoli is
         now crisp.

ASKING THE QUESTION: 'What is the preferred state?'

ANSWER-> The preferred state is: The broccoli is now crisp.

ASKING THE QUESTION: 'What was the plan to achieve the preferred
         state?'
```

```
ANSWER-> The plan was: Stir fry the sugar, soy sauce, rice wine,
                       garlic, corn starch, broccoli and beef for
                       three minutes.

ASKING THE QUESTION: 'What were the conditions that led to the
         failure?'

ANSWER-> There is thin liquid in the pan from the beef equaling 4.8
         teaspoons was caused by:
                    Stir fry the sugar, soy sauce, rice wine,
                    garlic, corn starch, broccoli and beef for
                    three minutes.

ASKING THE QUESTION: 'Do the conditions that caused the failure
          satisfy any goals?'

ANSWER-> The condition: There is thin liquid in the pan from the
         beef equaling 4.8 teaspoons
                       is a side effect only and meets no goals.

ASKING THE QUESTION: 'What goals does the step which caused the
         condition enabling the failure satisy?'

ANSWER-> The step: Stir fry the sugar, soy sauce, rice wine,
                   garlic, corn starch, broccoli and beef for
                   three minutes.
                       enables the satisfaction of the following
                       goals:
                       The dish now tastes savory.
                       The beef is now tender.

Applying TOP -> SIDE-EFFECT:DISABLED-CONDITION:CONCURRENT
to failure it is not the case that: The broccoli is now crisp.
in recipe BAD-BEEF-AND-BROCCOLI

Asking questions needed for evaluating strategy:
       SPLI-AND-REFORM

ASKING-> Can plan
             Stir fry the sugar, soy sauce, rice wine, garlic,
             corn starch, broccoli and beef for three minutes.
          be split and rejoined

Found plan: Instead of doing step: Stir fry the sugar, soy sauce,
             rice wine, garlic, corn starch, broccoli and beef
             for three minutes.
       do:
       S1 = Stir fry the broccoli for three minutes.
       S2 = Remove the broccoli from the result of action S1.
       S3 = Stir fry the sugar, soy sauce, rice wine, garlic, corn
            starch and beef for three minutes.
       S4 = Add the result of action S2 to the result of action S3.
       S5 = Stir fry the result of action S4 for a half minute.
```

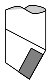
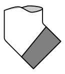
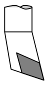
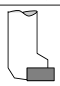
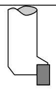
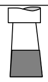
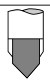
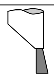
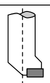
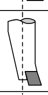
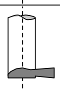
```
Asking questions needed for evaluating strategy: ALTER-PLAN:SIDE-EFFECT

ASKING-> Is there an alternative to
              Stir fry the sugar, soy sauce, rice wine, garlic,
              corn starch, broccoli and beef for three minutes.
       that will enable
              The dish now tastes savory.
       which does not cause
              There is thin liquid in the pan from the beef equaling
              4.8 teaspoons.

No alternate plan found.

Asking questions needed for evaluating strategy: ADJUNCT-PLAN

ASKING-> Is there an adjunct plan that will disable
              There is thin liquid in the pan from the beef equaling
              4.8 teaspoons.
       that can be run with
              Stir fry the sugar, soy sauce, rice wine, garlic,
              corn starch, broccoli and beef for three minutes.

No adjunct plan found.

Deciding between modification plans suggested by strategies:
       Only one modification can be implemented -> SPLIT-AND-REFORM

Implementing plan -> Instead of doing step: Stir fry the sugar,
              soy sauce, rice wine, garlic, corn starch,
              broccoli and beef for three minutes.
do:    S1 = Stir fry the broccoli for three minutes.
       S2 = Remove the broccoli from the result of action S1.
       S3 = Stir fry the sugar, soy sauce, rice wine, garlic,
            corn starch and beef for three minutes.
       S4 = Add the result of action S2 to the result of action S3.
       S5 = Stir fry the result of action S4 for a half minute.
Suggested by strategy SPLI-AND-REFORM.


Building demons to anticipate failure.

Building demon: DEMON0 to anticipate interaction between rules:
       "Meet sweats when it is stir-fried."
       "Stir-frying in too much liquid makes vegetables soggy."

Indexing demon: DEMON0 under item: MEAT
by test:
       Is the item a MEAT.

