KfK-PDV 135
Oktober 1978

PDV-Berichte

Eine Untersuchung der Anforderungen an eine Prozeßprogrammiersprache bei der Automatisierung von Stückprozessen

M. E. Helfert
Gesellschaft für Automatisierung
mit Datenverarbeitungsanlagen mbH
Sindelfingen

Kernforschungszentrum Karlsruhe

PDV-Berichte

Die Kernforschungszentrum Karlsruhe GmbH koordiniert und betreut im Auftrag des Bundesministers für Forschung und Technologie das im Rahmen der Datenverarbeitungsprogramme der Bundesregierung geförderte Projekt Prozeßlenkung mit Datenverarbeitungsanlagen (PDV). Hierbei arbeitet sie eng mit Unternehmen der gewerblichen Wirtschaft und Einrichtungen der öffentlichen Hand zusammen. Als Projektträger gibt sie die Schriftenreihe PDV-Berichte heraus. Darin werden Entwicklungsunterlagen zur Verfügung gestellt, die einer raschen und breiteren Anwendung der Datenverarbeitung in der Prozeßlenkung dienen sollen.

Der vorliegende Bericht dokumentiert Kenntnisse und Ergebnisse, die im Projekt PDV gewonnen wurden.

Druck und Verbreitung:

Kernforschungszentrum Karlsruhe GmbH
Postfach 3640 7500 Karlsruhe 1

Bundesrepublik Deutschland

Eine Untersuchung der Anforderungen an eine Prozeßprogrammiersprache bei der Automatisierung von Stückprozessen

Von der Universität Stuttgart
zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Abhandlung

vorgelegt von
Manfred Ernst Helfert
geb. in Saaz

Hauptberichter: Prof. Dr.-Ing. R. Lauber
Mitberichter: Prof. Dr.-Ing. G. Stute

Tag der Einreichung: 16.6.1977
Tag der mündlichen Prüfung: 12.7.1978

KfK-PDV 135

PROJEKT PROZESSLENKUNG MIT DV-ANLAGEN

FORSCHUNGSBERICHT KfK-PDV 135

EINE UNTERSUCHUNG DER ANFORDERUNGEN AN EINE PROZESSPROGRAMMIERSPRACHE BEI DER AUTOMATISIERUNG VON STÜCKPROZESSEN

VON

MANFRED E. HELFERT

GESELLSCHAFT FÜR AUTOMATISIERUNG
MIT DATENVERARBEITUNGSANLAGEN MBH,
SINDELFINGEN

152 SEITEN
39 BILDER
1 TABELLE
74 LITERATUR-
STELLEN

OKTOBER 1978

KURZFASSUNG

Die operativen Aufgabenstellungen bei der prozeß-gekoppelten Automatisierung von Stückprozessen werden zusammengestellt. Aufgrund einer Untersuchung von Elementen und Struktur von Stückprozessen werden abstrakte Prozeßmodelle entwickelt. Der gegenständliche Modellaufbau eines Stückprozesses im Labor und dessen Betrieb an einem Prozeßrechner AEG 60-50 mittels PEARL-Programmen dient u.a. der Veranschaulichung und praktischen Demonstration der Verfahren. Die Ortsveränderung von Stücken ist Anlaß, die Anforderungen an eine Prozeßprogrammiersprache bei der Automatisierung von Stückprozessen zu diskutieren und zusammenzustellen.

Vorliegende Arbeit ist während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Regelungstechnik und Prozeßautomatisierung entstanden.

Herrn Prof. Dr.-Ing. R. Lauber danke ich hiermit sehr herzlich für die Diskussionen und Hinweise bei ihrer Erstellung.

Bei Herrn Professor Dr.-Ing. G. Stute bedanke ich mich ebenfalls recht herzlich für die Übernahme des Mitberichts

Inhaltsverzeichnis

1.	Einleitung und Aufgabenstellung	3
1.1	Das Problem der Beurteilung von Programmiersprachen bei Gebrauch und Entwurf	3
1.2	Abgrenzung der Aufgabenstellung und Gang der Arbeit	5
2.	Eine Untersuchung der Elemente und der Struktur des Stückprozesses	8
2.1	Die Objekte des Stückprozesses : Die Stücke	8
2.2	Klassifizierung von Stückprozessen	15
2.3	Die geschlossene Materialflußanordnung als Übertragungsglied	17
2.4	Elemente des Stückprozesses	27
2.5	Elementare Prozeßmodelle als Grundlage zur Prozeßführung	31
2.6	Zusammenfassung und Diskussion der Ergebnisse	36
3.	Grundsätzliche Anforderungen an einen Rechner im Stückprozeß aufgrund möglicher Verfahren zur Stückverfolgung und Lenkung	38
3.1	Möglichkeiten der Lokalisierung von Stücken	38
3.2	Ein Modell zur Lösung der abschnittsweisen, individuellen Stückgutverfolgung	45
3.3	Grundlegende Methoden zur Zielsteuerung von Stücken	50
3.4	Zusammenfassung	53
4.	Ein gegenständliches Modell eines Stückprozesses	57
4.1	Zielsetzung	57
4.2	Der Aufbau eines Warenverteilsystems	60
4.3	Das gegenständliche Modell eines Hochregallagers	71
5.	Grundlegende Anforderungen an eine Prozeßprogrammier-sprache bei der Automatisierung eines Stückprozesses	78
5.1	Ereignisse	78
5.2	Verarbeitung von M Beobachtungen	79
5.3	Zusammenfassung	85

6.	Mechanismen und Spracheigenschaften zur Behandlung von Aufträgen	86
6.1	Mechanismen in PEARL und Problemstellung	87
6.2	Ermittlung des Auftraggebers	95
6.3	Parallelität von Auftraggeber und Auftragnehmer	100
6.4	Mechanismen zur Bindung von Auftraggeber und Auftragnehmer	104
6.5	Diskussion von Lösungsvorschlägen	112
6.6	Zusammenfassung und Kritik	117
7.	Laufzeitbetrachtungen und Konkurrenzsituation im Stückprozeß	119
7.1	Einzelbeobachtung und die elementaren Laufzeitgesetze	119
7.2	Konkurrenz der Beobachter	122
7.3	Laufzeitbetrachtungen	126
7.4	Eine potentielle Verbesserung des Verhaltens des technischen Prozesses	133
7.5	Schlußbemerkungen	140
8.	Zusammenfassung	143

Anhang 1: Verteilungen in Inzidenz- und Adjazenzmatrizen von Graphen

Anhang 2: Verzeichnis der Kurzzeichen

Literaturverzeichnis

1. Einleitung und Aufgabenstellung

1.1 Das Problem der Beurteilung von Programmiersprachen bei Gebrauch und Entwurf

Beschäftigt man sich mit Aufgaben, deren Lösung zu irgendeinem Zeitpunkt den Einsatz von elektronischen Datenverarbeitungsanlagen erfordert, so stellt sich die Frage nach dem Hilfsmittel, mit dem eine Problemlösung so formuliert werden kann, daß "sie in einem automatischen Datenverarbeitungssystem ..... ausgeführt werden" kann /1.1/ : der Programmiersprache[+].

Während man auf dem Gebiet der konventionellen, höheren Programmiersprachen über praktische Erfahrung von bis zu 20 Jahren verfügt, ist dies auf dem Gebiet der Prozeßprogrammiersprachen[++] anders :

Erstens müssen diese sich von den konventionellen Sprachen u. a. unterscheiden durch :

- Einrichtungen zur Einplanung und Durchführung von Programmen auf bestimmte, externe asynchrone Ereignisse hin,
- Einrichtungen zur Ablaufsteuerung von parallel ablauffähigen Programmen,
- Einrichtungen zum Schutz von Betriebsmitteln gegen ungewollte Interferenzen,
- Einrichtungen zur Handhabung spezieller Datentypen, insbesondere in der Ein-/Ausgabe, wie sie in technischen Prozessen vorkommen.

Zweitens gibt es erst seit einigen Jahren Anstrengungen, auch höhere Programmiersprachen (die bedeutendsten finden sich in /5.2/) für die Prozeßautomatisierung mit obigen Eigenschaften zu entwerfen und zu implementieren.

[+] DIN 44300 : "Eine zum Abfassen von Programmen geschaffene Sprache" /1.2/

[++] Unter konventionellen Programmiersprachen werden in dieser Arbeit Sprachen für Anwendungsgebiete technisch-naturwissenschaftlicher, mathematischer und kommerzieller Art verstanden. Eine Prozeßprogrammiersprache ist eine "Echtzeit-Programmiersprache für Prozeßrechner" /1.3/. Als höhere Programmiersprache gilt allgemein eine maschinenunabhängige Programmiersprache.

Es fehlt daher ganz allgemein die breite, langjährige Anwendung und daraus-folgend das fundierte Wissen darüber, welche Eigenschaften eine Prozeßprogrammiersprache haben sollte, mit deren Hilfe bestimmte, charakteristische Aufgabenstellungen im zu automatisierenden technischen Prozeß zu lösen sind.

Eine erstrebenswerte Hilfestellung wären sicherlich Maßzahlen, die zur Beurteilung herangezogen werden könnten. Da es schwierig und sehr problematisch ist, im Zusammenhang mit Programmiersprachen zu verläßlichen und aussagekräftigen Maßzahlen zu gelangen, wird in dieser Arbeit ein anderer Weg beschritten.

Nicht die Aussage "so und so gut" wird gesucht. Sondern es werden - im Grunde binäre - Aussagen angestrebt, welche Eigenschaften eine (Prozeß-) Programmiersprache enthalten muß, will man bestimmte, typische Problemstellungen mit deren Hilfe lösen.

Der Verfasser dieser Arbeit ist sich bewußt, daß diese Vorgehensweise - genau wie etwa die der statistischen Messreihen z. B. in /1.1/ - nur ein Beitrag dazu sein kann, Diskussionen über die Notwendigkeit von bestimmten Spracheigenschaften auf eine festere Basis zu stellen.

Diese feste Basis ist aber dringend erforderlich für :

- A) Den Benutzer einer Programmiersprache :
  Vor der Lösung einer konkreten Aufgabe mit Hilfe einer (Prozeß-) Programmiersprache muß der Benutzer mindestens wissen, ob sich eine ihm angebotene oder eine von ihm ins Auge gefaßte Programmiersprache für sein spezielles Problem überhaupt eignet.

- B) Den Entwickler einer Sprache
  Beim Entwurf einer (Prozeß-) Programmiersprache[+)] benötigt der Entwickler neben seinen - nicht weiter meßbaren - Erfahrungen ("know-how") definitive Hilfestellungen bei der Entscheidung, welche Softwarewerkzeuge, d. h. letztlich Eigenschaften er in eine Programmiersprache aufnehmen muß, um dem Benutzer bestimmte Problemlösungen erst zu ermöglichen bzw. bestimmte Problemlösungen nicht grundsätzlich zu verbauen.

+) Der Autor dieser Arbeit war beteiligt an der Weiterentwicklung der Prozeßprogrammiersprache PEARL /1.4/ und hat an einer der ersten Implementationen eines PEARL-Subsets mitgewirkt (/1.5 - 1.9/).

1.2 Abgrenzung der Aufgabenstellung und Gang der Arbeit

Abgrenzung

Weil es nicht möglich ist, alle Arten von technischen Prozessen hinreichend unter den geschilderten Gesichtspunkten zu analysieren, findet in dieser Arbeit eine Beschränkung auf die Klasse der Stückprozesse statt. Anhand der charakteristischen Aufgabenstellungen in diesem Prozeßtyp sollen fundierte Aussagen über die Anforderungen an eine Prozeßprogrammiersprache gewonnen werden.

Zwei Aufgabenstellungen werden in dieser Arbeit scharf getrennt :

A) Dispositions- oder Planungsaufgaben : Dies sind alle Probleme und Aufgabenstellungen, die sich mit (längerfristigen) Planungsaufgaben im Sinne eines Entwurfs von technischen Anlagen beschäftigen. Dispositionsprobleme werden mit mathematischen Hilfsmitteln und Methoden (z. B. /1.11 - 1.26/) angegangen. Es sind letztlich Aufgaben, die keine oder nur simulierte Realzeitbezüge enthalten. Sie können daher "off-line" auf beliebigen Rechnern zu beliebigen Zeitpunkten gelöst werden. Die Lösbarkeit derlei numerischer Dispositionsprobleme wird in dieser Arbeit nicht zu den typischen, schwerpunktmäßigen Anforderungen an eine Prozeßprogrammiersprache gerechnet, obwohl selbstverständlich in einer Prozeßprogrammiersprache auch Werkzeuge zur Bewältigung numerischer Probleme unerläßlich sind.

B) Operationsaufgaben : Dies sind all die Aufgaben, die während des Betriebs eines (Stück-) Prozesses durch die überwachende und steuernde Instanz "Prozeßrechner" on-line zu bewältigen sind. (Auch die (kurzfristige) Dispositionsaufgabe "Lenke Stück s_i vom Ort A nach Ort B" wird hierbei zu den operativen Aufgaben gezählt, weil diese Aufgabe vom Prozeßrechner zeitgerechte, d.h. Realzeit- oder Echtzeit-Aktionen erfordert). Sie umfassen in der höchsten Automatisierungsstufe die Teilaufgaben

- Prozeßzustandserfassung
- Prozeßzustandsführung anhand von Prozeßmodellen
- Prozeßlenkung.

Eine Prozeßprogrammiersprache dient der Formulierung der "on-line"-Problem-Lösung. Es sind daher die Aufgaben der Kategorie B, die mit Hilfe einer Prozeßpogrammiersprache primär lösbar sein müssen ; aus ihnen rekrutieren sich die charakteristischen, von den konventionellen Programmiersprachen abweichenden, funktionellen Anforderungen an eine Prozeßprogrammiersprache bei der Automatisierung eines Stückprozesses.

Aus diesem Grunde beschäftigt sich die vorliegende Arbeit ausschließlich mit dem Problemkreis B).

Gang der Arbeit

Vorliegende Arbeit umfaßt drei Teile :

1) Grundlegende Untersuchung am technischen Prozeß

   Als eine der Grundlagen der letzlich angestrebten Sprachuntersuchungen wird zunächst die vorliegende Prozeßart analysiert. Daher findet sich im nächsten Kapitel eine Untersuchung der Elemente und der Struktur von Stückprozesses, die in die Bildung von Prozeßmodellen mündet. Eine Diskussion der Verfahren zur Verfolgung und Lenkung von Stücken schließt sich an.

2) Aufbau eines gegenständlichen Modells

   Um den Bezug zur praktischen Anwendung herzustellen, zur Verbreiterung der Untersuchungsgrundlagen und als Beispiel zur Veranschaulichung wurde ein gegenständliches Modell eines Stückprozesses entwickelt, das an einen Prozeßrechner AEG 60-50 angeschlossen wurde und somit geschlossen prozeßgekoppelt ("on-line closed-loop") betrieben werden kann. Eine Beschreibung mit Anwendung der entwickelten Modelle findet sich im Kapitel 4.

3) Untersuchung von Spracheigenschaften

   Die abschließenden Kapitel 5 bis 7 stellen die eigentliche Sprachuntersuchung dar, die sich von den grundlegenden Anforderungen an eine Prozeßprogrammiersprache bei der Automatisierung von Stückprozessen über Mechanismen zur Behandlung vieler freier, parallel anfallender Ereignisse bis zur Behandlung von Konkurrenzsituationen und deren Auswirkungen erstrecken. Als Leitfaden dient hierzu die Prozeßprogrammiersprache PEARL /1.4/.

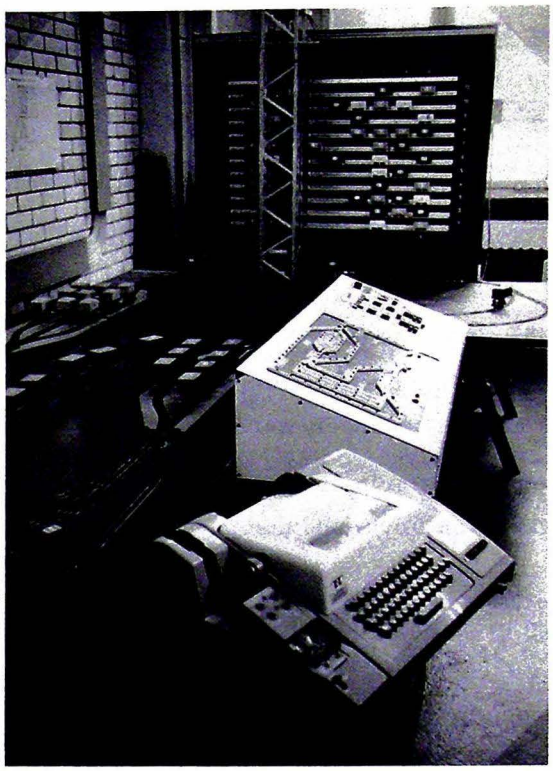


Bild 1.1 : Gesamtansicht des gegenständlichen Aufbaus eines Modells eines Stückprozesses

Eine Untersuchung von Prozeßprogrammiersprachen bliebe ohne die Möglichkeit, eine solche Sprache real anzuwenden und praktische Erfahrungen im Umgang damit zu sammeln, sehr theoretisch. Deshalb sei an dieser Stelle erwähnt, daß der Autor an einem PEARL-Kompiliersystem[+)] mitgewirkt und PEARL bei der Programmierung von realen Prozessen eingesetzt hat. Damit war eine weitere Grundlage für die angestrebte Untersuchung gegeben. Auf eine Beschreibung der Arbeiten am Kompiliersystem sei hier jedoch verzichtet.

+) Das PEARL-Kompiliersystem auf dem Prozeßrechner AEG 60-50 wurde in einer Gemeinschaftsarbeit von A. Ghassemi, R. Wiedenmann, A. Zeh und dem Autor im Rahmen der Arbeitsgemeinschaft ASME am Institut für Regelungstechnik und Prozeßautomatisierung implementiert (siehe u. a. /1.5 - 1.9/, /3.4/, /4.10/, /6.14/).

2. Eine Untersuchung der Elemente und der Struktur des Stückprozesses

2.1 Die Objekte des Stückprozesses : Die Stücke

2.1.1 Definitionen

Die VDI-Richtlinie 2411 /2.1/ definiert ein Stückgut als :

- "Einzelnes, eine Einheit bildendes Gut".

Die VDI-Richtlinie 3565 /2.2/ definiert Stückgüter etwas detaillierter als :

- "... Gegenstände..., die ohne Rücksicht auf Form und Gewicht während des Förderns als Einheit behandelt werden".

Für die weiteren Untersuchungen in dieser Arbeit werden weitere Kriterien neben der Unabhängigkeit von Form und der physikalischen Größe Gewicht herausgestellt.

So soll es hier gleichgültig sein, ob ein bestimmtes Stück einfach, d.h., homogen im Sinn von gleichstofflich ist, also nur aus einem bestimmten Material besteht oder ob ein Stück aus mehreren verschiedenen Materialien in möglicherweise verschiedenen Aggregatzuständen zusammengesetzt ist. Genau so kann es etwa in sich wieder - ohne körperliches Zerschneiden - in Einzelstücke unterteilbar bzw. zerlegbar sein.

Entscheidend für die Betrachtung als "Stück" ist also immer nur das Gebilde - einfach oder zusammengesetzt -, das gerade als Einheit manipuliert wird.