Indexing demon: DEMON0 under item: VEGETABLE
by test:
       Is the item a VEGETABLE.
and    Is the TEXTURE of item CRISP.
```

```
Goal to be activated = Avoid failure of type
SIDE-EFFECT:DISABLED-CONDITION:CONCURRENT
exemplified by the failure 'The broccoli is now soggy' in recipe
BEEF-AND-BROCCOLI.

Building demon: DEMON1 to anticipate interaction between rules:
       "Liquids make things wet."
       "Stir-frying in too much liquid makes vegetables soggy."

Indexing demon: DEMON1 under item: SPICE
by test:
       Is the texture of item LIQUID.

Indexing demon: DEMON1 under item: VEGETABLE
by test:
       Is the item a VEGETABLE.
and    Is the TEXTURE of item CRISP.

Goal to be activated = Avoid failure of type
SIDE-EFFECT:DISABLED-CONDITION:CONCURRENT
exemplified by the failure 'The broccoli is now soggy' in recipe
BEEF-AND-BROCCOLI.
```

Anhang C

Werkmitteleinteilungen

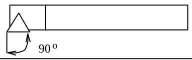
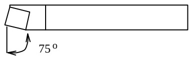
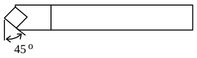
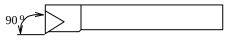
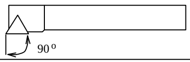
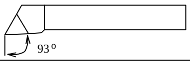
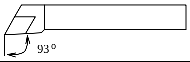
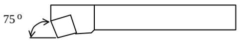
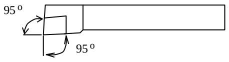
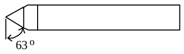
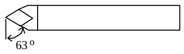
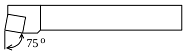
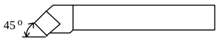
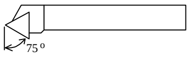
C.1 Drehmeißeleinteilung (DIN 4951 - 4981)

Form	Bezeichnung	Schneidenausführung	
		Schnellarbeitsstahl	Hartmetall
	gerader Drehmeißel	4951	4971
	gebogener Drehmeißel	4952	4972
	Eckdrehmeißel	4965	4978
	abgesetzter Eckdrehmeißel		4977
	abges. Seitendrehmeißel	4960	4980
	breiter Drehmeißel	4956	4976
	spitzer Drehmeißel	4955	4975
	Stechdrehmeißel	4961	4981
	Innen-Drehmeißel	4953	4973
	Innen-Eckdrehmeißel	4954	4974
	Innen-Stechmeißel	4963	

C.2 Einteilung der Wendeschneidplatten (DIN 4987)

Kennbuchstabe	Grundform	Eckenwinkel
H	sechseckig	120°
O	achteckig	135°
P	fünfeckig	108°
S	quadratisch	90°
T	dreieckig	60°
C	rhombisch	80°
D		55°
E		75°
M		86°
V		35°
W	dreieckig mit vergrössertem Eckenwinkel	80°
L	rechteckig	90°
A	rhomboidisch	85°
B		82°
K		55°
R	rund	

C.3 Lage der Schneidplatte am Halter (DIN 4984)

Form	Bild	Schnittrichtung
A		längs
B		längs
D		längs (beidseitig)
F		quer
G		längs
J		längs
		längs
K		quer
L		längs und quer
N		längs
		längs
R		längs
S		längs und quer
T		längs

Anhang D

Allgemeine Tabellen

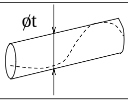
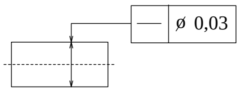
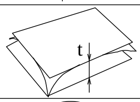
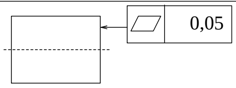
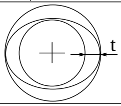
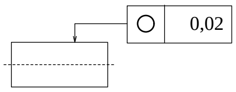
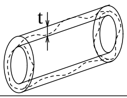
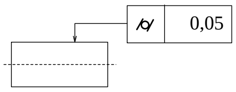
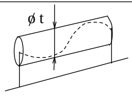
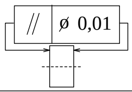
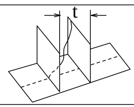
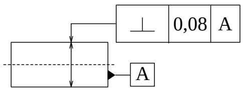
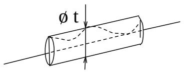
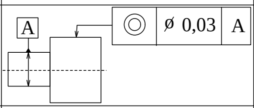
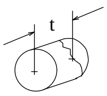
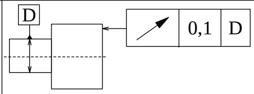
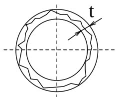
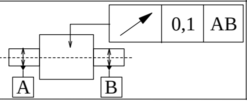
D.1 ISO-Grundtoleranzen (IT) für Längenmaße in μm (DIN 7151)

Nennmaß-bereich (mm)	IT																			
	01	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
von **1** bis **3**	0,3	0,5	0,8	1,2	2	3	4	6	10	14	25	40	60	100	140	250	400	600	-	-
über **3** bis **6**	0,4	0,6	1	1,5	2,5	4	5	8	12	18	30	48	75	120	180	300	480	750	-	-
über **6** bis **10**	0,4	0,6	1	1,5	2,5	4	6	9	15	22	36	58	90	150	220	360	580	900	1500	-
über **10** bis **18**	0,5	0,8	1,2	2	3	5	8	11	18	27	43	70	110	180	270	430	700	1100	1800	2700
über **18** bis **30**	0,6	1	1,5	2,5	4	6	9	13	21	33	52	84	130	210	330	520	840	1300	2100	3300
über **30** bis **50**	0,6	1	1,5	2,5	4	7	11	16	25	39	62	100	160	250	390	620	1000	1600	2500	3900
über **50** bis **80**	0,8	1,2	2	3	5	8	13	19	30	46	74	120	190	300	460	740	1200	1900	3000	4600
über **80** bis **120**	1	1,5	2,5	4	6	10	15	22	35	54	87	140	220	350	540	870	1400	2200	3500	5400
über **120** bis **180**	1,2	2	3,5	5	8	12	18	25	40	63	100	160	250	400	630	1000	1600	2500	4000	6300
über **180** bis **250**	2	3	4,5	7	10	14	20	29	46	72	115	185	290	460	720	1150	1850	2900	4600	7200
über **250** bis **315**	2,5	4	6	8	12	16	23	32	52	81	130	210	320	520	810	1300	2100	3200	5200	8100
über **315** bis **400**	3	5	7	9	13	18	25	36	57	89	140	230	360	570	890	1400	2300	3600	5700	8900
über **400** bis **500**	4	6	8	10	15	20	27	40	63	97	155	250	400	630	970	1550	2500	4000	6300	9700

Erläuterungen:

1. Die ISO-Grundtoleranzen gelten für alle Maße wie Durchmesser, Längen, Breiten usw.
2. Mit Schlichtdrehen lassen sich Maßgenauigkeiten von IT8 bis IT7 erreichen
3. Beim Feinschlichten sind bei optimalen Drehbedingungen Genauigkeiten von IT6 zu erreichen

D.2 Form- und Lagetoleranzen (Ausschnitt aus DIN 7184)

Symbol und tolerierte Eigenschaft			Anwendungsbeispiele: Toleranzzone	Zeichnungsangabe	Erklärung
Form	—	Geradheit			Die Achse des zylindrischen Teiles des Bolzens muß innerhalb eines Zylinders vom Durchmesser t = 0,03 mm liegen.
Form	▱	Ebenheit			Die tolerierte Flache muß zwischen zwei parallelen Ebenen vom Abstand t = 0,05 mm liegen.
Form	○	Rundheit			Die Umfangslinie jedes Querschnitts muß in einem Kreisring von der Breite t = 0,02 mm enthalten sein.
Form	⌭	Zylinderform			Die tolerierte Fläche muß zwischen zwei koaxialen Zylindern liegen, die einen radialen Abstand von t = 0,05 mm haben.
Lage	//	Parallelität			Die tolerierte Achse muß innerhalb eines zur Bezugsachse parallel liegenden Zylinders vom Durchmesser t = 0,1 mm liegen.
Lage	⊥	Rechtwinkligkeit			Die tolerierte Achse muß zwischen zwei parallelen zur Bezugsfläche und zur Pfeilrichtung senkrechten Ebenen vom Abstand t = 0,08 mm liegen.
Lage	◎	Konzentrizität			Die Achse des tolerierten Teiles der Welle muß innerhalb eines Zylinders vom Durchmesser t = 0,03 mm liegen, dessen Achse mit der Achse des Bezugselementes fluchtet.
Lage	↗	Planlauf			Bei Drehung um die Bezugsachse D darf die Planlaufabweichung in jedem Meßzylinder 0,1 mm nicht überschreiten.
Lage	↗	Rundlauf			Bei Drehung um die Bezugsachse AB darf die Rundlaufabweichung in jeder senkrechten Meßebene 0,1 mm nicht überschreiten.