Ein Beispiel für ein einfaches Stück ist etwa das Stück "Messingschraube" : ein solches Stück besteht in aller Regel nur aus dem Material "Messing" und es ist normalerweise körperlich - ohne zerschneiden - nicht in Einzelobjekte zerlegbar.

Ein "Behälter, gefüllt mit Flüssigkeit" ist z.B. ein typischer Vertreter für ein zusammengesetztes Stück : eine Materialie 1 im Aggregatzustand "flüssig" wird von einer Materialie 2 umschlossen ; beide zusammen bilden notwendigerweise die Einheit.

Eine "Palette, beladen mit Kisten" ist ebenfalls ein (sogar mehrfach) zusammengesetztes Stück. Zwar wird diese Palette möglicherweise sehr viele, beispielsweise n Kisten befördern, aber solange letztere alle zusammen-

bleiben, kann man sich auf die Behandlung des Stückes "beladene Palette" beschränken, mit dem ja die Kisten assoziiert sind. Eine Unterteilung bzw. Zerlegung in die Einzelteile "eigentliche Palette" und in n "Kisten" ist bei Bedarf aber ohne weiteres möglich.

Zur Lösung der grundlegenden operativen Aufgaben im Stückprozeß, der Stückverfolgung und -lenkung, ist diese Betrachtungsweise ausreichend. Herrscht nämlich zu einem bestimmten Zeitpunkt Kenntnis über die Position eines zusammengesetzten Objekts, so hat man auch hinreichend Kenntnis über die Position derjenigen Subobjekte, aus denen das Objekt besteht.

In Anlehnung an die erwähnten VDI-Richtlinien und an /2.3/ sei ein Stück für die folgende Untersuchung definiert :

Definition 2.1 : Stücke

Stücke sind abgegrenzte, materielle Einzelobjekte, die ohne Rücksicht auf

Gestalt und Zusammensetzung

während des Förderns als Einheit behandelt werden.

Diese Definition beschreibt nicht, ob es sich bei dem Objekt um ein Transportgut oder etwa um ein - diskretes, frei bewegliches - Transportmittel (oder gar um beides) handelt.

Diese übliche Unterteilung unterbleibt hier ausdrücklich, weil bei der Stückgutlenkung zwischen beiden kein prinzipieller Unterschied besteht ; für beide muß ermittelt werden, wo zu welchem Zeitpunkt welches Stück - ob Transportgut oder Transporthilfsmittel - welche Operationen erfordert. +)

2.1.2 Die Beschreibung von Stücken

Zur Beschreibung eines bestimmten diskreten Objekts - eines Stückes - lassen sich seine physikalischen Merkmale und Eigenschaften heranziehen. Daneben existieren aber auch logische Merkmale und Eigenschaften einer bestimmten, umgangssprachlichen Semantik wie z.B. :

+) In der Studie /2.6/ wird in diesem Zusammenhang deshalb der Begriff der Transporteinheit eingeführt; siehe auch Kapitel 3.

- Typ, Bauart, Bauweise,
- Funktionsweise,
- Verwendungszweck, d.h. Ziel bzw. Zukunft,
- Ursprung, d.h. Geschichte bzw. Vergangenheit,
- Zusammensetzung

usw.

Eine Beschreibung eines Stückes wird üblicherweise gebildet durch Aneinanderreihen von Einzelmerkmalen, d.h. durch Bildung einer Folge von Einzelbeschreibungen, im Folgenden Beschreibungskette genannt. Die Beschreibung gilt i.a. als umso "genauer", je mehr Elemente die Kette enthält. Wie exakt die Beschreibung sein muß, ist jedoch abhängig

a) vom Informationsumfang, den der Verwendungszweck des Objektes erfordert,

b) von der Anforderung an eine Beschreibung, die Unterscheidbarkeit eines einzelnen Objektes von anderen Objekten herbeizuführen.

Hierzu ein Beispiel :

Die Beschreibung eines Stückes in der Form "Schraube, M3 x 10" z.B. beinhaltet die technischen Einzelmerkmale "Schraube metrisches Gewinde" mit den Maßzahlen "3 x 10 mm". Diese Kette besitzt dann genügend Informationsgehalt, wenn beispielsweise der Verwendungszweck "Schrauben in 3 mm Gewinde, 10 mm lang" lautet.

Alle Stücke, auf die jedes Einzelelement der Beschreibung zutrifft, sind gemessen am Verwendungszweck hinreichend genau nach Kriterium a) beschrieben.

Die Anforderung nach Kriterium b), ein bestimmtes Stück von anderen mit Hilfe dieser Beschreibungskette zu unterscheiden, kann nur dann erfüllt

werden, wenn es nur ein einziges Stück in der betrachteten Materialflußanordnung gibt, auf das alle Elemente der Folge von Einzelbeschreibungen der Kette zutreffen. Dieses Kriterium der Unterscheidbarkeit sei in zwei Definitionen niedergelegt:

Definition 2.2 : Eindeutige Stückbeschreibung

Eine Beschreibung eines Stückes wird dann als eindeutig bezeichnet, wenn alle Einzelbeschreibungen der Beschreibungskette nur auf ein einziges Stück in der betrachteten abgegrenzten Materialflußanordnung zutreffen.

Eine eindeutige Beschreibung zeigt so nur auf ein Element s aus der Menge aller Stücke S.

Definition 2.3 : Mehrdeutige Stückbeschreibung

Eine Beschreibung eines Stückes wird dann als mehrdeutig bezeichnet, wenn alle Einzelbeschreibungen der Beschreibungskette auf mehr als ein Stück in der betrachteten, abgegrenzten Materialflußanordnung zutreffen.

Eine solche Beschreibung bezeichnet dann nicht ein einzelnes Stück $s \in S$, sondern eine Untermenge $S' \subseteq S$.

Die Einschränkung bezüglich der Materialflußanordnung ist notwendig, da ein- und dieselbe Beschreibung in einer Anordnung eindeutig, in einer anderen aber mehrdeutig sein kann (zur Abgrenzung siehe Abschnitt 2.3).

Weil ein Stück als fester, starrer Körper - eine Einheit während des Förderns - angesehen wird, sei die Stückbeschreibung in dieser Arbeit etwas statisches : Solange die Einheit nicht körperlich verändert wird, gelte für sie die unveränderte Beschreibungskette aus den entsprechenden

physikalischen und logischen Einzelmerkmalen. So zeitabhängige, flüchtige Attribute wie Geschwindigkeit und Ort sollen dagegen nicht Gegenstand der Beschreibung des Stückes, sondern Gegenstand der Beschreibung des Prozeßzustandes sein.

Wird die Einheit körperlich verändert, so entsteht durch Veränderung der Einzelmerkmale auch eine andere, neue Stückbeschreibung. Kennt man nur die alte und nur die neue Beschreibung, ohne vom Vorgang der Veränderung informiert zu sein, so ist der Rückschluß aus dem Vergleich zwischen jetzigem Prozeßzustand und dem vorherigen so, als wäre ein (neues) Stück entstanden und ein (altes) verschwunden. (Siehe Abschnitt 2.4).

2.1.3 Die Identität von Stücken

Identitätskennung als Beschreibungsersatz

Anstelle der statischen Stückbeschreibung in Form von Einzelmerkmalen benutzt man üblicherweise das Verfahren der Benennung durch Vergabe von Kennungen[+].

Damit erreicht man primär einen Ersatz einer Beschreibungskette durch einfachere, nach beliebigen Kriterien aufgebaute Symbole : der Identitätskennzeichnung oder Identitätskennung.

Identitätskennung als Mittel zur Herstellung von Eindeutigkeit

Entspricht eine gegebene, bestimmte Beschreibungskette nicht einer aufgestellten Forderung nach Eindeutigkeit, so könnte man versuchen, die Beschreibung iterativ so lange weiter zu detaillieren, bis Eindeutigkeit erreicht wird.

+) Allgemein bekannt in diesem Zusammenhang sind die Begriffe Teilenummer oder Artikelnummer. Diese stellen letztlich eine nichtumgangssprachliche, synthetische Benennung dar.

Es leuchtet unmittelbar ein, daß dieses Verfahren in vielen Fällen nicht sehr praktikabel ist.

Ist etwa die Beschreibung "Messingschraube M 3 x 10" in einer bestimmten Materialflußanordnung gemäß der Eindeutigkeitsanforderung nicht hinreichend, so wird man rasch zum Verfahren der Kennungsvergabe greifen, weil es schon nach einigen Iterations-Schritten schwerfällt, praktisch verwertbare, d.h. neue und unterscheidende Merkmale zu entdecken.

Die Vergabe von Identitätskennung stellt somit ein einfaches Verfahren zur Herstellung von Eindeutigkeit dar, die sonst nur durch großen Beschreibungsaufwand erzielt werden könnte. (Selbstverständlich wird in diesem Zusammenhang vorausgesetzt, daß hieraus nicht zusätzliche Mehrdeutigkeiten entstehen).

Identität bei der Stück-Identifizierung

Eine wichtige Rolle spielt die Identität beim Vorgang der Stück-Erkennung. Dabei will man an verschiedenen Orten der Stückgutanordnung Stücke identifizieren, um so den Fluß der Stücke zu überwachen. Man könnte anhand der Einzelmerkmale - am Ort der Identifizierung festgestellt - aus der Menge aller Merkmale auf die Identität des betreffenden Stückes schließen.

Diese Vorgehensweise scheitert aber möglicherweise schon an Art und Zahl der festzustellenden Einzelmerkmale. So kann beispielsweise das Stück A mit der willkürlich als eindeutig angenommenen Beschreibungskette

A : = "Messingschraube M 3 x 10"

anhand der Beschreibungen der Stücke

B : = "Messingstück" oder
C : = "Zylindrisches Stück, 10 mm lang"

nicht als das Stück A wiedererkannt, d.h. identifiziert werden. Die Beschreibungsketten A, B und C stimmen weder in Länge noch Inhalt überein (obwohl menschliche Phantasie und Wissen die inhaltliche Übereinstimmung nahelegt).

Wie das Beispiel zeigt, kann nicht angegeben werden, welches Verfahren exakt zur Wiedererkennung eines Stückes führen wird : Weder die vage - und damit unbrauchbare - Vorschrift "bilde möglichst große Kette" noch die exaktere Vorschrift "bilde Kette aus n Elementen" gewährleistet den Erfolg des Identifizierungsversuchs.

Die Verwendung der Identitätskennung stellt somit die einzig praktikable Möglichkeit dar, beliebige Stücke in definierter Weise zu identifizieren. An die Stelle der Feststellung von Einzelmerkmalen tritt dann die Ablesung der Kennung, die das Stück als eine Art "Paß " oder Etikett begleitet.

Identitätskennung als Kommunikationshilfsmittel

Als Ersatz der Beschreibungskette entworfen, stellt die Identitätskennung selbstverständlich - genau wie die originäre Beschreibung - primär ein Kommunikationshilfsmittel dar.

Wenn alle Kommunikationspartner über die Zuordnungsliste Identitätskennung zu Beschreibungskette verfügen, so kann aus der (synthetischen) Kennung die (umgangssprachliche) Folge von Einzelmerkmalen wiedergewonnen werden.

Aus folgenden Gründen ist die Darstellung in Form von synthetischen Kennungen der originären Darstellung überlegen :

- a) Zur Kommunikation zwischen allen Beteiligten werden lediglich Symbole oder Zeichenketten benötigt, die durch die bekannten Verfahren der Nachrichtentechnik, z.B. /2.8/, mittels gezielter Redundanz gegen Übertragungsfehler tolerant gestaltet werden können.
- b) Mißverständnisse und Mehrdeutigkeiten aus umgangssprachlicher Ungenauigkeit und ungezielter umgangssprachlicher Redundanz sind leicht vermeidbar.
- c) Die Ablesung und Überprüfung von Identitätskennungen kann leicht von Automaten vorgenommen werden.

2.1.4 Zusammenfassung

Die Analyse über die Objekte des Stückprozesses ergab, daß die Beschreibung eines Stückes zweckmäßigerweise mit Hilfe von Identitätskennzeichen vorgenommen wird. Es dient als

1) Beschreibungsersatz bzw. Abkürzung
2) Hilfsmittel zur Herstellung von Eindeutigkeit
3) Hilfsmittel bei der Stückidentifizierung
4) Kommunikationshilfsmittel

In den folgenden Untersuchungen wird ein Stück ausschließlich mit dem Hilfsmittel des Identitätskennzeichens beschrieben. Dann muß wegen 3) vorausgesetzt werden, daß diese Kennung das Stück stets begleitet, da die Vorgänge bei der Vergabe nicht reversibel seinkönnen (das Begleiten muß dabei nicht unbedingt gegenständlich sein).

2.2 Klassifizierung von Stückprozessen

Wie erläutert, kann es sinnvoll sein, jedem Stück einer Materialflußanordnung eine Identität zuzuordnen, und damit letztlich jedes Stück zu etikettieren.

Diese Zuordnung Stück - Identitätskennung muß nicht notwendigerweise umkehrbar in dem Sinne sein, daß jede Identität nur ein Stück beschreibt.

Es sind zwei extreme Situationen denkbar : in einer Stückgutanordnung existieren nur

- Anonyme Stücke
  Die Stücke besitzen keine eigene Identitätskennung, oder - was bezüglich des Informationsgehaltes gleichbedeutend ist -, sie besitzen alle dieselbe Identitätskennung. Dann können die Stücke mittels Identitätskennzeichen nicht voneinander unterschieden werden.
- Eindeutig identifizierbare Stücke
  Jedes Stück hat seine eigene, nur ihm zugehörige Identitätskennung, die eindeutig nach Definition 2.2 ist. So kann jedes Stück vom anderen durch diese Identität unterschieden werden.

Anhand dieser Unterscheidung lassen sich Stückprozesse in zwei extreme Klassen unterteilen :

Definition 2.4 : Anonyme Stückprozesse

Ein anonymer Stückprozeß ist eine Stückgutanordnung, in der die Stücke nicht voneinander unterschieden werden.

Definition 2.5 : Stückprozeß mit eindeutig identifizierbaren Stücken

Stückprozesse mit eindeutig identifizierbaren Stücken sind Stückprozesse, in denen jedes Stück von jedem anderen unterschieden wird.

Im anonymen Stückprozeß genügt die Beschreibung des Prozeßzustandes durch Stückströme und Stückbilanzen. Es ist keine bessere, genauere Auflösung beabsichtigt (oder möglich). Gegenüber den kontinuierlichen Massenströmen wird lediglich als Detaillierung die Maßzahl "Stück (je Zeiteinheit)" angegeben.

Als Beispiel sei etwa eine Abfüllanlage genannt; die diskreten Einzelobjekte sind die abzufüllenden Behälter, die man i. a. nicht voneinander unterscheidet. Stücke werden daher nicht einzeln verfolgt und auch nicht gelenkt, sondern nur die entsprechenden Ströme.

Im Stückprozeß mit eindeutig identifizierbaren Stücken ist im Gegensatz hierzu eine genauere Auflösung möglich, d. h. eine detaillierte Beschreibung, wo sich welches individuelle Stück zu welchem Zeitpunkt befindet.

Zwischen beiden Extremen existieren Abstufungen und sogar Vermischungen.

So gibt es Stückprozesse, in denen zwar jedes Stück eine eigene Identität hat, diese aber nicht eindeutig ist. So erhält z. B. ein Brief in einer automatischen Briefverteilanlage eine Benennung, die Zielpostleitzahl, aber es gibt i. a. viele Briefe mit derselben Identität.

Ob ein Stückprozeß als anonym oder nicht anonym im Sinne der Definition 2.5 zu bezeichnen ist, hängt nur von der gestellten Aufgabe im Prozeß, nicht aber von der Identifizierungsmöglichkeit an sich ab.

Als Beispiel sei der motorisierte Straßenverkehr angeführt : Zwar wäre jedes Stück (Pkw, Lkw, Motorrad) einzeln und eindeutig identifizierbar anhand des Etiketts, der Zulassungsnummer des Kraftfahrzeugs, trotzdem ist (bis

heute) der Straßenverkehr vom Standpunkt des Verkehrswesens ein Prozeß, in dem nur (Verkehrs-)Ströme gelenkt werden /2.5/.

Ein Mischfall wäre etwa in diesem Zusammenhang die Lenkung eines bestimmten Fahrzeugs (z.B. Feuerwehrfahrzeug, Krankenwagen etc.) durch den Straßenverkehr, also die Lenkung eines einzelnen identifizierbaren Stückes durch eine anonyme Stückgutanordnung.

Aus dieser extremen Klassifizierung ergeben sich die Grenzen der Anforderungen bei der Automatisierung von Stückprozessen :

a) im anonymen Stückprozeß beschränkt sich die Aufgabe darauf, "Ströme" im weitesten Sinn zu erfassen und zu lenken.

b) Im Stückprozeß mit einzelnen und eindeutig identifizierbaren Stücken besteht die Aufgabe darin, jedes Stück als eine Art Individuum zu behandeln. Es muß durch die gesamte Stückgutanordnung individuell verfolgt und individuell gelenkt werden.

Selbstverständlich entstehen durch die individuelle Lenkung vieler Stücke ebenfalls "Ströme", die Aufgabe a) ist so in b) enthalten. Deshalb wird die Aufgabenstellung b) im Folgenden bevorzugt behandelt.

2.3 Die geschlossene Materialflußanordnung als Übertragungsglied

Betrachtet man die Materialzu- und Abflüsse einer beliebigen geschlossenen Materialflußanordnung, so scheint der Zweck der Anordnung zu sein, Materialien in irgendeiner Weise umzuschlagen bzw. zu manipulieren.

Die "wirkungsmäßige Betrachtungsweise" (nach /2.7/) des Übertragungsgliedes "geschlossene Materialflußanordnung" läßt nur in Sonderfällen direkte analytische Beziehungen zwischen den Ausgangs- und Eingangsgrößen - den Materialströmen - erkennen und damit die Erstellung eines analytisch leicht faßbaren Prozeßmodells zu.

Für jede beliebige geschlossene Materialflußanordnung läßt sich dagegen immer eine Massenbilanz, für geschlossene Stückgutanordnung zusätzlich noch eine Stückzahlbilanz angeben, die den Durchsatz der Anordnung beschreibt.

2.3.1 Die abgeschlossene Materialflußanordnung

VDI-Richtlinie 2411 /2.1/ definiert ein (Materialfluß-)System als "... ein zwischen Eingang ... und Ausgang ... abgegrenzter Bereich".

In dieser Arbeit soll unter Eingang bzw. Ausgang ausschließlich eine Einrichtung verstanden werden, mittels derer Materie die Anordnung betritt bzw. verläßt; es werden also nur Schnittstellen betrachtet, die dem Materialfluß dienen. Die Schnittstelle(n) Eingang bzw. Eingänge sind die Verbindungen von Umwelt zur Materialflußanordnung, die letztere beeinflussen. Die Schnittstelle(n) Ausgang bzw. Ausgänge sind die Verbindung, die auf die Umwelt einwirken.

Definition 2.6 : Geschlossene Materialflußanordnung

Eine geschlossene Materialflußanordnung ist eine Anordnung, die durch eine gedachte oder reelle Hüllfläche von ihrer Umgebung abgetrennt ist. Alle Elemente innerhalb der Hülle sind Bestandteil der Anordnung, alle Elemente außerhalb sind Bestandteile der Umwelt. Die einzigen, Materie betreffenden Schnittstellen zwischen Anordnung und Umwelt stellen N Eingänge und M Ausgänge dar, durch die Materialien in die Anordnung gelangen bzw. durch die Materialien die Anordnung verlassen.

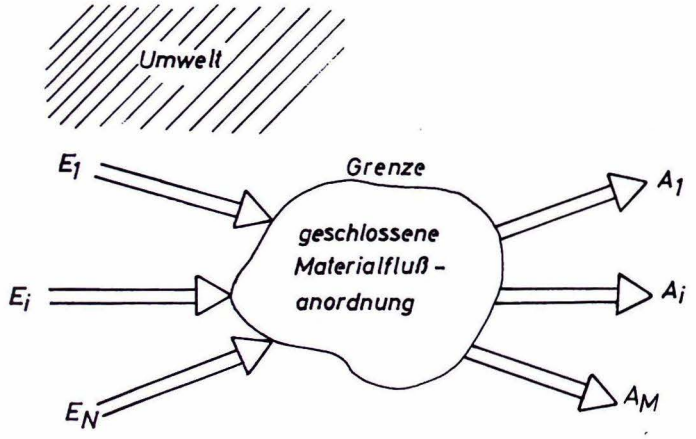


Bild 2.1 : Bezüge der geschlossenen Materialflußanordnung zur Umwelt

Werden mit der Materialflußanordnung einzelne Stücke manipuliert, so wird sie zur Stückgutanordnung.

Definition 2.7 : Geschlossene Stückgutanordnung : Stückprozeß

Eine geschlossene Stückgutanordnung - ein Stückprozeß - ist eine geschlossene Materialflußanordnung, die einzelne Stücke manipuliert. Stücke betreten die Anordnung durch $N' \leq N$ Stückgut-Eingänge, durch $M' \leq M$ Stückgut-Ausgänge verlassen sie die Anordnung[+].

Was zur Materialflußanordnung und was zur Umwelt gehören soll, ist eine reine Frage des Untersuchungszweckes. So werden in der Definition der VDI-Richtlinie 2411 /2.1/ als Beispiele für Materialflußsysteme angeführt : Arbeitsplatz, Abteilung, Werkstätte, Betrieb und Werk. Die Richtlinien 2689 und 3300 /2.9/ unterscheiden außerwerklichen, innerwerklichen, innerbetrieblichen Materialfluß und Materialfluß am Arbeitsplatz. Genauso könnte man jedes Land, jede Stadt, jede Region als Materialflußanordnung bezeichnen. Charakteristisch ist, daß je nach Untersuchungszweck aus irgendeiner übergeordneten Materialflußanordnung irgendein Teil - ein Modul - herausgelöst wird. Zwangsläufig ist so jede geschlossene Materialflußanordnung in eine Hierarchie eingebettet /2.3, 2.4/.

2.3.2 Der Durchsatz der geschlossenen Materialflußanordnung

Die Massenbilanz

Für jede beliebige, geschlossene Materialflußanordnung läßt sich die Massenbilanz über alle Ein- und Ausgänge bilden :

$$(2.1) \quad \sum_{i=1}^{N} \int_{t=T_1}^{T_2} \dot{m}_{Ei}(t)\,dt + m_{MF}(T_1) = \sum_{j=1}^{M} \int_{t=T_1}^{T_2} \dot{m}_{Aj}(t)\,dt + m_{MF}(T_2)$$

[+] Die $\Delta N = N - N'$ Eingänge und die $\Delta M = M - M'$ Ausgänge, mit denen keine Stücke manipuliert werden, sollen nicht als Schnittstellen zur Umwelt in Bezug auf Stücke gelten.

Dabei bedeuten :

N	: Zahl der Eingänge
M	: Zahl der Ausgänge
$m_{MF}(T_1)$	: Masse in der Materialflußanordnung zu Beobachtungsbeginn $t=T_1$
$m_{MF}(T_2)$	: Masse in der Materialflußanordnung zum Beobachtungsende $t=T_2$
$\dot{m}_{Ei}(t) = \frac{dm_{Ei}(t)}{dt}$	: Massenstrom am Eingang E_i
$\dot{m}_{Aj}(t) = \frac{dm_{Aj}(t)}{dt}$	: Massenstrom am Ausgang A_j

Das Beobachtungsintervall ergibt sich zu

$$(2.2) \qquad \Delta T = T_2 - T_1$$

Dem Beobachter der Anordnung ist i.a. weder $m_{MF}(T_1)$ noch darausfolgend $m_{MF}(T_2)$ bekannt ; beide Größen sind jedoch in diesem Zusammenhang uninteressant. Bildet man nämlich die Differenz

$$(2.3) \qquad \Delta m(\Delta T) = m_{MF}(T_1) - m_{MF}(T_2)$$

$$= \sum_{j=1}^{M} \int_{t=T_1}^{T_2} \dot{m}_{Aj}(t)\,dt - \sum_{i=1}^{N} \int_{t=T_1}^{T_2} \dot{m}_{Ei}(t)\,dt,$$

so erhält man in $+\Delta m(\Delta T)$ die Massendifferenz bezüglich der Umwelt, in $-\Delta m(\Delta T)$ die Massendifferenz bezüglich der Materialflußanordnung.

Die Flußdifferenz ergibt sich zu :

$$(2.4) \qquad \Delta\dot{m}(t) = \sum_{j=1}^{M} \dot{m}_{Aj}(t) - \sum_{i=1}^{N} \dot{m}_{Ei}(t)$$

Bild 2.2 zeigt ein einfaches Beispiel.

Die folgenden Aussagen gelten für das Intervall ΔT :

- Ist Δm nach (2.3) > 0, dann hat mehr Materie die Materialflußanordnung verlassen als zugeflossen ist. Dies könnte wie folgt erklärt werden :
  .. Es gibt in der Anordnung Elemente, die Materialien erzeugen, also Quellen ![+)]
- Ist Δm nach (2.3) < 0, dann ist mehr Material ab - als zugeflossen. Dies könnte wie folgt erklärt werden :
  .. Es gibt in der Anordnung Elemente, die Materialien verbrauchen, also Senken !

Quellen und Senken

Bildet man für ein- und dasselbe Materialflußsystem die Massendifferenz nach (2.3) zu verschiedenen Zeitpunkten, so kann man zu einander widersprechenden Ergebnissen gelangen : in einem Intervall I_1 scheint die Anordnung Quellen, in einem anderen Intervall I_2 scheint sie Senken zu enthalten (vgl. Beispiel Bild 2.2)

Ob eine Materialflußanordnung Quellen oder Senken enthält, kann exakt nur durch genügend lange Beobachtung des Durchsatzes, d.h. während der gesamten Lebenszeit $[0, T]$ der Anordnung entschieden werden [++)]. Wenn in der Lebenszeit aus der Anordnung wieder genau so viel Materie ausströmt wie eingeströmt ist, dann ist die Bilanz ausgeglichen und so muß der Betrachter annehmen, daß weder Senke noch Quelle in der Anordnung existiert. Wird die Massendifferenz nicht zu 0, so ist die Existenz von Senken bzw. Quellen bewiesen.

Definition 2.8 : Materialflußanordnung ohne Quellen und Senken

Eine geschlossene Materialflußanordnung sei dann frei von Quellen und Senken, wenn in ihrer Lebenszeit $[0, T]$ gilt :

+) Theoretisch könnte die Existenz einer Quelle durch Vergleich der Materialien an Aus- und Eingängen festgestellt werden. Dies scheitert aber möglicherweise am Identifizierungsproblem.

++) Jede Materialflußanordnung beginnt zu einem bestimmten Zeitpunkt als solche zu arbeiten ($t = 0$), zu einem anderen Zeitpunkt erlischt ihre Funktion ($t = T$).

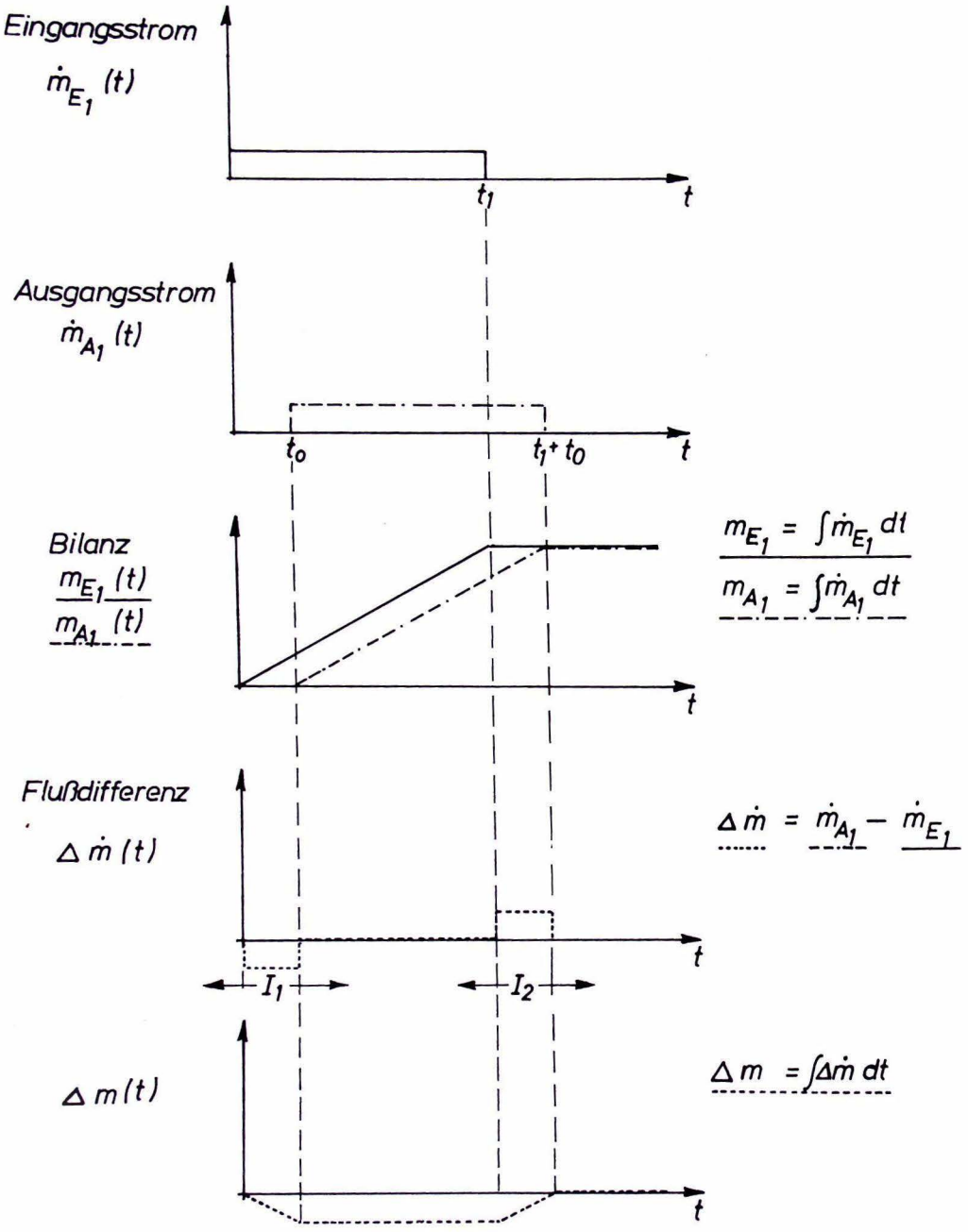


Bild 2.2: Beispiel für Massenbilanz und Flußdifferenz anhand der Materialflußanordnung "Förderband" (M=N=1)

$$(2.5) \qquad \int_{t=0}^{T} \Delta \dot{m}\,(t) = 0 .$$

Definition 2.9 : Materialflußanordnung mit Quellen-Dominanz

Eine geschlossene Materialflußanordnung besitzt dann wenigstens eine Material-Quelle, wenn in ihrer Lebenszeit $[0, T]$ gilt :

$$(2.6) \qquad \int_{t=0}^{T} \Delta \dot{m}\,(t)\,dt > 0 . \qquad \left[\begin{array}{l}\text{Quellen-}\\ \text{Dominanz-}\\ \text{Bedingung}\end{array}\right]$$

Definition 2.10 : Materialflußanordnung mit Senken-Dominanz

Eine geschlossene Materialflußanordnung besitzt dann wenigstens eine Material-Senke, wenn in ihrer Lebenszeit $[0, T]$ gilt :

$$(2.7) \qquad \int_{t=0}^{T} \Delta m\,(t)\,dt < 0 . \qquad \left[\begin{array}{l}\text{Senken-}\\ \text{Dominanz-}\\ \text{Bedingung}\end{array}\right]$$

In den Definitionen 2.9 und 2.10 wird der Begriff der Dominanz benutzt, weil aufgrund der Beziehungen (2.6) bzw. (2.7) nur ausgesagt werden kann, daß eine geschlossene Materialflußanordnung an materieller Substanz verliert bzw. gewinnt.

Eine Aussage darüber, wieviele Quellen bzw. Senken in einer Anordnung existieren, ist nicht möglich. Es kann auch nicht ausgesagt werden, daß z. B. Quellen-Dominanz die Existenz von Senken in einer Anordnung verbietet, sondern nur, daß alle Quellen über die eventuell vorhandenen Senken dominieren.

Die Aussage der Relation (2.5) ist ähnlich zu interpretieren : Frei von Quellen und Senken heißt letztlich nur, daß möglicherweise doch vorhandene Quellen und Senken nicht nach außen aufscheinen, sich also genau aufheben. Will man jedoch genauere Auskünfte über die Zusammensetzung der Materialflußanordnung, so muß man zum Verfahren der Modularisierung/2.4/ greifen. Diese neuen Moduln als Teile der ursprünglichen Anordnung stellen wiederum Materialflußanordnungen dar.

Damit kann je nach Wahl der Modulgrenzen durch die Anwendung der hergeleiteten Beziehungen lokalisiert werden, in welchen Moduln sich Quellen bzw. Senken befinden. Durch Abzählen erhält man dann die (Mindest-)Anzahl an Quellen und Senken der ursprünglichen Gesamtanordnung.

Speicherung

Aus der Langzeit-Beobachtung ergaben sich bisher zwei Elemente, die in Materialflußanordnungen existieren : Senken und Quellen. Aus der Kurzzeit-Beobachtung ergibt sich die Existenz eines weiteren Grundelements : Ist nämlich das Kriterium (2.5) der Quellen- und Senken-Freiheit erfüllt, so läßt die kurzzeitige Schwankung der Flußdifferenz nur den Schluß zu :

> Es gibt in der betrachteten Materialflußanordnung Elemente, die in der Lage sind, Materialien aufzunehmen, eine bestimmte Zeit aufzubewahren und irgendwann - jedoch in der Lebenszeit der Anordnung - wieder abzugeben : Materialpuffer oder Speicher.

2.3.3 Die Stückzahlbilanz

Werden in der Materialflußanordnung Stücke manipuliert, dann kann noch die Stückzahlbilanz angegeben werden :

$$(2.8) \qquad \sum_{i=1}^{N'} s_{Ei}(\Delta T) + s_{MF}(T_1) = \sum_{j=1}^{M'} s_{Aj}(\Delta T) + s_{MF}(T_2)$$

Dabei bedeuten :

N'	:	Zahl der Stückguteingänge
M'	:	Zahl der Stückgutausgänge
$s_{MF}(T_1)$	:	Zahl der Stücke, die sich zum Zeitpunkt $t=T_1$ in der Stückgutanordnung befinden
$s_{MF}(T_2)$	:	Zahl der Stücke, die sich zum Zeitpunkt $t=T_2$ in der Stückgutanordnung befinden.

$s_{Ei}(\Delta T)$: Zahl der Stücke, die der Anordnung im Beobachtungsintervall über den Eingang E_i zufließen

$s_{Aj}(\Delta T)$: Zahl der Stücke, die im Beobachtungsintervall über den Ausgang A_j abfließen

Ähnlich wie bei der Massenbilanz interessiert auch hier lediglich die Stückzahldifferenz im Beobachtungsintervall ΔT :

$$(2.9) \quad \Delta s(\Delta T) = s_{MF}(T_1) - s_{MF}(T_2) = \sum_{j=1}^{M'} s_{Aj}(\Delta T) - \sum_{i=1}^{N'} s_{Ei}(\Delta T)$$

Ihre Untersuchung führt auf dieselben 3 Elemente wie im letzten Abschnitt, die jetzt Bestandteil einer Anordnung sind, die Stücke manipuliert :

- Quellen
- Senken
- Speicher

Eine Stücke behandelnde, geschlossene Materialflußanordnung - ein Stückprozeß - sei dann frei von Quellen und Senken, wenn in ihrer Lebenszeit $[0, T]$ gilt :

$$(2.10) \quad \Delta s(T) = 0 .$$

Sie ist quellendominant in Bezug auf Stücke, wenn gilt :

$$(2.11) \quad \Delta s(T) > 0.$$

Senkendominant in Bezug auf Stücke, wenn gilt :

$$(2.12) \quad \Delta s(T) < 0.$$

Der Zusatz "in Bezug auf Stücke" ist deshalb erforderlich, weil sich die Aussagen der Beziehungen (2.5) ... (2.7) der Massenbilanz und (2.10) ... (2.12) der Stückzahlbilanz nicht notwendigerweise entsprechen.

Anhand der Stückzahlbilanz bzw. -differenz kann nur ausgesagt werden, ob eine bestimmte Anordnung mehr, weniger oder gleichviel Stücke abgibt als sie aufnimmt, nicht aber, ob sie dabei an materieller Substanz etwa gewinnt oder verliert.

Viele Stückprozesse, wie Einrichtungen zur Fertigung und Kommissionierung haben zwar eine ausgeglichene Massenbilanz, aber eine mehr oder weniger stark unausgeglichene Stückzahlbilanz. Ein charakteristisches Beispiel hierfür ist etwa der Stückprozeß "Warenhaus" : Die Zahl der abgehenden Stücke (Verkäufe) ist i.a. bedeutend größer als die Zahl der zugehenden Stücke (Lieferungen) ; der Sinn eines Warenhauses liegt ja schließlich in der Vereinzelung von Waren. Die Massenbilanz über alle Stückgut-Ein- und Ausgänge ist jedoch ausgeglichen.

2.3.4 Zusammenfassung

Der Ansatz der Bilanzierung führte zum Rückschluß auf drei mögliche grundlegende Elemente, die Bestandteile der beliebigen Materialfluß- bzw. Stückgutanordnung sein können :

Definition 2.11 : Material-/Stück-Quellen

Eine Material-/Stück-Quelle ist ein Element, das Materialien/Stücke produziert.

Definition 2.12 : Material-/Stück-Senke

Eine Material-/Stück-Senke ist ein Element, das Materialien/Stücke konsumiert.

Definition 2.13 : Material/Stück-Speicher

Ein Material-Stück-Speicher ist ein Element, das Materie/ Stücke aufbewahrt und wieder abgibt.