D.3 Oberflächensymbole in Zeichnungen

Oberflächen-zeichen	zulässige größte Rautiefe R in µm				Bedeutung
	Reihe 1	Reihe 2	Reihe 3	Reihe 4	
(Oberfläche ohne Zeichen)	beliebig				Oberfläche, an die keine bestimmte Anforderungen gestellt werden
▽	160	100	63	25	Oberflächen mit einer Rauhheit, die die größte zulässige Rauhtiefe nicht überschreiten darf
▽▽	40	25	16	10	
▽▽▽	16	6,3	4	2.5	

Abbildung D.1: Oberflächenzeichen und Zuordnung der Rautiefen (DIN 3141)

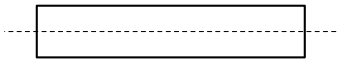
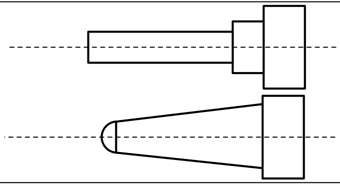
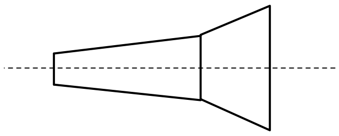
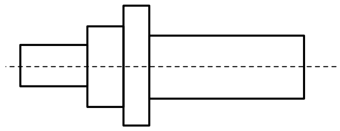
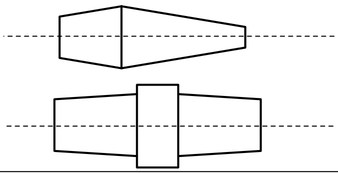
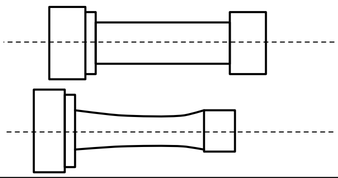
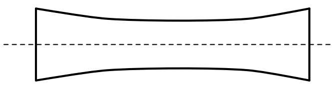
D.4 Vorschub in Abhängigkeit von der Oberflächengüte

Spitzen-radius	Vorschub f (mm/U) = $f(R_{th}, r)$					
	Feindrehen		Schlichten		Schruppen	
	▽▽▽		▽▽		▽	
r (mm)	R_{th} 4 µm	R_{th} 6,3 µm	R_{th} 16 µm	R_{th} 25 µm	R_{th} 63 µm	R_{th} 100 µm
0,5	0,13	0,16	0,26	0,32	0,50	0,63
1,0	0,18	0,22	0,36	0,45	0,71	0,89
1,5	0,22	0,27	0,44	0,55	0,87	1,10
2,0	0,25	0,31	0,50	0,63	1,00	1,26
3,0	0,31	0,38	0,62	0,77	1,22	1,55

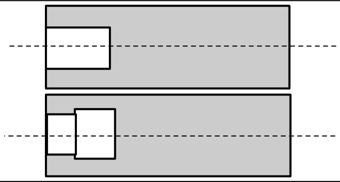
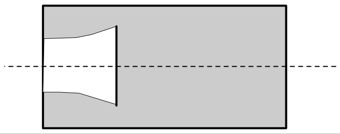
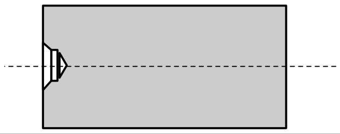
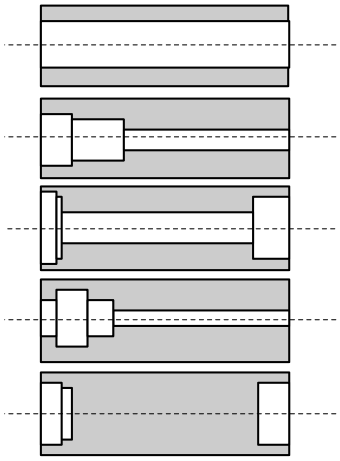
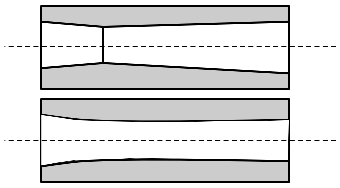
Abbildung D.2: Vorschub f in mm in Abhängigkeit von der geforderten Formrauhigkeit R_{th} und dem Spitzenradius r

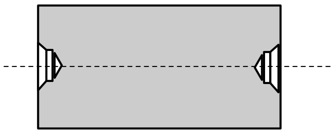
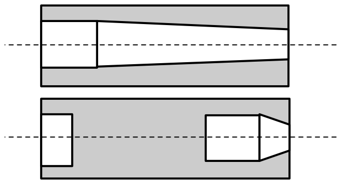
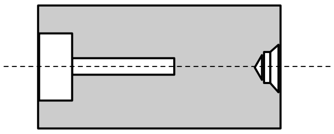
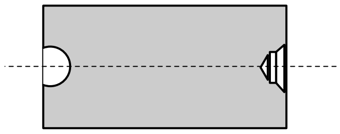
D.5 Kodierungstabellen der Grundformen

D.5.1 Kodierung der äußeren Grundform

Code	Bezeichnung	Beispiel
1	Das Teil besitzt eine achsensymmetrische Außenkontur	
2	Das Teil ist einseitig steigend und besteht ausschließlich aus Stufen oder besitzt mindestens eine Stufe an einem Ende	
3	Das Teil ist einseitig steigend und besteht ausschließlich aus Freiformflächen (nicht zylindrischen Flächen) oder besitzt an keinem der Enden eine Stufe (zylindrische Spannfläche)	
4	Das Teil ist zur Mitte steigend und besteht ausschließlich aus Stufen oder besitzt mindestens eine Stufe an einem Ende	
5	Das Teil ist zur Mitte steigend und besteht ausschließlich aus Freiformflächen oder besitzt an keinem der beiden Enden eine Stufe	
6	Das Teil ist zur Mitte fallend und besteht ausschließlich aus Stufen oder besitzt mindestens eine Stufe an einem Ende	
7	Das Teil ist zur Mitte fallend und besteht ausschließlich aus Freiformflächen oder besitzt an keinem der beiden Enden eine Stufe	

D.5.2 Kodierung der inneren Grundform

Code	Bezeichnung	Beispiel
0	Das Teil besitzt keine Innenkontur	
1	Das Teil besitzt an genau einer Stirnseite eine Innenkontur, welche eine Stufe hat	
2	Das Werkstück besitzt an genau einer Stirnseite eine Innenkontur, die eine Freiformfläche hat	
3	Das Teil besitzt an genau einer Stirnseiten eine Zentrierbohrung	
4	Das Teil hat an jeder seiner Stirnseiten eine Eintrittsfläche für die Innenkontur. Dabei kann die Innenkontur durchgängig oder getrennt sein. Beide stirnseitigen Elemente sind Stufen	
5	Das Teil hat an jeder seiner Strirnseiten eine Eintrittsfläche für die Innenkontur. Dabei kann die Innenkontur durchgängig oder getrennt sein. Beide stirnseitigen Elemente sind Freiformflächen	

Code	Bezeichnung	Beispiel
6	Das Teil besitzt an jeder seiner zwei Stirnseiten eine Zentrier-bohrung	
7	Das Teil hat an jeder seiner Stirnseiten eine Eintritts-fläche fur die Innenkontur. Dabei kann die Innenkontur durchgängig oder getrennt sein. Ein stirnseitiges Element ist eine Stufe, das andere eine Freiformfläche	
8	Das Teil hat an jeder seiner Stirnseiten eine Eintrittsfläche für die Innenkontur. Dabei kann die Innenkontur durch-gängig oder getrennt sein. Ein stirnseitiges Element ist eine Stufe, das andere ist eine Zentrierbohrung	
9	Das Teil hat an jeder seiner Stirnseiten eine Eintrittsfläche für die Innenkontur. Dabei kann die Innenkontur durch-gängig oder getrennt sein. Ein stirnseitiges Element ist eine Freiformfläche, das andere ist eine Zentrierbohrung	

Anhang E

Englische Übersetzung von Spezialausdrücken

abdrehen	face, turn (off,down)
Ablauf	outlet
Abrundung	rounding off
Arbeitsablauf	sequence of operations
Arbeitsgang	operation, process
Arbeitsseite	operating side
Aufnahmedorn	expanding mandrel
Aufspannplatte	turret block
Aufspannung	setting, (work) mounting, chucking
Aufspannvorrichtung	clambing device, chuck; work holding attachment, work-fixture
Aufspannzeit	setting-up time
Aufsteckhalter	arbor for shell tools
ausdrehen, innen	turn inside diameter
Ausgangswerkstoff	basic material
ausspannen	(Werkstücke:) unclamp, release; (Werkzeuge:) unload
Bearbeitung in zweiter Aufspannung	second-setting machining
Bohrdurchmesser	diameter drilled
Bohrlünette	boring steady
Bohrung, abgesetzte	stepped hole, shouldered hole
Drehbacken	turning jaws
Drehbank	lathe
Drehbohrmeißel	rotary boring tool
Drehdurchmesser	swing, turning diameter

drehen	turn; rotate, revolve; (plane Flächen:) face; (langdrehen:) turn; (formdrehen:) form; (plandrehen:) face
Drehleistung	turning capacity
Drehteil	(als Werkstück:) part to be turned, lathe work; (als Drehbankerzeugnis:) turned part
Drehvorschub	rotary feed
Dreibackenfutter	three-jaw chuck
Ebenheit	evenness
Einspannung	(Werkstücke:) loading; (Werkzeuge:) chucking
einstechen	groove, slot, neck; recess, cut in
Einstellwinkel	plan angle; side angle; setting angle
fasen	chamfer, bevel
Feinschlichten	fine finishing
Feinstdrehen	super-finish turning
Fertigdrehen	finish turning
Festklemmen	clamping, binding
Formtoleranz	tolerance of form
Freifläche	(e. Schneidstahls:) rake, slope; (der Hauptschneide:) front rake, back rake; (der Nebenschneide:) side rake
Freischnitt	clearance cut
Freiwinkel	clearance angle
Futteraufspannung	chucking
Futterscheibe	(für Klemmfutter:) adapter; (e. Backenfutters:) backplate
Geraddrehen	straight turning
Geradheit	straightness
Gesamtspanquerschnitt	total cutting capacity
Gewinde	thread
Gewindeauslauf	run-out, back taper, end thread
Gewindehinterstechen	grooving at back of thread