2.4 Elemente des Stückprozesses

2.4.1 Arbeitsgänge in Materialflußanordnungen

Die Manipulation von Materialien bzw. Stücken erfolgt mit entsprechenden Einrichtungen, sogenannten "Arbeitsmitteln" /2.3/. (Die Fragestellung nach der technologischen Realisierung dieser Arbeitsmittel ist ausdrücklich nicht Gegenstand dieser Arbeit.) Der Versuch, die Funktionsweise einer beliebigen, geschlossenen Materialflußanordnung anhand von Bilanzen zu beschreiben, führte auf die 3 Grundelemente "Quelle", "Senke" und "Speicher". Ihnen lassen sich folgende Arbeitsgänge zuordnen :

Element (Arbeitsmittel)	Manipulation (Arbeitsgang, Funktion)
Quelle	Erzeugen, d.h. Materie/ Stücke der Anordnung neu zuführen
Senke	Verbrauchen, d.h. Materie/Stücke der Anordnung entnehmen
Speicher	Materie/Stücke puffern

2.4.2 Eine Ergänzung : Die Förderstrecke

Zur Erklärung des Materialflusses sind diese Elemente nicht hinreichend ; man benötigt dazu als Ergänzung eine Verbindung der Elemente untereinander : Transportwege oder Förderstrecken[+)] (genauer: Räume, vgl. Kap. 3). Erst solche Bindeglieder ermöglichen den Material-"Fluß". Aus der Anordnung hinaus über Ausgänge, in die Anordnung hinein über Eingänge und innerhalb der Anordnung zwischen den einzelnen Elementen.

Natürlich kann eine Förderstrecke wiederum als Materialflußanordnung angesehen werden, denn sie ist sicherlich eine Anordnung zum Zwecke des Materialdurchsatzes.

+) Nach /2.1/ "ein Förderweg bestimmter Länge". Die Länge des Weges sei hier zunächst ohne Bedeutung.

Definition 2.14 : Elementare, gerichtete Förderstrecke

Eine elementare, gerichtete Förderstrecke ist eine geschlossene Materialflußanordnung mit folgenden ausgezeichneten Eigenschaften :

a) Sie besitzt nur einen Eingang und einen Ausgang.

b) Ein- und Ausgang legen die Richtung des Materialflusses fest, es gilt $\dot{m}_E(t)$, $\dot{m}_A(t) \geq 0$.

c) Massen- und Stückzahlbilanzen nach (2.1) bzw. (2.8) sind ausgeglichen, d.h. es gilt (2.5) und (2.10).

d) Materialien, am Eingang aufgegeben, erreichen erst nach einer bestimmten, nicht unterschreitbaren Mindest-Verzögerungs- oder Totzeit $T_T \leq T$ [+)] den Ausgang.

Ein einfaches Beispiel hierzu beinhaltet Bild 2.2.

Der elementaren, gerichteten Förderstrecke, die ausschließlich Stücke befördert, lassen sich zusätzlich die folgenden Eigenschaften zuordnen:

e) Jedes einzelne Stück, das am Eingang aufgegeben wird, erscheint integer, d.h. in keiner Weise verändert wieder am Ausgang.

f) Alle Stücke erscheinen in derselben Reihenfolge am Ausgang wie sie am Eingang aufgegeben wurden, d.h. Überholvorgänge sind ausgeschlossen.

Insbesondere die Eigenschaft f) unterscheidet das Element "elementare, gerichtete Förderstrecke" vom Element "Speicher". Trotzdem ist der temporäre Speichereffekt nach d) charakteristisch für die Eigenschaften einer Materie befördernden Strecke ; er wird häufig in Perversion des Begriffs "befördern" in Form von Staustrecken, Linien-, Umlauf-, Durchlauflagern etc. genutzt.

Oft existieren in einer Materialflußanordnung jedoch Strecken, deren Transporteinrichtung bidirektional ist, z.B. jede gewöhnliche Straße. Sie lassen sich durch zwei elementare Förderstrecken mit jeweils gegensinniger Transportrichtung substituieren (möglicherweise darf dann jeweils zu einem bestimmten Zeitpunkt nur eine von beiden benutzt werden).

+) T ist die Lebenszeit der Materialflußanordnung (vgl. 2.3.2), zu der das Element "Förderstrecke" gehört.

2.4.3 Elemente des Stückprozesses, zugeordnete Arbeitsgänge und Substitutionen

Mit den drei Grundelementen Quelle, Senke und Puffer und dem Hilfselement Förderstrecke lassen sich die relevanten Arbeitsgänge mit Stücken und damit die Vorgänge in Stückprozessen hinreichend erklären und darstellen :

Element (Arbeitsmittel)	Manipulation (Arbeitsgang)
1) Quelle	Stück erzeugen - produzieren - , d.h. der Stückgutanordnung zuführen
2) Senke	Stücke verbrauchen - konsumieren - , d.h. der Stückgutanordnung entnehmen
3) Puffer (Speicher)	Stücke aufnehmen, während einer bestimmten Zeitdauer aufbewahren und dann abgeben
4) Förderstrecke	Stücke zwischen den Elementen 1), 2) und 3) bewegen gemäß Definition 2.14

Tabelle 2.1 : Die vier Grundelemente des Stückprozesses und zugehörige Arbeitsgänge

Nun sind durch die in Tabelle 2.1 angeführten noch lange nicht alle denkbaren Detailarbeitsgänge mit Stücken in Stückprozessen beschrieben. Alle anderen lassen sich jedoch auf diese zurückführen durch Substitution.

Die Arbeitsgänge bzw. Vorgänge "Gewinnen", "Transportieren", "Lagerung" und "Aufenthalt" nach /2.9/ sind durch die angeführten Elemente abgedeckt. (Daneben sind in /2.9/ noch "Be- und Verarbeiten" und "Prüfen" angeführt, nicht erwähnt wird der Vorgang des Verbrauchens.)

Das Be- oder Verarbeiten setzt ein Element voraus, das Stücke mit bestimmten Merkmalen (vgl. 2.1) aufnimmt, und das Stücke mit bestimmten, anderen Merkmalen abgibt. Dieses Element ist dann eine geschlossene Materialfluß-anordnung (ein Stückprozeß für sich), deren Massen- und Stückzahlbilanz ausgeglichen sein kann oder auch nicht.

Das Verhalten eines solchen Elementes[+)] läßt folgende Interpretation zu :

- Es gibt in der Anordnung wenigstens eine Senke, die alle Stücke konsumiert, die die Anordnung betreten.
- Es gibt in der Anordnung wenigstens eine Quelle, die die Stücke produziert, die an den Ausgängen erscheinen.
- Ist die Bilanz nicht ausgeglichen, so gibt es zusätzliche Quellen oder Senken zur Deckung des entsprechenden Defizits.

Damit kann das Element, das die Manipulation "Be- bzw. Verarbeiten" bewältigt, auf die ermittelten Elemente zurückgeführt, d.h. durch diese substituiert werden.

Diese Interpretation hat den folgenden Hintergrund :

Ein Stück, von einer bestimmten Be- oder Verarbeitungsstation behandelt, hat vor der Behandlung sicherlich andere Beschreibungsmerkmale als nach der Behandlung.

Aus den in 2.1.2 dargelegten Gründen versah man jedes Stück im Stückprozeß mit einer bestimmten Identitätskennung, die abkürzend u.a. auch den Bearbeitungszustand vor der Behandlung wiedergibt. Um behandelte von unbehandelten Stücken unterscheiden zu können, erhalten die Stücke, die die Bearbeitungsstation verlassen, so zwangsläufig neue Identitäten; es liegt also eine Art von Wandlung vor.[++)]

So ist es nur naheliegend, diese Wandlung als Folge von "Verschwinden" und "Auftauchen" zu bezeichnen, d.h. die Substitution durch Quellen und Senken in der geschilderten Weise vorzunehmen.

Es ist exakt diese Substitution, die bei der Stückgutverfolgung mit Rechnern realisiert werden muß : Stücke, die eine solche Bearbeitungsstation betreten,

---

+) Häufig gibt es für derlei Be- oder Verarbeitungsstationen im Gegensatz zum gesamten Stückprozeß einfach faßbare Mengen- und Stückzahlrelationen zwischen Ein- und Ausgängen.

++) Es ist nicht notwendig, daß diese Identitäten jedes Stück physikalisch begleiten. Es genügt auch virtuelle Begleitung, beispielsweise die Notation im Prozeßmodell eines Rechners.

werden im Prozeßmodell des Rechners letzlich gestrichen ("Verschwinden"). Stücken, die die Station verlassen, wird beim Austritt eine neue Identität zugewiesen ("Auftauchen").

2.5 Elementare Prozeßmodelle als Grundlage zur Prozeßführung

Nach der Zusammenstellung der Elemente von Stückprozessen werden in diesem Abschnitt einige elementare Prozeßmodelle - "die Beschreibung oder Nachbildung eines Prozesses aufgrund des Ergebnisses einer Prozeßerkennung" /2.10/ - entwickelt. Es handelt sich dabei nicht um Modelle, wie sie bei den mathematischen Methoden zur Planung oder Quantifizierung von Stückgut- oder allgemeinen Materialflußprozessen Verwendung finden, um zu globalen Aussagen über Dispositions- und Durchflußprobleme, über Kosten, Nutzen, Effektivität o.ä. zu gelangen. Vielmehr sind dies Modelle, die man zur Lösung der operativen Grundaufgaben im Stückprozeß, der Verfolgung und Lenkung von Stücken benötigt :

a) formale Denkmodelle zum Erkennen, zur Abstraktion und zur Veranschaulichung der Vorgänge für den Menschen

b) Prozeßabbilder für den Rechner zum "on-line"-Betrieb des Prozesses, hergeleitet aus den Denkmodellen nach a).

2.5.1 Das Stationenmodell

Allgemein lassen sich Stückgutanordnungen als

- eine Menge V von (Verkehrs-) Knoten oder "Stationen" und
- eine Menge X von elementaren, gerichteten Strecken

darstellen. Die Knoten oder Stationen können dabei beinhalten :

- Eine Anordnung aus Quellen und/oder Senken (z.B. Bearbeitungsstationen oder Ein/Ausgänge nach 2.3)
- Speicher
- Weder Quellen noch Senken noch Speicher. Dann ist dieser Knoten rein als Kreuzungspunkt verschiedener Strecken entstanden. Seine Funktion ist die des Zusammenführens und/oder Verzweigens.
- Eine allgemeine, untergeordnete Stückgutanordnung, die aus bestimmten Gründen nicht weiter detailliert werden soll oder kann.

Mit diesen Stationen und den Förderstrecken läßt sich ein anschauliches Modell in Form eines gerichteten Graphen oder Digraphen[+)] bilden. Den $p = |V|$ Stationen sollen die p "Ecken", den $q = |X|$ Förderstrecken die q "Kanten" des Digraphen entsprechen (Bild 2.3), in dem jede Kante mit $x_i \in X$ und jede Ecke mit $v_j \in V$ bezeichnet wird (Indizierung).

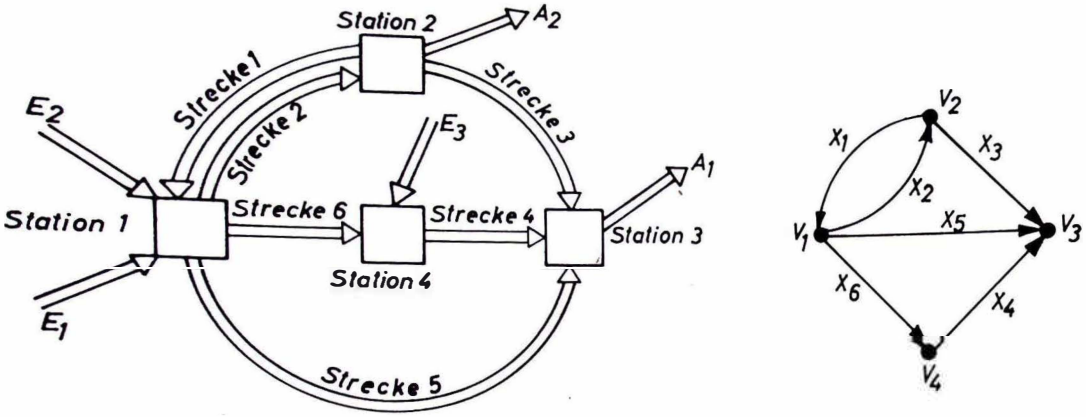


Bild 2.3: Modell einer Stückgutanordnung als zusammenhängender, indizierter Graph

Der Vorteil der graphischen Darstellung ist die Anschaulichkeit, zumindest bei kleineren Graphen. Sie ist aber für die Darstellung der Relationen benachbart bzw. inzident in Rechnern nicht geeignet. Diese lassen sich dagegen in der rechnergeeigneteren Form von Matrizen angeben, deren wichtigste im folgenden angeführt sind:

(2.13) $B = (b_{ij})$ Inzidenzmatrix

mit

(2.14)
$$b_{ij} = \begin{cases} +1, & \text{wenn Ecke } v_i \text{ mit Kante } x_j \text{ inzidiert und die Kante } x_j \text{ von } v_i \text{ weggerichtet ist} \\ -1, & \text{wenn Ecke } v_i \text{ mit Kante } x_j \text{ inzidiert und die Kante } x_j \text{ auf } v_i \text{ zu gerichtet ist} \\ 0 & \text{sonst} \end{cases}$$

+) Terminologie nach Harary /2.11/, die im Zusammenhang mit graphentheoretischen Problemen in dieser Arbeit verwendet wird.

Die quadratische (p, p) Nachbar- oder Adjazenzmatrix beschreibt direkt, welche Ecken eines ungerichteten Graphen benachbart sind ; sie ist so symmetrisch zur Hauptdiagonalen. Die ebenfalls quadratische (p, p) Nachbarmatrix des Digraphen ist dagegen i. a. nicht mehr symmetrisch, weil nicht notwendigerweise zu einer gerichteten Kante $v_i v_j$ auch eine gerichtete Kante $v_j v_i$ existieren muß (siehe Bild **2.** 3).

(2.15) $A = (a_{ij})$ Nachbarmatrix (Adjazenzmatrix)

mit

(2.16) $$a_{ij} = \begin{cases} 1, & \text{wenn } v_i v_j \text{ bzw. } v_j v_i \text{ eine Kante des Digraphen ist} \\ 0 & \text{sonst} \end{cases}$$

Es gibt eine ganze Reihe von weiteren Darstellungsmöglichkeiten, wie z. B. Erreichbarkeits-, Distanz-, Verbindungsmatrix /2.12/ etc., die letzlich alle aus (2.13) ableitbar sind. Ein Beispiel für Adjazenz- und Inzidenz-Matrix zeigt Bild 2.4.

$$A = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \qquad B = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} -1 & +1 & 0 & 0 & +1 & +1 \\ +1 & -1 & +1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & +1 & 0 & -1 \end{pmatrix} \end{array}$$

Bild 2.4 : Adjazenz- und Inzidenz-Matrizen des Digraphen des Bildes 2.3

2.5.2 Das Wegemodell

Der Begriff der "Station" lehnt sich an die Arbeit /2.4/ an, wo allerdings der Übergang zum Graphen nicht vollzogen wird. Ansätze, Stückgutanordnungen bzw. Materialflußanordnungen als (Stationen-) Graphen darzustellen, sind naheliegend und finden sich bereits bei Ford und Fulkerson /1.25/.

Einen entscheidenden, nicht unmittelbar evidenten Schritt der Abstraktion stellen jedoch m. E. die im Folgenden vorgestellten Modelle dar.

In Stückgutanordnungen, deren überwiegendes oder alleiniges Ziel die Kommissionierung[+)] ist, steht meist die Speichereigenschaft einer Förderstrecke im Vordergrund. Die Verkehrsknoten entstehen sozusagen als Nebenprodukt, eine Be- oder Verarbeitung von Stücken findet in den Stationen nicht statt. Dann ist es zweckmäßig, sich ein (Graphen) Modell zu bilden, in dem die Förderstrecken einer Ecke und die potentiellen Verbindungen von Strecke zu Strecke einer Kante entsprechen. Die Ecken sind dann die Ziele, zu denen Stücke gelenkt werden müssen. Die Kanten entsprechen nun nicht mehr physikalischen Wegstrecken, sondern potentiellen Verbindungen.

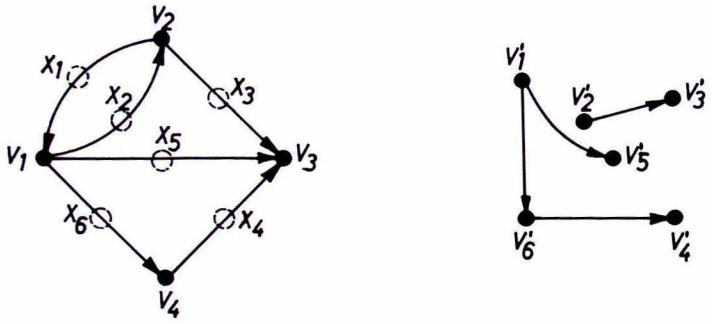


Bild 2.5 : Gegenüberstellung von Stationen- und Wegegraph [++)]

Vorteilhaft an dieser Darstellung ist die direkte Abstraktion des Materialflusses zu den (Kommissionier-) Zielen. Nachteilig ist, daß im Modell "Wegegraph" die Konfliktstellen, nämlich die Stationen, nicht mehr unmittelbar sichtbar sind. Es sind aber die Konflikt-Situationen in Verzweigungen und Zusammenführungen, die in zu automatisierenden Stückprozessen zu bewältigen sind.

Natürlich würde man genau wie beim Stationenmodell in einem Rechner eine der Matrizendarstellungen als Prozeßmodell ins Auge fassen.

+) Nach /2.16/ "die Zusammenstellung von Waren nach vorgegebenen Aufträgen".

++) Der äquivalente Wegegraph ist nicht mehr zusammenhängend. Graphentheoretische Probleme wie Zusammenhang, Erreichbarkeit, Durchlaufbarkeit, Schnitte etc. sind primär Gegenstand des Entwurfs und der Planung von Stückgutanordnungen und erst sekundär Gegenstand der Stückgutlenkung.

2.5.3 Das Wege/Stationenmodell

Eine weitergehende Abstraktion wird erreicht, wenn man wie im Bild 2.6 gezeigt, sowohl Stationen - also z.B. Bearbeitungsstationen, Speicher, Verkehrsknoten - als auch Förderwege als Ecken eines Graphen ansetzt.

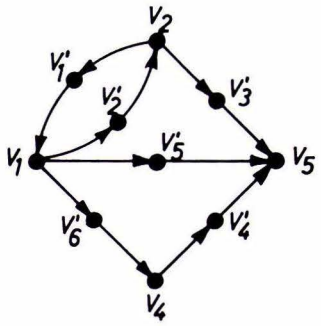


Bild 2.6 : Wege/Stationenmodell der Stückgutanordnung nach Bild 2.3

Dieses gemischte Modell - letzlich aus der Vereinigung des Stationen- mit dem Wegemodell entstanden - , hat die folgenden Eigenschaften :

1) Kanten dienen der idealisierten, quasi "immateriellen" Verbindung der Ecken : Sie geben zwar die Richtung des Materialflußes wieder, sie enthalten aber zu keinem Zeitpunkt Stücke oder Materie.

2) Ecken sind die einzigen, eigentlichen "Materieträger". Sie können zu bestimmten Zeiten Stücke beinhalten oder auch nicht. Ihr Zustand läßt sich binär mit "besetzt" oder "leer", durch die Zahl der Stücke in der Ecke, durch deren Identität etc. beschreiben.