Gewindelänge	length of thread; threaded length
Gewindeschneiden	(allg.:) thread cutting; (Außengewinde:) threading; (Innengewinde:) tapping
Gewindestärke	size of thread, diameter of thread
Grobschlichten	rough-finishing
Handspannfutter	hand operated chuck
Hartmetall	cemented metal carbide, cutting metal, cutting alloy

Hartmetallwerkzeug	carbide tipped tool
Hauptschneide	main cutting edge, cutting edge on the feed side
Hauptzeit	productive time, machining time
innen ausdrehen	turn inside diameter
Inneneinstechen	internal recessing
Innen-Einstechwerkzeug	internal recessing tool
Innengewindeschneiden	tapping
Innenplandreheinrichtung	internal facing attachment
Innenspannbacke	internal chuck jaw
Innenspannung	(e. Spannvorrichtung:) internal gripping
Kegelbohrung	taper bore
Kegeldrehen	taper turning; angular turning
Keilwinkel	(e. Schneidstahls:) lip angle
Konzentrizität	concentricity
Kraftspannfutter	heavy duty chuck
Lagetoleranz	tolerance of position
Langdrehen	longitudinal turning, straight turning, plain turning; (meist kurz:) turning
Lünette	steady
mitlaufende Drehbankspitze	live center
Nutenfräsarbeit	keyway cutting, keywaying
Parallelität	parallelism
Planarbeit	facing
Plandrehen	facing, transverse turning
Plandrehwerkzeug	facing tool
Planfläche	face, end face
Planlauftoleranz	run-out
Planscheibe	faceplate
Planschlichten	finish facing, transverse finishing
Planschnitt	facing cut
Planschruppen	rough facing
Rauhtiefe	peak-to-valley height
Rechtwinkligkeit	sqareness, rectangularity
Reitstock	tailstock
Revolverdrehbank	turret lathe
Rohling	blank
Rundheit	concentricity, roundness
Rundlauf	true running

Schlichtdrehen	smoothening; finish turning
Schneidenwinkel	lip angle
Schneidmetall	cutting alloy, cutting metal
Schneidwinkel	cutting angle; cutting tool angle
Schnellfutter	quick-acting chuck, quick catch chuck
Schnellspannung	quick gripping
Schnitt	cut; cutting
Schnittrichtung	direction of cut
Schruppen	roughing
Spanabfluß	chip flow
Spanfläche	(e. Schneidstahls:) true rake
Spannbereich	chucking capacity
Spanndorn	expanding mandrel
Spanndurchmesser	gripping diameter
Spanneinrichtung	gripping mechanism, holding fixture, chucking fixture
Spannen	(e. Spannzange:) locking; (in ein Futter:) chucking, gripping, clambing
Spannfutter, selbstzentrierendes	scroll chuck, self-centering chuck
Spannkraft	(e. Spanneinrichtung:) chucking power, chucking effect
Spannstelle	gripping position
Spannzange	collet chuck
Spanwinkel	(e. Schneidstahls:) true rake angle, front rake angle
Spitzen	lathe centres
Spitzenwinkel	(e. Schneidstahls:) nose angle
Stangen-Spannfutter	rod chuck
Stangenzuführung	rod feeding attachment
Steifigkeit	rigidity, stiffness
Steigung	ascent, slope, gradient; (gearing) lead; pitch
Stirnflächen bearbeiten	face
Stirnmitnehmer	face driver
Stirnseite	face, end
Toleranzbereich	permissible variation
Umspannzeit	loading time
Universaldreibackenfutter	universal three-jaw geared scroll lathe chuck
Universallünette	universal back rest

Universal-Spannfutter	scroll chuck
unlegierter Werkzeugstahl	plain carbon tool steel
Verspanungsvorgang	stock removing process
Vorschubschlitten	feed slide
Wärmeabfluß	heat transfer
Werkstoffdurchlaß	(e. Drehbank:) diameter of work admitted; größter _ für Stangenarbeiten, maximum diameter of bar work admitted; größter _ für Futterarbeiten, maximum diameter of chuck work admitted
Werkstück in zweiter Aufspannung	second-operation work
Werkstückaufspannung	work-mounting
Werkzeuganordnung	tooling lay-out, tooling diagram
Werkzeugfutter	tool chuck
Winkeleinstellung	angular setting, angular adjustment, angularity
Winkelfläche	angular surface
Zangenspannfutter	spring collet chuck
Zentrierklemmfutter	self-centering chuck
Zentrierspindel	centering spindle
zerspanen	cut, remove metal
Zweibackenfutter	two-jaw chuck
Zylinder	cylinder
Zylinderform	cylinder shape

Abbildungsverzeichnis

Literaturverzeichnis

[1] Althoff, K.-D. et al *MOLTKE – an Integrated Workbench for Fault Diagnosis in Engineering Systems.* In: *Proc. 4th International Conference Artificial Intelligence & Expert System Applications* (EXPERSYS-92), 1992

[2] Carbonell, J. G. *Learning by Analogy: Formulation and Generating Plans from Past Experience.* In: *Machine Learning: An Artificial Intelligence Approach,* 1, Tioga, Palo Alto, 1983

[3] Chapman, D. *Planning for Conjunctive Goals.* In: *Artificial Intelligence 32(3),* 1987

[4] DIN Deutsches Institut für Normierung e.V. (Hrsg.) *Anwendung der Normen über Form- und Lagetoleranzen in der Praxis.* Normenheft/DIN 7. 1987

[5] Ernst, O. *Spannmittelauswahl für die Außendrehbearbeitung als Baustein der CAD/CAM-Technologieplanung.* Studienarbeit FBK, Universität Kaiserslautern, 1988.

[6] Fikes, R., Nilson, N. *STRIPS: A New Approach to Theorem Proving in Problem Solving.* In: *Artificila Intelligence,* 2 (189), 1971

[7] Friedland, P. E., Iwanski, I. *The Concept and Implementation of Skeletal Plans.* In: *Automated Reasoning,* 1 (161), 1985

[8] Hammond, Kristian J. *Case–Based Planning: Viewing Planning as a Memory task.* Academic Press, Boston, MA, 1989

[9] Hertzberg, Joachim. *Planen. Einführung in die Planerstellungsmethoden der Künstlichen Intelligenz.* Reihe Informatik, Band 65, BI-Wiss.-Verl., 1989

[10] Hoch, Holger. *Eine Komponente zur Modifikation von Fertigungsplänen in der fallbasierten Arbeitsplanung.* Projektarbeit, Universität Kaiserslautern, AG Prof. Dr. M. M. Richter, 1992

[11] Kambhampati, S., Hendler, J. *A Validation-Structure-Based Theory of Plan Modification.* In: *Artificial Intelligence 55,* 1992

[12] Kammer, P., Becker, C. CAbPlan*: Ein System zur fallbasierten Arbeitsplanung bei rotationssymmetrischen Drehteilen.* Projektarbeit, Universität Kaiserslautern, AG Prof. Dr. M. M. Richter, 1992

[13] Kiefer, A. *GraMoD: Ein objektorientiertes, featurebasiertes Entwurfssystem für rotationssymmetrische Drehteile.* Diplomarbeit, Universität Kaiserslautern, AG Richter, 1992

[14] Kolodner, J.L. *An Introduction to Case-Based Reasoning.* In *Artificial Intelligence Review 6,* S. 3-34, 1992

[15] Lueg, Herbert. *Systematische Fertigungsplanung*

[16] Mertens, P. *Expertensysteme in der Produktion, Praxisbeispiele aus Diagnose und Planung.* München, Wien: Oldenburg, 1990