Die Bedeutung dieses (Denk-) Modells liegt darin, daß es bei der abschnittsweisen Stückgutverfolgung (siehe Kapitel 3) eben entscheidend darauf ankommt, den Zustand der Ecken zu kennen, d.h., zu wissen, welche Stücke sich zu welchem Zeitpunkt in welcher Ecke dieses Wege/Stationenmodells befinden. (Welches die physikalische Realisierung einer solchen Ecke ist, ist unter diesem Gesichtspunkt zweitrangig.)

Der "Zustand" der Kanten ist undefiniert und ohne Bedeutung, wohl aber deren Aussage über die Verbindungen.

2.6 Zusammenfassung und Diskussion der Ergebnisse

Ziel der Untersuchungen in diesem Kapitel war es - neben der Zusammenstellung einiger wichtiger Grundlagen - über die Analyse der Elemente und Struktur beliebiger Stückgutanordnungen zu einer fundierten Methode zur Bildung von (in Prozeßrechnern verwendbaren) Prozeßmodellen zu gelangen.

Der Ansatz der Bilanzierung führte zunächst auf die logischen, funktionellen Elemente von Materialflußprozessen. Der Ansatz zur Modellbildung war die Zuordnung der 4 Grundelemente zu den 2 Elementklassen eines per definitionem nicht-maßstäblichen (Di-) Graphen. Bei der wohl naheliegendsten Zuordnung entstand ein Graph, der der geometrischen Anordnung noch mehr oder weniger anschaulich entspricht. Alle anderen hier vorgestellten Modelle sind-obwohl äquivalent- Modelle, die in stärkerem Maße der Abstraktion, d.h., der Loslösung von störenden Details bzw. Gegebenheiten dienen.

Zunächst käme als Prozeßabbild in Rechnern eine der angeführten Matrizendarstellungen in Frage[+].

Beliebige Graphen (bis auf Pseudographen) können durch die Inzidenz-Matrix äquivalent dargestellt werden, so daß man im allgemeinsten Fall mit einer (p, q)-Matrix mit einem Platzbedarf von $p * q$ Elementen rechnen muß. Die Nachbarmatrix mit p^2 Elementen ist zwar kleiner, wenn die Zahl der Kanten größer ist als die der Ecken ($q > p$). Sie ist jedoch nur dann äquivalent, wenn der Graph kein Multigraph ist, d.h., wenn es zwischen zwei benachbarten Ecken v_i, v_j höchstens eine gerichtete Kante $v_i v_j$ ($i, j = 1 \ldots p$) gibt.

Beiden Matrizenarten ist jedoch gemeinsam, daß i.a. viele Matrixelemente mit Nullen besetzt sind. So ist zwar beispielsweise beim vollständigen, gerichteten Graphen - jede Ecke ist mit jeder anderen mit je einer hin- und rückwärts gerichteten Kante verbunden - die Nachbarmatrix auf $p\,(p-1)$ Plätzen mit Elementen $\neq 0$ besetzt ; nur die Diagonalelemente a_{ii} sind 0. Dagegen ist die Inzidenzmatrix desselben Graphen auf $p\,(p-1)\,(p-2)$ Matrizenplätzen mit Nullen besetzt ; wie sich leicht zeigen läßt (Anhang 1), strebt der relative Anteil dieser mit Nullen besetzten Plätze an der gesamten Matrix mit $(1-2/p)$ für größere Eckzahlen rasch gegen 1.

[+] Die Matrizenelemente müssen dabei nicht nur zwei- oder dreiwertig sein. Sie könnten durchaus Kapazitäten, Entfernungen, Belegungen, Kosten o.ä., d.h. also beliebige Zahlen enthalten.

Deshalb wird man in vielen Fällen nicht die primären Modelle in Form der Nachbar- oder Inzidenz-Matrix verwenden, sondern aus Gründen der Speicherplatzersparnis im Rechner auf sekundäre Formen ausweichen, die auf einer der zahlreichen, sehr ökonomischen, aber unanschaulichen Speichertechniken mit Mehrfachverzeigerung o.ä. basieren (z.B. in /2.16/ -/2.18/).

Mit solcher Speicherplatzersparnis einher geht i.a. ein Ansteigen der Zugriffszeiten. Ein ähnliches Optimierungsproblem stellt die Frage nach dem zu verwendenden Modell dar. Es kann günstig sein, gleichzeitig mehrere (primäre oder sekundäre) Modelle zu verwenden. Es ist aber eine Frage der verfügbaren Rechenzeit, ob man es sich leisten kann, nur eines abzuspeichern und ein Modell aus dem anderen online bei Bedarf immer wieder neu zu errechnen. Ob man alle abspeichern kann, ist wiederum vom verfügbaren (Haupt-) Speicherplatz abhängig.

Bei der Beschreibung des gegenständlichen Modellaufbaus eines Stückprozesses im Kapitel 4 finden sich hierzu ausführliche Anwendungsbeispiele.

Jedes Modell ist eine Nachbildung, so daß grundsätzlich die Genauigkeit von Modellen diskutiert werden muß.

Die in der hier vorgenommenen Modellbildung dominierende, drastische Vereinfachung liegt vor allem in der Einrichtung der "Station". Die Vorgänge dort - insbesondere in Stationen mit Verarbeitungscharakter - können sehr verwickelt und umfangreich sein. Es ist aber nach Ansicht des Autors dieser Arbeit Aufgabe von lokalen Einrichtungen, solche Vorgänge zu bewältigen, wie etwa ein Loch in ein Werkstück zu bohren, Stücke zu verschweißen, zu trennen etc. Solche Aufgaben können regional und völlig autonom bewältigt werden.

Die nach Auffassung des Autors überregionale Aufgabe in einem Stückprozeß, die Ver- und Entsorgung mit Stücken - etwa mittels eines Rechners - zu bewerkstelligen, rechtfertigt diese Vereinfachungen. Eine Diskussion der regionalen und überregionalen Aufgaben findet im Folgenden statt.

3. Grundsätzliche Anforderungen an einen Rechner im Stückprozeß aufgrund möglicher Verfahren zur Stück-Verfolgung und -Lenkung

Das Kennzeichen der Stückgutprozesse, daß einzelne Stücke ihre räumliche Position ändern, führt unmittelbar auf die Problemstellungen in diesem Kapitel :

- Wie kann die "räumliche Position" und
- wie kann die "Änderung"

erfaßt und beschrieben werden. Die grundsätzlichen Möglichkeiten zur Stücklenkung behandelt ein weiterer Abschnitt. Beides, Stückverfolgung und -Lenkung bestimmen maßgeblich die Aufgabenstellungen im Stückprozeß.

Darausfolgend lassen sich die Anforderungen ableiten, die allgemein an einen (steuernden) Rechner entstehen.

3.1 Möglichkeiten der Lokalisierung von Stücken

3.1.1 Ortsbestimmung : Die makroskopische und mikroskopische Beschreibung

Ein Stück befindet sich zu einem bestimmten Zeitpunkt an einem bestimmten Ort. Legt man ein - beliebiges - Koordinatensystem mit einem - willkürlichen - Bezugs-Nullpunkt, dem Koordinaten-Ursprung im Raum fest, so läßt sich die momentane Position eines Stückes relativ zu diesem Ursprung in Zahlen - den Koordinaten - beschreiben. Ist die Anordnung eben, so genügten 2 Koordinaten.

Näher betrachtet wird klar, daß, weil sich mit dieser makroskopischen Beschreibung mit drei bzw. zwei Zahlen natürlich nur die Lage eines Punktes im Raum bzw. in der Ebene angeben läßt, Stücke jedoch eine räumliche Ausdehnung haben, die Lagebeschreibung durch Koordinaten-Paare oder Tripel nicht hinreichend ist. Vielmehr müßte eine möglicherweise umfangreiche Detaillierung beigefügt werden, etwa in Form von Hüllflächenbeschreibungen, deren Lage im Bezug zum Koordinaten-System o.ä. Diese detaillierte Positionsbeschreibung werde als mikroskopische Beschreibung bezeichnet.

Wenn man die Identitätskennung eines Stückes als Abkürzung für die diversen, invarianten Einzelbeschreibungen auch geometrischer Natur nutzt, benötigt man die mikroskopische Positionsbeschreibung nicht mehr. Vielmehr kann man dann mit Hilfe der Punktdarstellung in Koordinatenform unter Angabe der Identitätskennung abkürzend den Raum angeben, den das Stück momentan einnimmt (die räumliche Ausdehnung eines Stücks mit einer bestimmten Identität ist ja unveränderlich, vergl. 2.1.2).

Der durch die Koordinaten beschriebene Punkt ist dabei ein fester Referenzpunkt auf oder in dem betreffenden Stück ; seine relative Lage auf oder in dem Stück ist als unveränderliches Datum ebenfalls Teil der Stückbeschreibung, d.h. anhand der Identität zu erfahren.

Ist jedoch die Lage eines Stückes im Sinne von "Neigung" - z.B. gegen bestimmte Bezugsflächen - auf bestimmten Wegen unterschiedlich und dies auch von Bedeutung, so muß zusätzlich zu den Ortskoordinaten noch ein Maß hierfür angegeben werden. (Ein solches Maß ist von der Zeit abhängig, kann also nicht in der Stückbeschreibung und damit nicht in der Identitätskennung enthalten sein. Es gehört also eindeutig zu den veränderlichen Größen im Stückprozeß und wird in diesem Zusammenhang zur Positionsbeschreibung gezählt.)

Ist die Lage eines Stückes ohne Bedeutung oder in seiner Lebenszeit invariant, so ist die Ortsbeschreibung durch Koordinaten - unter implizitem Einschluß der Beschreibung anhand der Stückidentität - ausreichend ; es wird abkürzend ein bestimmter Raum beschrieben.

Nun gibt es aufgrund der physikalischen Gegebenheiten in der geschlossenen Stückgutanordnung i.a. Räume, in denen sich zu keinem Zeitpunkt Stücke befinden, und solche, die Stücke beinhalten können. Letztere entstehen durch den Transport von Stücken längs Leitflächen.

Die Beschreibung[+)] aller Positionen aller Stücke zu einem bestimmten Zeitpunkt könnte man auch als eine Unterteilung des gesamten für den Material-

---

[+)] Die makroskopische Beschreibung der Position eines Körpers durch Koordinaten-Tripel oder -Paare betrachtet man oft dann als völlig hinreichend, wenn die Ausdehnung des Körpers im Vergleich zur Distanz vom Koordinatenursprung "vernachlässigbar klein" ist. Wenn die räumliche Lage bei der Kraftübertragung zum Bewegungsvorgang eine Rolle spielt, genügt diese Beschreibung nicht.

fluß in Form von Stücken zulässigen Raumes in bestimmte Räume begreifen, die momentan von bestimmten Stücken "besetzt" sind und in einen oder mehrere Resträume, die "frei" von Stücken sind. Damit ist der erste Schritt zu einer Abschnittsbildung getan, auch wenn die "Abschnitte" noch kontinuierlich - mit der Stückbewegung - wandern.

3.1.2 Anmerkungen zur Ortsveränderung von Stücken

Ortsveränderung bedeutet trivialerweise zunächst einmal Bewegung, die aus dem auf einen Körper einwirkenden Kräftespiel resultieren. Es ist nicht Gegenstand dieser Arbeit, die Einrichtungen zu untersuchen, die zur Kräfteübertragung benötigt werden, die also der technischen Realisierung des Bewegungsablaufs, des Beförderns dienen. Vielmehr steht der Bewegungsvorgang, d.h., die Bewegung im Vordergrund des Interesses.

Stückbewegungen lassen sich mit Hilfe der elementaren physikalischen Grundgesetze der Mechanik mit kontinuierlichen Größen wie "Weg", "Geschwindigkeit", "Beschleunigung" etc. entweder als Funktion der Zeit oder als Funktion einer jeweils anderen Größe beschreiben. Wie sich ein Stück bewegen wird, das einen bestimmten, bekannten Kräftespiel ausgesetzt ist, kann daher exakt vorausgesagt werden.

Tatsächlich finden aber unvorhersehbare Beeinträchtigungen der Kräfte statt, so daß der Bewegungsablauf einen mehr zufälligen Charakter erhält. Ein typisches Beispiel hierfür ist etwa der motorisierte Straßenverkehr /3.1/, wo neben Straßen- und Witterungsbedingungen vor allem zufällige, menschliche Verhaltensweisen den Verkehr maßgeblich beeinflussen.

Um also den Ortsveränderungen von Stücken auf die Spur zu kommen, gibt es letztlich nur zwei Möglichkeiten :

A) Absolutes Ausschalten von störenden Umwelteinflüssen.
Kennt man alle einwirkenden Kräfte exakt, dann ist der Bewegungsablauf absolut determiniert und es kann anhand von Bewegungsgleichungen allein über die Kenntnis des Zeitpunktes auf die räumliche Position eines Stückes geschlossen werden.

B) Einsatz von Meßfühlern

Kann man die einwirkenden Kräfte nicht zu jedem Zeitpunkt genau bestimmen, so muß man sich damit begnügen, die tatsächliche Bewegung mittels Messung nachzuvollziehen (im Gegensatz zu A , wo ja eine exakte Voraussage möglich war).

Absolutes Ausschalten störender Umwelteinflüsse ist illusorisch. Es muß ja damit gerechnet werden, daß gewisse Anlageteile eines technischen Prozesses mit gewissen Streuungen oder Toleranzen arbeiten, mit einer bestimmten Wahrscheinlichkeit nicht mehr ordnungsgemäß funktionieren oder gar ausfallen. Das heißt dann aber doch, daß die Voraussagen über die Bewegungsvorgänge "streuen" bzw. eben mit dieser Wahrscheinlichkeit möglicherweise nicht zutreffen ; die Vorgänge haben so auf jeden Fall wieder Merkmale des Zufälligen (wenn vielleicht auch in reduziertem Maße).

Man benötigt so auf jeden Fall Meßfühler im Prozeß, die Aufschluß über die Ortsveränderung von Stücken geben. (Daß Meßfühler natürlich ebenfalls Fehlerquellen darstellen, sei hier nicht weiter erörtert.)

3.1.3 Stückverfolgung im Kontinuierlichen

Es stellt sich die Frage, welche Arten von Meßfühlern in einem zu automatisierenden Stückprozeß benötigt werden, um die Ortsveränderung von Stücken mittels eines Rechners nachzuvollziehen. Wie stets in dieser Arbeit, steht auch hier nicht eine bestimmte technische Realisierung, sondern die Art und Weise der Messung im Vordergrund.

Die Größen, mit der Bewegungen beschrieben werden können - etwa Weg, Geschwindigkeit - sind Größen mit grundsätzlich kontinuierlichem, stetigem oder stückweise stetigem Wertebereich. D.h., die Bewegung von Stücken muß eindeutig der Klasse der Fließ- oder Dynamischen Prozesse zugeordnet werden.

Man könnte also versuchen, z.B. den Weg eines Stückes mit analogen Meßfühlern zu verfolgen und einem Rechner - in digitalisierter Form - zur Verfügung zu stellen. Dabei treten aber eine Reihe von Problemen auf.

Wie bereits geschildert, ist natürlich dabei zu beachten, daß eine einfache Wegmessung sich in der Regel nur auf einen (Referenz-) Punkt der Umhüllenden des Stückes bezieht, d.h., eine Wegmessung allein ist zunächst wegen der räumlichen Ausdehnung des Stücks noch nicht hinreichend (s.o.).

Ein weiteres Problem ist das Meßverfahren. Berührende Verfahren, also Verfahren mit ständiger mechanischer, körperlichen Verbindungen von Stück und analogen Meßwertgebern (Wegaufnehmer) scheiden aus der Erwägung heraus aus, daß eine freie Beweglichkeit der Stücke längs beliebiger Förderwege damit nicht mehr gegeben ist.

Berührungslose Meßverfahren zur Positionsbestimmung arbeiten in der Regel mit Reflexion (Licht, Schall, elektromagnetische Wellen) oder Beeinflußung von Feldern. Sie liefern ebenfalls analoge Werte (oder Werte, die so fein quantisiert sind, daß sie quasi als analog betrachtet werden können). Bei diesem Verfahren bleibt die freie Beweglichkeit der Stücke absolut erhalten.

Es stellt sich jedoch die Frage, was man mit der Kenntnis der analogen Position eines Körpers anfängt, ob diese Genauigkeit benötigt wird.

Will man den Zustand eines Stückes im Sinne einer Bearbeitung verändern, so benötigt man natürlich möglichst genaue Maßzahlen, um entsprechende steuernde Eingriffe in den Bearbeitungsprozeß vornehmen zu können. Man benötigt dann auch zusätzliche Maßzahlen, die über die (momentane) räumliche Ausdehnung von Körpern Auskunft geben ; es ist der Fall, in der die mikroskopische Lokalisierung benötigt wird.

Dasselbe gilt etwa für bestimmte Steuerungsprobleme, etwa die optimale Lenkung beim Einschleusvorgang /3.2/ oder die Steuerung von Regalförderzeugen in Hochregalanlagen /3.3/, in denen der Zustand eines Stückes nicht verändert wird.

Als eng begrenzte Aufgabenstellungen können alle Aktionen lokal und autonom abgewickelt werden. Es ist funktionell überflüssig, daß außerhalb dieses Raumes, in dem diese Aktionen abgewickelt werden, eine andere Stelle Kenntnis über die mikroskopische Position von Stücken in dieser Station erhält. Eine übergeordnete Zentralstelle wird allenfalls dazu benötigt, um solche Stationen mit Stücken zu versorgen oder zu entsorgen, d.h., um die gegenständlichen Voraussetzungen zur Abwicklung der lokalen Aktionen zu schaffen.

Für manche lokalen Aufgabenstellungen in Stückprozessen ist zwar eine analoge, in gewissem Rahmen möglichst genaue Positionsbestimmung unerläßlich.

Weil aber eine übergeordnete Zentralstelle davon keine Kenntnis benötigt und aus Gründen des Aufwandes greift man zum im folgenden Abschnitt dargelegten Verfahren.

3.1.4 Stückverfolgung mit binären Einrichtungen : Abschnittsbildung und Bearbeitungsverfahren

Wie dargelegt, kann die Frage "wo befindet sich ein bestimmtes Stück momentan" exakt im Grunde nur mit "es belegt gerade den und den Raum" - mit allen nötigen Details - beantwortet werden.

So liegt es nahe, sich die Positionsverfolgung dadurch zu vereinfachen, daß man von vornherein den für den Materialfluß zulässigen Raum in Teile oder Abschnitte zerlegt, um obige Frage einfach mit der entsprechenden Abschnittskennung beantworten zu können.

Die veränderte Fragestellung "Befindet sich ein Stück im Abschnitt i (eigentlich : Raum i) ?" kann gar mit einem simplen "Ja" oder "Nein" beantwortet werden. Damit wird die Stückverfolgung wesentlich vereinfacht, insbesondere durch die für Digitalrechner ideal geeignete Antwortform. Die Meßfühler oder Detektoren sind als Digital-Einrichtungen sehr viel einfacher zu realisieren als analoge Wegemeßgeräte.

Zwei Arten von (Binär-) Signalen sind bei der abschnittsweisen Stückverfolgung ableitbar :

A) Statische Signale : Sie geben direkt den Zustand wieder, solange der Zustand anhält, z.B. : "Abschnitt belegt".

B) Dynamische Signale : Sie geben die Änderung des Zustandes wieder, unabhängig davon, wie lange der Zustand anhält, z.B. : "Betreten eines Abschnittes".

Je nachdem ob statische oder dynamische Signale verwendet werden, ist die Bearbeitungsstrategie unterschiedlich, obwohl beide untrennbar miteinander verknüpft sind : Ohne "Zustand" keine "Änderung" und umgekehrt.

1) Bei statischen Signalen - wie übrigens auch bei analoger Wegemessung - muß ein Rechner i. a. von sich aus selbsttätig Aktionen[+)] unternehmen, sozusagen einfach auf Verdacht hin, um die Signale einzulesen. Nur durch Vergleich mit dem Zustand zum Zeitpunkt der letzten Einleseoperation ist eine Veränderung zu erkennen.

2) Dynamische Signale kann man unmittelbar an die Unterbrechungseingänge eines Prozeßrechners anschließen, um möglichst sofort Aktionen des Rechners bei Zustandsänderungen des technischen Prozesses auszulösen.

Das Verfahren 1) werde künftig als "zyklisches Verfahren", das Verfahren 2) als "spontanes Verfahren" bezeichnet.

Natürlich sind für die gewonnene Vereinfachung auch Nachteile in Kauf zu nehmen.

Zwar ist die Lokalisierbarkeit von Stücken in Bezug auf die Genauigkeit - mit voller Absicht natürlich - herabgesetzt und auf eine endliche Zahl von Abschnitten begrenzt. Je nach Anordnung der Detektoren kann es aber sein, daß in dem Zeitintervall, in dem ein Stück den Abschnitt i verläßt, den unmittelbar nachfolgenden Abschnitt j aber schon betreten hat, mehrdeutige Signale entstehen. Z. B. : Stück befindet sich (nach Angaben der Meßfühler) gleichzeitig im Abschnitt i und im Abschnitt j. Oder : Stück befindet sich momentan (nach Angaben der Meßfühler) weder im Abschnitt i noch in Abschnitt j. Es sind auch Kombinationen denkbar, die anstatt auf eine Bewegung von A nach B auf gerade die umgekehrte Bewegung von B nach A schließen lassen.

Derlei Fälle lassen sich softwaremäßig beherrschen, wenn man sich die Vorgeschichte merkt und Zustandsänderungen registriert, so daß falsche Rückschlüsse vermieden werden können. Die Einfachheit des Verfahrens muß mit erhöhtem algorithmischem Aufwand bezahlt werden.

+) Entweder in einer Schleife oder in einem von einer Uhr initiierten, i. a. zyklisch gestarteten Programm

3.2 Ein Modell zur Lösung der abschnittsweisen, individuellen Stückgutverfolgung

Die erste zu lösende Aufgabe im zu automatisierenden Stückprozeß die sich bei der Ver- und Entsorgung von Stationen stellt, ist die individuelle Verfolgung der Stücke. Die grundsätzlichen Möglichkeiten wurden in vergangenen Abschnitten zusammengestellt. Gründe, warum hierzu eine abschnittsweise Verfolgung ausreichend ist, wurden dargelegt.

Nun darf nicht übersehen werden, daß die bloße Feststellung der Anwesenheit eines Stückes noch nicht die Aufgabe löst, welches individuelle Stück sich gerade in einem Beobachtungsabschnitt befindet. Die Gesamtaufgabe bei der Stückverfolgung lautet :

Wo befindet sich welches Stück wann ?

Der folgende Modellansatz dient der Loslösung von den manigfaltigsten denkbaren technischen Realisierungsmöglichkeiten, die vom Endschalter über den Leseautomaten bis hin zum Menschen reichen, der etwa die Teilenummer eines Stückes eintastet. Zudem beinhaltet er, daß vor Ort, also lokal und dezentral gewisse Tätigkeiten auszuführen sind.

3.2.1 Der Beobachter

Gegeben sei eine abgeschlossene Stückgutanordnung, die nach den in dieser Arbeit entwickelten Modellvorstellungen z. B. aus

- N Knoten oder Stationen und
- M' elementaren, gerichteten Transportstrecken besteht (Vergleiche hierzu 2.5)

Zur Erfassung der Stückbewegungen wird an jeder der Strecken und nach Bedarf an bestimmten Stationen je ein Beobachter aufgestellt. Sein Beobachtungsbereich erstreckt sich auf die ganze Strecke bzw. den ganzen Knoten, also den gesamten Abschnitt.

Ist z. B. eine Strecke zu groß, d. h. ist genauere Lokalisierung nötig, so wird durch rein formales Einfügen von Hilfsknoten und entsprechend vielen realen, zusätzlichen Beobachtern eine Unterteilung der betreffenden Strecke vorgenommen. Sinngemäßes gilt für Stationen. Formal können Stationen frei von Beobachtern gehalten werden mittels Substitution der Knoten durch Strecken

mit Beobachtern und mehreren beobachterfreien Knoten. Umgekehrt kann man erreichen, daß Beobachter nur an Knoten stationiert sind, wenn man das Stationen-Modell umformt (siehe 2.5). Weil alle Modifikationen letzlich dieselbe Aussagekraft besitzen, wird hier die anschaulichere Darstellung in Form des Stationen-Modells bevorzugt.

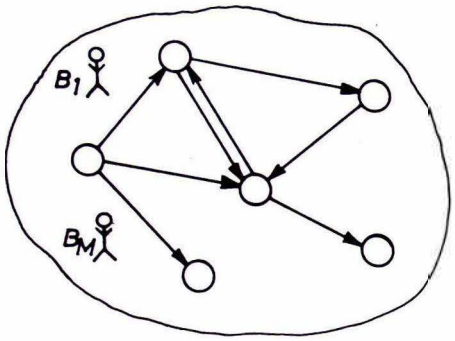


Bild 3.1 : Einsatz von "Beobachtern" anhand des Stationenmodells

Insgesamt gebe es M Beobachter. Die Aufgabenstellung eines jeden sei wie folgt : Erscheint im Beobachtungsbereich b_m des Beobachters B_m ($m=1,2,3...M$) ein Stück, so registriert er das Ereignis : "Ein Stück ist im Bereich b_m zum Zeitpunkt t_k angekommen".

Wie lange das Stück in seinem Beobachtungsbereich bleibt, sei zunächst unerheblich. Nach Notieren des Ereignisses ist seine Tätigkeit für das ereignisauslösende Stück beendet. So hat der Beobachter B_m im Intervall $[0, t_k]$ die Ereignissequenzen

$$\begin{array}{c}(t_1,\ b_m)\\(t_2,\ b_m)\\\vdots\\(t_k,\ b_m)\end{array}$$

erfaßt und notiert, so daß sich vereinfacht diese Folge von Ereignissen als Folge von Zeitpunkten

$$(3.1) \qquad B_m = B_m(t) = \{t_1,\ t_2,\ \ldots\ t_k\}$$

beschreiben läßt.

3.2.2 Der identifizierende Beobachter

Für den anonymen Stückprozeß ist die Erfassung des Prozeßzustandes mit solchen Folgen hinreichend, um die gewünschten Ergebnisse wie etwa Stückzahl, Stückdichte etc. zu ermitteln, da allein aus der Folge von Zeitpunkten auf die Folge von anonymen Stücken geschlossen werden kann.

Dagegen ist im Stückprozeß mit einzelnen identifizierbaren Stücken durch bloße Ereignisfolgen der Prozeßzustand nicht ausreichend zu erfassen, d.h., der Prozeßzustand läßt sich nicht hinreichend ereignisabhängig bestimmen.

Daher wird nun die Aufgabenstellung des Beobachters um die Identifizierung erweitert, so daß er bei Erscheinen eines Stücks in seinem Beobachtungsbereich b_m das Ereignis registriert :

"Das Stück mit der Identität i_k ist im Bereich b_m zum Zeitpunkt t_k angekommen".

Diese Art von Beobachter wird im Folgenden als "identifizierender Beobachter" bezeichnet, im Gegensatz zum seitherigen "nichtidentifizierenden Beobachter".

Der identifizierende Beobachter B_m hat dann im Zeitraum $[0, t_k]$ die Beobachtungen

$$\begin{array}{c} (t_1, b_m, i_1) \\ (t_2, b_m, i_2) \\ \vdots \\ (t_k, b_m, i_k) \end{array}$$

notiert.

Wenn alle Stücke eindeutig identifizierbar sind, unterscheidet sich im allgemeinen jede Beobachtung von der anderen außer im Zeitpunkt zusätzlich noch durch die Identität.

Diese Folge von beobachteten Ereignissen läßt sich als Folge

$$(3.2) \qquad B_m = B_m(t, I) = \left\{(t_1, i_1), (t_2, i_2), \ldots (t_k, i_k)\right\}$$

beschreiben.

Verwendet man alle M Beobachter-Sequenzen, so kann mit ihrer Hilfe im Idealfall (vgl. 7.3) zu jedem Zeitpunkt der Prozeßzustand hinreichend genau bestimmt werden.

Es ist einleuchtend, daß eine bezüglich der Zeitpunkte und des Belegungsabbildes genauere Lokalisierung erreicht werden kann, wenn man etwa die Aufgabenstellung der Beobachter verfeinert, etwa um die Meldung "Stück hat Bereich verlassen" oder "Stück befindet sich im Bereich" erweitert. Weil sich durch diese Verfeinerung prinzipiell keine Änderung an der Forderung nach Sammlung der Meldungen ergibt, werde künftig immer nur das "Ankommen" behandelt.

3.2.3 Ein Modell zur Erfassung des Prozeßzustandes : Das M-Beobachter-Modell

Die Aufgabe der Beobachter ist es, "vor Ort", also im technischen Prozeß die Vorgänge in bestimmten Teilräumen oder Abschnitten zu erfassen. Wenn ein Stück einen solchen Abschnitt betritt, wird es irgendwann im Verlauf des Durchschreitens registriert oder beobachtet, möglicherweise auch identifiziert.

Diese Tätigkeiten sind lokale Aufgaben. Jeder der M Beobachter ist zunächst einmal unabhängig vom anderen, jeder von ihnen weiß, ob sich ein Stück - evtl. sogar welches - in seinem Abschnitt, in seinem Beobachtungsbereich befindet.

Um eine Übersicht über das Geschehen in der gesamten Stückgutanordnung zu erhalten, muß jede lokale Nachricht in einer Zentrale gesammelt werden ; d.h., die Beobachtertätigkeit ist um die der Abgabe der Beobachtungen zu erweitern.

Losgelöst von allen Details der Hardware, der Geometrie etc. läßt sich der zur Stückverfolgung nötige Informationsfluß im Stückprozeß als Graph (Bild 3.2) darstellen. Ähnlich wie beim Stationenmodell des Stückprozesses selbst, beinhalten die Ecken des Graphen Quellen und Senken. In den Beobachter-Stationen entstehen eben hier Nachrichten, die Zentrale ist die einzige Senke. Die Kanten dienen ebenfalls dem Transport.

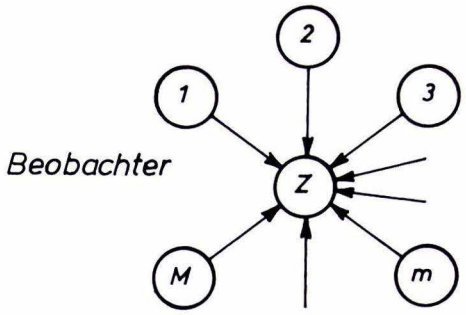


Bild 3.2 : Ein Graphen-Modell zur Erfassung des Prozeßzustandes in Stückprozessen : Das M-Beobachter-Modell mit M Beobachtern und einer Zentrale (Z)

Dieses sehr schlichte Modell führt bei näherer Betrachtung auf die wichtigste Problemstellung bei der Automatisierung von Stückprozessen : Nämlich die Beherrschung vieler, bis zu M paralleler, evtl. gleichzeitig anfallender Stückbewegungen und damit von M Beoachtungen.

Diese Problemstellung - das "M-Beobachter-Problem" -, ursächlich aus der Aufgabe der Stückgutverfolgung stammend, wird als ein zentrales Thema in den Kapiteln 6 und 7 dieser Arbeit behandelt.

3.3 Grundlegende Methoden zur Zielsteuerung von Stücken

Bei der Lösung der Aufgabe "Bringe Stück s_i vom Ort A zum Ort B" gibt es m.E. zwei grundlegende Varianten :

A) Das Ziel kann fest oder veränderlich sein.
Veränderlich heißt, daß die Zielbestimmung für ein Stück nach der ersten oder Ur-Aufgabenstellung nachträglich geändert wird : Für das Stück s_i lag zunächst ein Transport-Auftrag vom Ort A_k zum Ort B_l vor ($k = 1, 2, \ldots K$, $l = 1, 2, \ldots L$; K = Zahl der möglichen Ursprungsorte, L = Zahl der möglichen Zielorte) und der Zielort B_l wird nun verändert in $B_{l'}$.

Hat das Stück den Ort A_k noch nicht verlassen oder ist das Stück schon am Ort B_l angelangt, gibt es weiter kein Problem : Entweder kommt die neue Order dann noch rechtzeitig oder es ergibt sich einfach ein zweiter Transport eben von B_l nach $B_{l'}$.

Geschieht die Veränderung des Ziels jedoch unterwegs, wenn das Stück also bereits den Ursprungsort verlassen, den Zielort jedoch noch nicht erreicht hat, so ist die Lenkung zum neuen Ort nur möglich, wenn die Zieladresse das Stück nicht (ausschließlich) physikalisch als eine Art Etikett begleitet. Das Stück kann in diesem Fall nicht über lokale Ableseeinrichtungen mit angeschlossener, dezentraler Zielsteuerungshardware zum Ziel gelenkt werden.

Ist das Ziel fest, d.h. invariant während der Lebenszeit eines Transportauftrages, so kann jedoch eine entsprechende Einrichtung die Zielsteuerung dezentral und autonom übernehmen, so daß für diese Aufgabe eine (intelligente) Zentrale - ein Rechner - nicht benötigt würde.

B) Der zu wählende Weg kann fest oder veränderlich sein.
Fester, d.h., invarianter Weg bedeutet, daß für ein Stück s_i vom dem Ort A_k zum Ort B_l für jedes Paar (A_k, B_l), für das eine Verbindung - eine Kantenfolge im Graphen - existiert, immer der gleiche Weg - die gleiche Kantenfolge - eingeschlagen wird, unabhängig von der aktuellen Belegung der Stückgutanordnung[+].

[+] Der Fall, daß für ein Paar (A_k, B_l) für das Stück s_j möglicherweise ein anderer Weg als für das Stück s_i eingeschlagen wird, ändert nichts an dieser Betrachtungsweise.

Der einzuschlagende Weg ist veränderlich, wenn es von den Ursprungsorten A_k zu den Zielorten B_l erstens mehrere Kantenfolgen gibt und so zweitens abhängig vom Belegungszustand verschiedene Wege eingeschlagen werden können. Dazu benötigt man Wissen über den gesamten Prozeß, d.h., eine Zentrale, die den aktuellen Belegungszustand erfaßt, notiert und (per Optimierungsalgorithmen) entsprechende Maßnahmen zur Lenkung der Stücke trifft.

Beide Varianten der Zielsteuerung sind teilweise kombinierbar (Bild 3.3)

		Weg	
		fest	variabel
Ziel	fest	X	X
	variabel	–	X

Bild 3.3 : Kombinationen der Varianten zur Zielsteuerung

Insgesamt ergeben sich so die folgenden Möglichkeiten der Zielsteuerung :

Zielsteuerungsvefahren		Merkmale
Ziel	Weg	
1) fest	fest	- Zieladresse kann Stück physikalisch begleiten - Zielsteuerung dezentral und lokal, nicht adaptiv - keine Beeinflussungsmöglichkeiten während des Förderablaufs - keine Fehlerkorrektur (siehe 3.4.3)
2) fest	variabel	- Zieladresse kann Stück physikalisch begleiten - Zielsteuerung aufgrund übergeordneter Notierung der Belegung zentral und adaptiv bezüglich des Weges - Einflußmöglichkeiten auf den Weg während des Transportes, evtl. Fehlerkorrektur nach Falschfahrt

3) variabel	variabel	- Physikalisches Begleiten der Zieladresse nicht sinnvoll - Zielsteuerung zentral und adaptiv bezüglich des Weges - Alle Einflußmöglichkeiten auf Ziel und Weg während des Förderablaufs bis hin zur evtl. Fehlerkorrektur nach Falschfahrt
4) variabel	fest	nicht sinnvoll, es sei denn, man betrachtet dieses Verfahren als Teilmenge von 1), wobei die Zieladresse das Objekt i. a. nicht begleiten kann.

3.4 Zusammenfassung

Nach Zusammenstellung der möglichen Verfahren zur Stückgut-Verfolgung und Lenkung lassen sich die globalen Anforderungen bzw. Aufgabenstellungen in Stückprozessen zusammenfassen, die beim Einsatz von Rechnern als steuernde Instanz entstehen.

3.4.1 Mikroskopische Aufgabenstellungen

Bei der Verfolgung, der Lenkung und insbesondere bei der Bearbeitung von Stücken können sich auch die Aufgabenstellungen der Fließ- oder Dynamischen Prozesse ergeben :

Als von der steuernden Instanz - dem Rechner - zu erfassende Größen treten im Zusammenhang mit der Behandlung von Stücken eben die Größen mit kontinuierlichem Wertebereichen auf, die man zur analytischen Beschreibung bewegter Körper (bzw. Massenpunkte) benötigt.

Auch die auszugebenden Stellgrößen, die der detaillierten Lenkung der Bewegung, der Steuerung des Bearbeitungsvorgangs etc. dienen, können der genauen Zustandserfassung, d.h. Messung wegen kontinuierliche, analoge Größen sein (die bei Digitalrechnern unumgängliche Quantifizierung sei dabei vernachläßigt).

Diese Aufgabenstellungen stammen zwar aus Stückprozessen, da sie sich aber in nichts von den Aufgabenstellungen in beliebigen Fließprozessen unterscheiden, kann man sie nicht als die typischen, aus Stückprozessen resultierenden Aufgabenstellungen bezeichnen. Es sind lokale, mikroskopische Aufgaben, die vollkommen autonom abgewickelt werden können.

3.4.2 Makroskopische Aufgabenstellungen

Als übergeordnete, makroskopische Aufgabenstellung werden hier die Aufgaben näher betrachtet, die bewältigt werden müssen, um die Ver- und Entsorgung von Stationen mit Stücken zu gewährleisten. Sie hängen ausschlaggebend vom Verfahren zur Stückgutlenkung ab.

Das Verfahren 1) mit festem Ziel und festem Weg benötigt man zunächst überhaupt keine Zentralstelle, wenn es nur um den Vorgang der Wegelenkung geht. Die Lenkung der Stücke erfolgt lokal mit lokalen, autonomen Einrich-

tungen. Der Einsatz eines Zentral-Rechners, der nur die Belegung aufgrund der Meldungen der Beobachter notiert, kann dann z.B. Vorteile erbringen, wenn durch Erfassen des Istzustandes Hilfestellungen bei Fehlerkennung und Fehlerdiagnose benötigt werden.