[17] Milberg, J., Peiker, S. *Geometrie- und technologieorientierte Verbindung von CAD-Systemen mit NC-Programmiersystemen.* In: *wt Werkstattstechnik 77*, S. 583-586, 1987

[18] Paucksch, Eberhard. *Zerspantechnik.* Viewegs Fachbücher der Technik

[19] Paulokat, J. *Unterstützung der Kontrolle bei der Konfigurierung durch ein problemklassenspezifisches TMS.* In: *Proc.2. Deutsche Tagung Expertensysteme, XPS-93*, Hamburg, Springer Verlag, 1993

[20] Petrie, C. *Planning and Replanning with Reason Maintenance.* Dissertation, University of Texas, Austin, 1991

[21] Radtke, M., Filser, F. *Ein Werkstückklassifizierungssystem zur Unterstützung der Aufspannplanung.* CIM-Centrum Kaiserslautern, 1992

[22] Richter, Michael M., O. Wendel. *Lernende Systeme*, 1991

[23] Ritzer, H. *Konzeption und Implementierung einer TMS-basierten Komponente zur Verwaltung von Abhängigkeiten bei der Konfiguration und Planung.* Diplomarbeit, Universität Kaiserslautern, 1992

[24] Simpson, R. L. *A Computer Model of Case-Based Reasoning in Problem Solving: An Investigation in the Domain of Dispute Mediation.* PhD thesis GIT-ICS-85/18, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, Georgia, 1985

[25] Spur,G. *Optimierung des Fertigungssystems Werkzeugmaschine.* aus der Reihe: *Fortschritte der Fertigung auf Werkzeugmaschinen*, Band3

[26] Steinberger, J. Einfluß des Spannzeugs auf die Wirtschaftlichkeit beim NC-Drehen. In: *Werkstatt und Betrieb 116*, S. 97-101, 1983

[27] Tanaka, E., Tanaka, K. *The Tree-To-Tree Editing Problem.* In: *International Journal of Pattern Recocnition and Artificial Intelligence*, Vol 2 No. 2, 1988, S. 221-240

[28] Tai, K.-C. *The Tree-to-Tree Correction Problem.* In: *Journal of the Association for Computing Machinery*, Vol. 26, No. 3, 1979, S. 422-433

[29] Tate, A. *Generating Project Networks.* In: *Proc. IJCAI*, 1977

[30] Thoben, Jörg. *Untersuchungen zur Fertigungsplanung.* Praktikumsbericht, 1990

[31] Töllner, K. *Spanen und Spannen, nicht nur sprachliche Verwandte.* In: *wt Werkstattstechnik 77*, Nr.1, S. 25-29, 1987

[32] Tschätsch, H. *Handbuch spanende Formgebung.* Hoppenstedt Technik Tabellen Verlag, 1988

[33] G. Warnecke (Band-hrsg.) *Expertensysteme in CIM.* Springer Verlag, Verlag TÜV Rheinland, 1991

[34] Wäsch, J. *Konzeption und Implementierung eines generativen Planungssystems zur Unterstützung der Arbeitsplanung bei der Computerintegrierten Fertigung (CIM).* Diplomarbeit, Universität Kaiserslautern, 1993

[35] Weck, M. *Werkzeugmaschinen Fertigungssysteme Band 1.* Studium und Praxis, VDI Verlag

[36] Wess, S. *PATDEX/2: Ein System zum fallfokusierenden Lernen in technischen Diagnosesituationen.* SEKI Working Paper SWP91/01, University of Kaiserslautern, Dept. of Computer Science, Januar 1991

[37] Wess, S. et al *Teilprojekt X9: Lernen und Analogie in technischen Expertensystemen*, Projektbericht X9, DFG, SFB314, 1993

[38] Wilkins, D. E. *Practical Planning – Extending the classical AI Planning Paradigm.* Morgan Kaufmann, 1988

[39] Wolf, M. *Matchen von geordneten markierten Bäumstrukturen.* Projektarbeit, Universität Kaiserslautern, 1993

Index