Die Aufgaben des Rechners umfassen beim Verfahren 1) höchstens

A) Erfassen des momentanen Belegungszustandes durch Sammlung der lokalen Beobachtungen

B) Notieren des momentanen Zustandes in Prozeßmodellen.

C) Ausgabe von Zustandsmeldungen an das Wartungspersonal, jedoch kein Eingriff in den Prozeß.

Das Verfahren 1) erfordert also unter diesem Gesichtspunkt höchstens die offene Prozeßkopplung, d.h., keine direkte Einwirkung auf das Prozeßgeschehen bei der Änderung der räumlichen Position von Stücken. Der durchaus mögliche Gewinn durch das Wissen über den Prozeßzustand wird so nicht vollständig genutzt [+].

Das Verfahren 2) mit festem Ziel, aber variablem Weg erfordert zwingend eine Zentralstelle. Nur wenn man an einer Stelle über alle Informationen - alle Beobachtungen- verfügt, läßt sich die Wegeführung an den momentanen oder einen daraus schon ableitbaren, künftigen, vorhersehbaren Zustand anpassen. Die Anforderungen an den zentralen Rechner sind beim Verfahren 2) :

A) Erfassen des momentanen Belegungszustandes (s.o.)

B) Notieren des momentanen Zustandes (s.o.)

C) Ausgabe von indirekten Stellbefehlen in Form vom Vorschriften (mittelbarer Prozeßeingriff) oder Ausgabe von direkten Stellbefehlen (unmittelbarer Prozeßeingriff) aufgrund des Prozeßzustandes.

Die Betriebsweise des Prozeßrechners muß geschlossen prozeßgekoppelt sein, um den Eingriff in das Prozeßgeschehen zu erlauben. Dabei gibt es bei der Zielsteuerung zwei Arten von Eingriffen :

[+] Ein bekanntes Beispiel ist etwa der Einsatz von Rechnern in Kernkraftwerken, wo der direkte Prozeßeingriff mittels Rechnern aus Sicherheitsgründen unterbleibt.

Begleiten die Zieladressen die Stücke physikalisch, so kann die Lenkung mittelbar, d.h., indirekt und lokal bewältigt werden. D.h., daß nach Übertragung der Vorschriften, wie mit welchen Zieladressen zu verfahren ist - zu prinzipiell beliebigen Zeitpunkten im Einbahnverkehr von der Zentrale zur lokalen Zielsteuerung -, alle Stücke autonom gelenkt werden können. Und dies so lange, bis wieder neue Vorschriften erlassen werden.

Begleiten die Zieladressen die Stücke nur virtuell im Prozeßmodell, so muß nach dem Betreten der Verteilstation, aber vor dem Verteilvorgang jeweils in der Zentrale nachgefragt werden, wie das Ziel lautet. Gleichgültig, ob die Station mit dem Ziel antwortet, das dann lokal, aber unmittelbar in entsprechende Vorgänge umgesetzt wird oder ob die Zentrale direkt einen Stellbefehl ausgibt, stellt sich durch dieses Frage- und Antwortspiel die Forderung nach dem unmittelbaren, möglichst rechtzeitigen Eingriff in das Prozeßgeschehen ein. Dies führt zur Verschärfung der Echtzeitanforderungen an einen Rechner.

Das Verfahren 3) mit variablem Ziel und variablem Weg ergibt dieselben Anforderungen wie das Verfahren 2), nur daß wegen der grundsätzlich virtuellen Begleitung der Zieladresse stets der unmittelbare Prozeßeingriff nötig ist.

Die Forderung nach dem unmittelbaren, rechtzeitigen Prozeßeingriff stellt sich grundsätzlich in Förderstrecken bei der Steuerung der Stückbewegungen. So muß z.B. nach Betreten eines bestimmten Förderabschnittes ein das betretende Stück betreffender Stellbefehl (z.B. ein Halt-Befehl, weil der vorausliegende Abschnitt besetzt ist) gegeben werden noch bevor das Stück den Abschnitt verlassen hat. Ähnliches gilt für Stationen, aus denen Stücke auch nicht unkontrolliert austreten dürfen.

(Eine ausführliche Betrachtung und Untersuchung dieser sehr wichtigen Problemstellungen findet sich in Kapitel 7 dieser Arbeit.)

3.4.3 Anmerkungen zur Fehlererkennung, Fehlerlokalisierung und Fehlerkorrektur in Stückprozessen

Die bessere Fehlererkennung und Fehlerlokalisierung, d.h., die Verbesserung der Wartbarkeit zählen allgemein zu den Vorteilen, die man sich vom prozeßgekoppelten Betrieb eines Rechners versprechen kann. Sie sind nicht

allein typisch für die Klasse der Stückprozesse, wenn auch dort insofern von einer bestimmten Bedeutung, als die Fehlerlokalisierung in hochautomatisierten, räumlich weitverzweigten technischen Prozessen ein besonderes Problem darstellt.

Fehler - ob drohende oder bereits aufgetretene - erkennt man entweder durch zusätzliche, im Grunde für den Normalprozeß überflüssige Prozeßgrößen oder durch bestimmte Prüfungen der normalen Prozeßgrößen /3.4/, also grundsätzlich durch redundante Maßnahmen. Da dies in jedem Prozeßtyp so ist, wurden diese Anforderungen seither nicht genannt ; Sie werden hier nicht weiter untersucht, sondern nur der Vollzähligkeit halber angeführt.

In bestimmten Stückgutanordnungen, in denen der Weg variabel ist, kann eventuell sogar automatisch eine Fehlerkorrektur im Sinne einer nachträglichen Beseitigung der Auswirkung von Fehllenkungen vorgenommen werden, wenn die Topologie des Stückprozesses dies zuläßt.

Diese Möglichkeit ist aber nur ein bestimmter, evtl. sehr nützlicher Sonderfall der Stückgutlenkung mit variablen Wegen : Sie setzt die Fehlererkennung und -Lokalisierung voraus. Daraufhin einen neuen Weg zu berechnen, unterscheidet sich in nichts von dem algorithmischen Problem der Wegesuche, das bei jeder normalen, ungestörten Stückgutlenkung vom Ursprungsort A_k zum Zielort B_1 auftritt.

4. Ein gegenständliches Modell eines Stückprozesses

Das im Folgenden vorgestellte gegenständliche Modell eines Stückprozesses ist das Ergebnis von fünf Diplomarbeiten, neun Semesterarbeiten, der Zuarbeit von zahlreichen wissenschaftlichen Hilfskräften im Institut für Regelungstechnik und Prozeßautomatisierung unter meiner Anleitung und Mitarbeit während eines Zeitraumes von beinahe fünf Jahren.

Weil es den Rahmen dieser vorliegenden Arbeit bei weitem sprengen würde, wollte man auf alle Details eingehen, wird in den folgenden Abschnitten auf viele Einzelheiten verzichtet. Die Vorstellung des gegenständlichen Modells beschränkt sich daher im Wesentlichen auf die funktionellen Aspekte und einige Randbedingungen, die bei der Realisierung zu beachten waren.

Gleichzeitig wird die Beschreibung des Modellprozesses durch die Anwendung der in den vergangenen Kapiteln vorgeschlagenen Modelle ergänzt.

4.1 Zielsetzung

Begriffe : Prozeßmodell und Modellprozeß

In DIN 66201 ist ein Prozeßmodell definiert als "Die Beschreibung oder Nachbildung eines Prozesses aufgrund des Ergebnisses einer Prozeßerkennung" /2.10/. Dortige Definition umfaßt zwar auch die gegenständliche Darstellungsform. Um nicht ständig zwischen gegenständlich und nichtgegenständlich unterscheiden zu müssen und um Mißverständnisse zu vermeiden, wurde in /4.1/ der Begriff "Modellprozeß" geprägt. Es läßt sich wie folgt definieren :

> Ein Modellprozeß ist die gegenständliche Nachbildung eines technischen Prozesses.

Im folgenden wird nur die nicht-gegenständliche, abstrakte Nachbildung als Prozeßmodell bezeichnet.

Ziele des Modellprozeßaufbaus

Die hier vorgestellte Realisierung des Modellprozesses "Stückprozeß" dient der Nachbildung der wichtigsten operativen Aufgabenstellungen im geschlossen prozeßgekoppelten Betrieb eines automatisierten Stückprozesses.

Er soll nicht eine Lösung von Hardwareproblemen in Materialflußanordnungen ("Materialhandling") wiederspiegeln. Es steht also nicht eine bestimmte Art von Fördereinrichtungen oder eine spezielle Fördertechnik im Vordergrund, obwohl sich beim gegenständlichen Aufbau auch dieses Problem stellt.

Der Modellprozeß soll auch nicht die Lösung dispositiver Probleme im Sinne von Planung oder Entwurf (siehe Kap. 1.3) demonstrieren. Er ist auch kein Simulationsersatz (obwohl er hierzu im beschränkten Maße auch herangezogen werden könnte) ; zur Simulation eignen sich abstrakte Modelle in Rechnern der leichten Veränderbarkeit wegen i.a. besser.

Vielmehr stellt der realisierte Modellprozeß im eigentlichen Sinn des Begriffs "Modell" eine "logische", d.h. funktionelle Nachbildung dar, losgelöst von obigen Aspekten. Die Schwerpunkte bei seiner Entwicklung lagen denn auch auf folgenden Gesichtspunkten :

- Anschaulichkeit und Übersichtlichkeit :

  Reale Stückgutprozesse sind oft schwer oder gar nicht überschaubar wegen ihres Aufbaus oder ihrer großen räumlichen Ausdehnung bzw. Verteilung. Der Modellprozeßaufbau sollte daher alle Vorgänge auf begrenztem Raum im Labor sichbar machen. Trotz prinzipiell funktioneller Nachbildung bezüglich der Hardware sollte die Anschaulichkeit im Sinne von Ähnlichkeit mit realen Stückgutprozessen möglichst erhalten bleiben. D.h., die typischen Vorkommnisse im realen Prozeß müssen auch im Modellprozeß sichtbar sein.

- Erprobung und Demonstration der Prozeßprogrammiersprache PEARL :

  Bei der Demonstration von Softwareprodukten stellt sich grundsätzlich die Frage, wie die Wirkung des implementierten, aber, wie der Name ja bereits aussagt, nicht faßbaren Produktes [+)] demonstriert werden kann.

  Blinken von Rechner-Konsollampen ist nichtssagend. Sog. schöne Protokolle auszudrucken ist auf Stapel-Rechnern üblich, für einen Prozeß-

---

+) PEARL-Subset-1-Compiler der ASME /1.5/

rechner jedoch unbefriedigend. So lag es nahe, die Wirkung einer Prozeß-programmiersprache auch mit Hilfe eines realen, technischen Prozesses praktisch, für jedermann sichtbar zu machen.

Zudem kann an einem solchen zur Realität vergleichsweise kleinen Modell-prozeß untersucht werden, welche Spracheigenschaften eine Prozeßprogram-miersprache haben muß, um die operativen Aufgaben im automatisierten Stückprozeß lösbar zu gestalten. Überdies gestattet ein solcher Labor-Prozeß das gefahrlose und billige Experimentieren, was in einer industriellen Anlage nur beschränkt oder gar nicht möglich ist, so daß mögliche Fehler und Lücken im Sprachentwurf leichter erkannt werden können. Der Nachteil, daß per de-finitionem jedes Modell eine Nachbildung ist und so nicht unbedingt der rauhen Wirklichkeit entspricht, muß dabei allerdings in Kauf genommen werden.

Studium und Erprobung von abstrakten Prozeßmodellen

An einer gegenständlichen Realisierung läßt sich die Brauchbarkeit von Pro-zeßmodellen (z. B. nach Kapitel 2.4) studieren und erproben : Will man einen Stückprozeß "on-line closed-loop" automatisieren, benötigt man ohnehin eine Nachbildung im Prozeßrechner.

4.2 Der Aufbau eines Warenverteilsystems

4.2.1 Abriß der Problemstellungen beim Transport der Stücke

Als "Blutader" des Materialflusses stellt die Förderstrecke (eigentlich genauer : Förderraum) das ausschlaggebende Betriebsmittel dar, das zu bestimmten Zeiten - einfach oder mehrfach - belegt oder frei sein kann. Dem Begriff des "Beförderns" widersprechend, dient sie sehr oft in dem Fall, daß gewisse Hindernisse auftreten, einfach als Speicher für ein oder mehrere Stücke[+]. D.h., in einer solchen Anordnung ist mit Staus zu rechnen. (Mit welcher Wahrscheinlichkeit ein Stau von wie vielen Stücken auftritt, welches die mittlere Staulänge und darausfolgend die einzurichtende Zahl von Warteplätzen ist etc., ist Gegenstand statistischer Betrachtungsweisen in der Theorie der Wartesysteme, z.B. in /4.3, 4.4, 1.13/).

Bei der Realisierung des Modellprozesses stand nicht dieses Planungsproblem, sondern die mehr organisatorische Abwicklung des Transportvorganges im Vordergrund. Nämlich, ob man bidirektionale Förderstrecken zuläßt, wie die Bewegung eines Stückes (Fahren, Halten) möglichst einfach gesteuert werden kann, wie das Problem der Nachführung von Stücken vor Bedienstationen bzw. Konfliktstellen (Staus) und wie die Konfliktsituationen an Verteiler- und Sammelelementen technisch und organisatorisch bewältigt werden.

Besonderes Augenmerk lag darauf, daß alle relevanten Vorgänge per Hand durch einen Bediener steuerbar sein sollten. Dazu dürfen die Vorgänge nicht zu schnell ablaufen und die Anlage muß in einen definierten, stabilen Ruhezustand versetzbar sein, um den menschlichen Bediener nicht zu überfordern.

Ein weiterer Akzent der Entwicklung lag auf der autonomen, d.h., vom Prozeßrechner unabhängigen Funktionsfähigkeit der Anlage. Der Aspekt der Fehlererkennung und -Lokalisierung wurde bewußt hintangestellt.

[+] Dies stellt eine Besonderheit beim Materialfluß dar, weil die Speichereigenschaft der Transportstrecke technisch genutzt werden kann. Es war der Grund für die Einführung der gemischten Modelle im Kapitel 2.

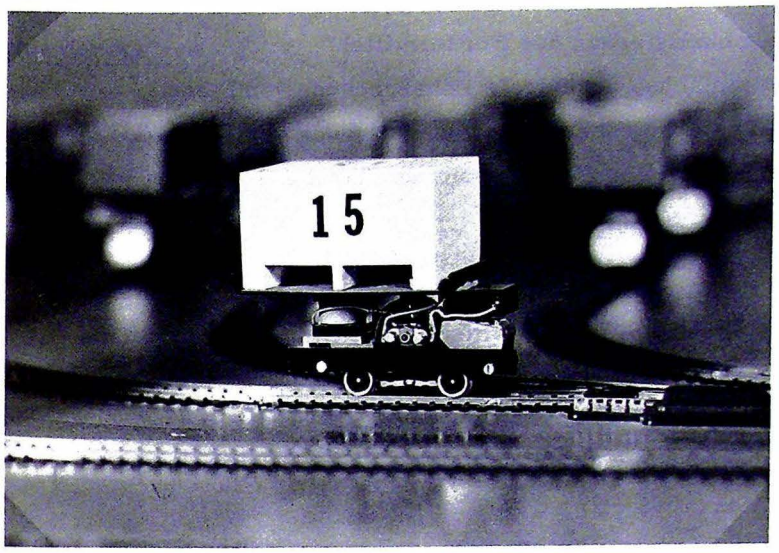


Bild 4.1 : Ansicht eines mit einer Modell-Palette beladenen, diskreten Förderelementes

Eine Untersuchung /4.5/, deren wichtigste Gesichtspunkte neben den geschilderten Zielkriterien

- Verhütung von Kollisionen
- leichte Realisierbarkeit der Anlage
- leichte Lokalisierbarkeit von Stücken bzw. Transporteinheiten
- leichte Lenkbarkeit der Transporteinheiten, insbesondere an Verzweigungen und Zusammenführungen

waren, hatte zum Ergebnis, daß zum Transport keine stetigfördernden Einrichtungen herangezogen, sondern am günstigsten als intermittierendes Fördermittel umgebaute Modelleisenbahn-Lokomotiven des Bildes 4.1 eingesetzt werden. Die Linienführung ist also "schienengebunden", die Förderrichtung "waagerecht", die Beweglichkeit "selbstfahrend" und die Antriebsart "Maschine" (Klassifizierung nach /4.6/).

Die Entscheidung zugunsten des intermittierenden Fördermittels erfolgte vor allem deshalb, weil im Zuge der vorgenommenen Unterteilung der

Förderstecken in Blockabschnitte

a) die Lokalisierung der Fördermittel
b) die einfache, dezentrale Beherrschung von Staus
c) die einfache, dezentrale Beherrschung von Konfliktsituationen bei Verzweigung und Zusammenführung

besonders einfach zu realisieren waren.

Der Nachteil ist, daß man im Grunde mit diskreten Fördermitteln zwei Materialflüße erhält : den - gewollten - Fluß der Stücke und den - eigentlich ungewollten - Fluß der Transportmittel. Da natürlich ohne Transporteinrichtung ein Nutz-Materialfluß nicht zustande kommt, kann man sich auf die Transporteinheit (vgl. 2.1.1) beschränken, ohne daß die beabsichtigte Untersuchung beeinträchtigt wird.

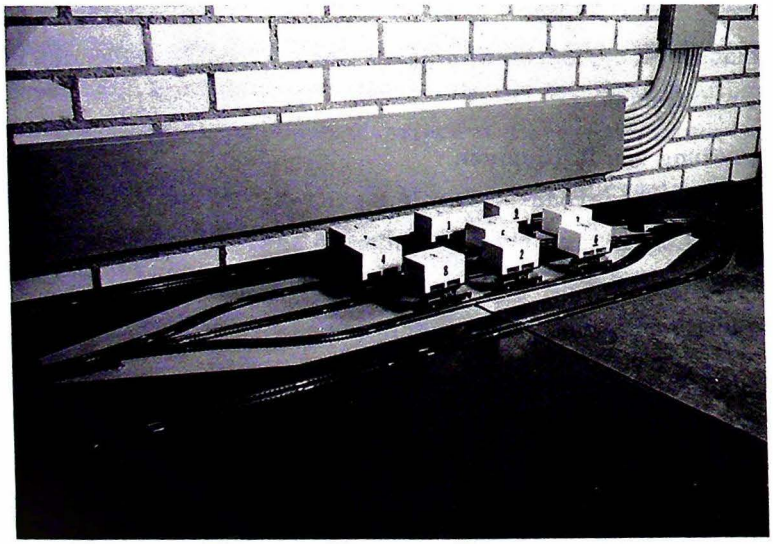


Bild 4.2 : Beispiele der Realisierung von Verzweigungen und Zusammenführungen im Modellprozeß

4.2.2 Die Warenverteilanlage

Struktur der Anlage

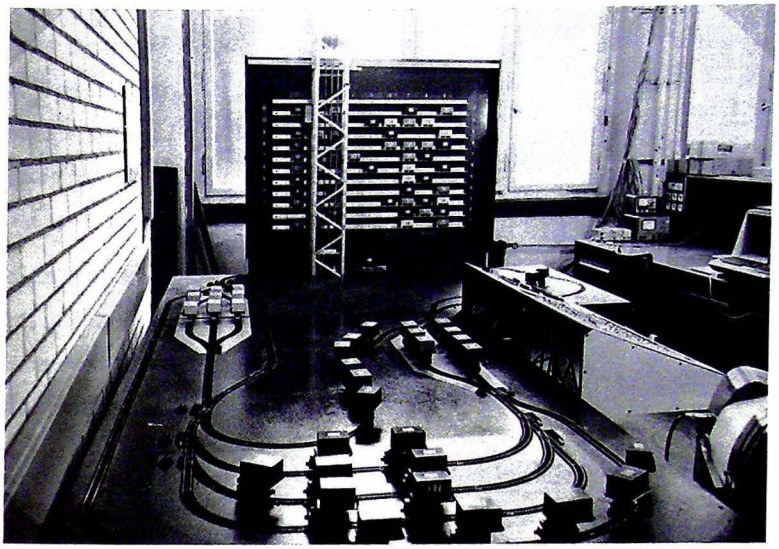


Bild 4.3 : Gesamtansicht der Warenverteilanlage

Bild 4.3 zeigt vor dem Hintergrund des aufgebauten Hochregallagers eine Gesamtansicht der realisierten Warenverteilanlage. (Die physikalischen Außenmaße waren durch die Platzverhältnisse im Labor vorgegeben). Das zugehörige Handsteuerpult, das mit Leuchtanzeigen die momentane Belegung mit Transporteinheiten wiedergibt, zeigt Bild 4.4. Mit seiner Hilfe ist es möglich, per Hand Ausfahrbefehle an die Strecken vor den Konflikt-Stellen "Zusammenführung" und "Verzweigung" zu geben.

Die Förderstrecken sind gerichtet. Die Transportrichtung ergibt sich aus den Pfeilen des Bildes 4.4.

Um das Verfahren der Ziellenkung mit variablem Weg auch im Modellprozeß anwenden zu können, wurde beim Entwurf darauf geachtet, daß mehrere Alternativen bei der Zielsteuerung auch physikalisch gegeben sind.

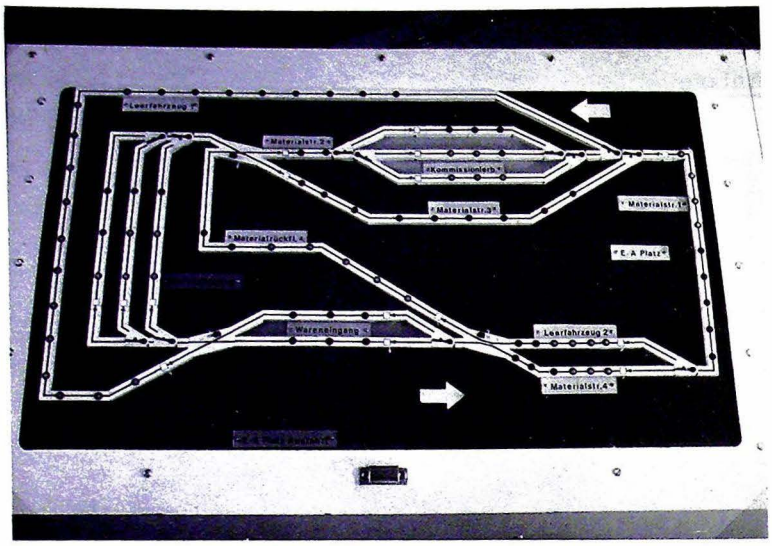
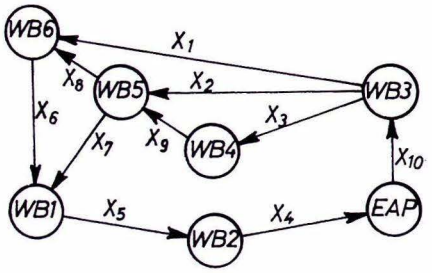


Bild 4.4 : Handsteuerpult zur Warenverteilanlage



Von \ Nach	WB1	WB2	WB3	WB4	WB5	WB6	EAP
WB1		■					
WB2							■
WB3				■	■		
WB4					■		
WB5	■					■	
WB6	■						
EAP			■				

EAP = Ein/Ausgabeplatz Hochregallager

WB1 ... WB5 = Weichenbereiche 1 ... 5

Bild 4.5 : Vereinfachter Stationendigraph des realisierten Stückprozesses mit schematischer Darstellung der Adjazenmatrix

Wie aus den Bildern 4.3 und 4.4 hierzu hervorgeht, sind zwischen verschiedenen Kreuzungen und Weichen bis zu drei Alternativ-Strecken vorhanden. Sie dienen außer als diverse Endziele bei der Kommissionierung, d.h., Ziel-

lenkung von Stücken auch als alternative Wegstrecken zur adaptiven Wegelenkung. Wie der vereinfachte Stationen-Digraph im Bild 4.5 zeigt, gibt es darüber hinaus "überregionale" Alternativwege über die Stationen hinaus.

Allein der Anschaulichkeit wegen wurden in der Anlage Strecken und Bereiche mit Bezeichnungen wie etwa "Wareneingang", "Warenausgang", "Materialstrecke x" etc. versehen, obwohl es natürlich von der Aufgabe der Stückverfolgung und -Lenkung her völlig gleichgültig ist, ob ein Stück nun in ein Lager oder zum Warenausgang zu bringen ist.

Wenn man den Ein/Ausgabe-Platz ("EAP") als einfachen Durchgangsplatz betreibt, erhält man zahlreiche Nutzflußkreise, die auch die automatische Fehlerkorrektur gestatten /4.2/. (Kreise sind für die Rückführung der Transportmittel ohnehin unerläßlich.)

Aussteuerung und Lokalisierung der Transporteinheiten

Die Anlage wurde in Blockabschnitte eingeteilt. Als Abschnitte gelten

- einzelne Teilstrecken längs einer Strecke zwischen den Stationen
- Weichen- und Kreuzungsbereiche (Stationen)
- die Übergabe-Station von Hochregallager zur Warenverteilanlage.

Diese Unterteilung dient der abschnittsweisen Lokalisierung und wird zur Aussteuerung der Transporteinheiten mitbenutzt.

Ein Strang der Eisenbahnschienen ist durchgehend, der andere durch Isolierstücke im Abstand von wenigstens 20 cm unterbrochen. Jedes dieser so entstehenden, elektrisch voneinander isolierten 91 Teilstücke ist an eine eigens hierfür entwickelte Elektronik angeschlossen, mit deren Hilfe festgestellt wird, ob sich eine Transporteinheit auf einem Abschnitt befindet. (Dies sind die Anzeigen auf dem Steuerpult, die auch dem Rechner zugeführt werden.)

Gleichzeitig werden diese 91 Einzelanschlüsse dazu benutzt, durch Aufschaltung einer Gleichspannung die umgebauten Elektroloks in Bewegung zu setzen. D.h., die Fahrbewegung wird binär - von Hand oder per Rechner - gesteuert : Entweder "Fahren", d.h. Fahrspannung anlegen ; oder "Halten", d.h., keine Fahrspannung anlegen.

Durch gegenseitige Verbindung der Einzel-Blocksicherungen untereinander längs durchgehender Förderstrecken zwischen den Stationen wurde erreicht, daß sich nur jeweils eine Transporteinheit in einem Abschnitt befinden kann. Die Bewältigung von Staus geschieht dezentral so, daß bei Ausfahrt des vordersten Elements alle dahinter gestauten automatisch nachrücken +).

Das automatische Nachrücken längs Strecken mit Zwangslauf könnte auch zentral - vom Rechner bzw. vom Menschen - bewältigt werden. Die vorgenommene Realisierung der dezentralen Stauabwicklung wurde aber für sinnvoll gehalten, weil es andernfalls den Bedienaufwand am Steuerpult enorm vergrößert hätte. (Die Belastung des Rechners sinkt ebenfalls bezüglich der auszugebenden Stellbefehle.)

Ansteuerung der Weichen

In Modelleisenbahntechnik ist im Gegensatz zur Realität die Stellung der Weichen bei Zusammenführungen unerheblich, was ebenfalls eine angenehme Entlastung bei der Bedienung darstellt.

Die Ansteuerung der verzweigenden Weichen, den Verzweigungen erfolgt vom Handsteuerpult über einen Impuls (Taster) und vom Rechner über eine zwischengeschaltete Logik auf die Wertänderung eines Ausgabebits hin. 15 Weichen sind insgesamt zu stellen.

Die Erfassung (und damit die unmittelbare Fehlerdiagnose) der tatsächlichen Weichenstellung ist aufgrund der hardwaremäßigen Gegebenheiten der verwendeten Modelleisenbahnweichen nicht möglich. Zu Beginn der Operationen im Modellprozeß müssen daher auf Verdacht hin alle Weichen in eine definierte Ausgangslage gebracht werden. Im Betrieb der Anlage zu erkennen, daß ein Stück falsch gelenkt wurde, ist erst durch einen Soll-Ist-Vergleich der Belegung der einzelnen Förderabschnitte mittels eines Prozeßmodells im Rechner möglich.

+) Das automatische Nachrücken bzw. Anhalten wird hier elektrisch bewältigt.

Übergabeplatz an das Hochregallager

Als Puffer zur Ver- und Entsorgung wurde in die Warenverteilanlage ein Hochregallager integriert (siehe 4.3).

Die Übergabestelle ist die Schnittstelle zwischen Fördersystem und Hochregallager. Der Übergabeplatz inklusive Hochregallager ist in der Modellvorstellung des Kapitels 2 dieser Arbeit aber nichts weiter als eine Station, die einen Speicher enthält. +) Sie ist deshalb im Bild 4.5 als Ecke dargestellt.

Das besondere technische Problem an diesem Übergabeplatz war die genaue Justierung der Transporteinheit so, daß ein Abheben oder Absetzen einer Palette durch das Regalförderzeug möglich war. Eine sehr gute Lösung - wenn man die unterschiedlichsten Laufeigenschaften von Gleichstrom-Modellelektroloks bedenkt - wurde durch Drei-Punkt-Regelung mittels zweier Lichtschranken und Pulsbreitenmodulation erreicht /4.7/.

Rechner/Handbetrieb

Alle Belegungsanzeigen der Förderanlage wurden grundsätzlich an das Handsteuerpult geführt. Zusätzlich sind alle Belegungen an den Prozeßrechner AEG 60-50 angeschlossen.

Durch eine umfangreiche Umschaltlogik können Stellbefehle an Strecken und Stationen entweder vom Handsteuerpult über Drucktaste oder vom Rechner über Digitalausgabe gegeben werden.

Ein gemischter, überlagerter Betrieb wurde nicht vorgesehen, d.h., entweder kommen alle Stellbefehle vom Rechner oder vom Handsteuerpult.

4.2.3 Einige Prozeßmodelle des Modellprozesses

Das Bild 4.5 zeigt das vereinfachte Stationenmodell des realisierten Modellprozesses, in dem die existierenden Mehrfachverbindungen zwischen den Ecken nicht ersichtlich sind. Mit der Nachbarmatrix ist man, wie die symbolische Darstellung in Bild 4.5 zeigt, eben nicht in der Lage, den durch die Mehrfachverbindungen zwischen einzelnen Ecken entstehenden Multigraphen zu beschreiben : die Adjazenzmatrix gibt eben nur die Nachbarschaft wieder.

+) Es handelt sich um die einzige Station im Modellprozeß, die nicht aus Zusammenführung oder Verzweigung besteht.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
WB1					+1	-1	-1			
WB2				+1	-1					
WB3	+1	+1	+1							-1
WB4			-1						+1	
WB5		-1					+1	+1	-1	
WB6	-1					+1		-1		
EAP				-1						+1

Bild 4.6 : Inzidenzmatrix des vereinfachten Stationenmodells des Modellzesses

Schon die zugehörige Inzidenzmatrix dieses einfachen Graphen in Bild 4.6 zeigt anschaulich, daß von den 10x7 Matrizenplätzen der größte Teil, nämlich 50 oder 71 % nicht benutzt sind. Hätte man den vollständigen Multigraphen mit q = 15 Kanten herangezogen, wäre dieser relative Anteil mit 1-2/p zwar gleich groß, die absolute Zahl aber mit 70 Plätzen bereits erheblich höher (siehe Anhang 1).

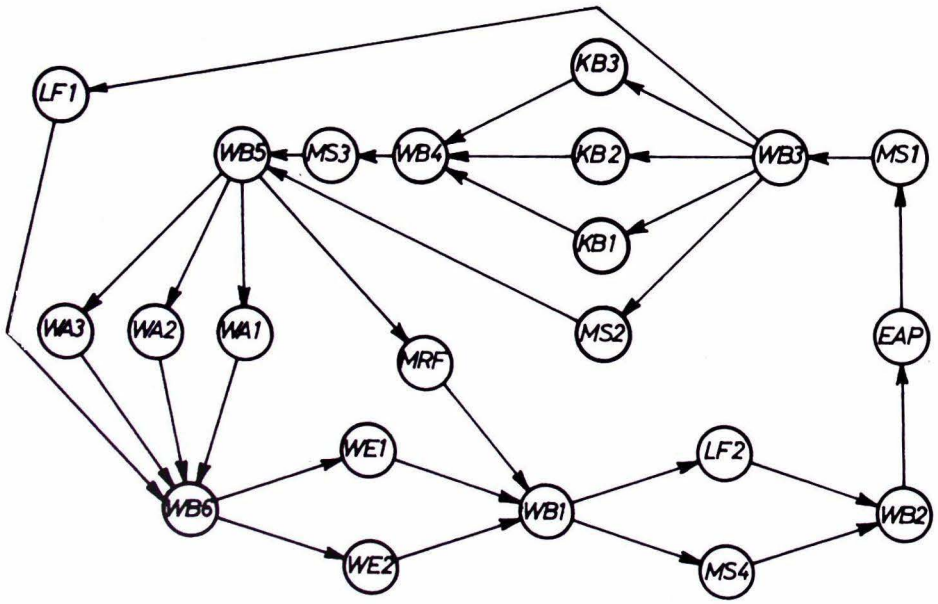


Bild 4.7a : Wege/Stationen-Modell des Modellprozesses

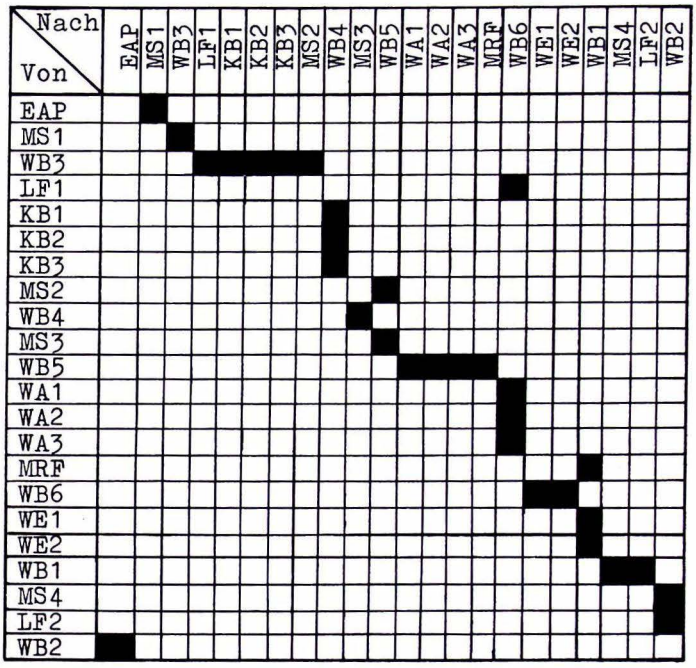


Bild 4.7b : Nachbarmatrix des Wege/Stationenmodells

Das grundlegende Modell bei allen Überlegungen zur Programmierung der geschlossen prozeßgekoppelten Stückguterfassung und Lenkung in PEARL in /4.6/ zeigt Bild 4.7. Die Ecken des Graphen können zu bestimmten Zeitpunkten eines oder mehrere Stücke bzw. Transporteinheiten enthalten, nicht dagegen die Kanten, die ausschließlich die Materialflußmöglichkeiten in idealisierter, immaterieller Form wiedergeben. Die zugehörige Nachbarmatrix beschreibt dann, welche Stationen oder Strecken miteinander verbunden sind. Auch hier zeigt sich, daß dieses so anschauliche Modell den Mangel der vielen unbesetzten Plätze hat. Wenn man erst alle 91 vorhandenen einzelnen Blockabschnitte als Ecken im Wege-/Stationen-Graph darstellt, zeigt sich, daß von den 91 x 91 = 8291 Plätzen der Nachbarschaftsmatrix nur 99 besetzt sind, also ein - untragbarer - Rest von 8192 unbenutzten Plätzen entsteht.

Man wird also, wie diese Beispiele anhand der geschlossen prozeßgekoppelten Inbetriebnahme des Modellprozesses bestätigt haben, auf Ersatzdarstellungen ausweichen müssen, weil eine derartige Platzverschwendung sicherlich für jede Prozeßrechnerinstallation untragbar ist.

Der Nutzen der hier vorgestellten Modellbildung liegt m.E. darin begründet, daß die Elemente des Stückprozesses zunächst einmal in Form von Graphen dargestellt werden, um die Zusammenhänge losgelöst von Details der Geometrie und Hardware zu abstrahieren. Daß die daraus abgeleiteten, äquivalenten Matrizenformen - nicht so anschaulich wie ein Graph - in Rechnern zweckmäßigerweise durch ökonomischere Abspeicherungsformen ersetzt werden, berührt den Wert der Modellbildung erst sekundär.

4.3 Das gegenständliche Modell eines Hochregallagers

4.3.1 Das Hochregallager als Station

Als ein typisches Beispiel für die dezentralen, lokalen Aufgabenstellungen im Stückprozeß kann das im Modellprozeß realisierte Hochregallager des Bildes 4.8 gelten. In der makroskopischen Betrachtungsweise ist dies eine

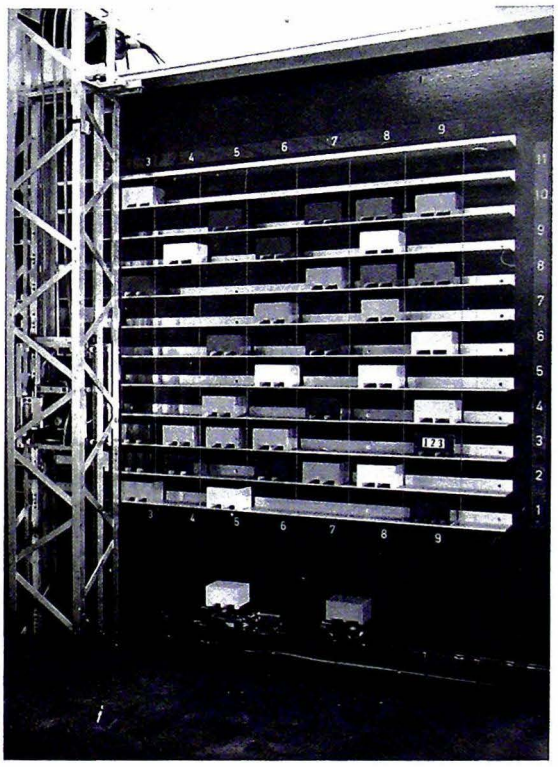


Bild 4.8 : Ansicht des Hochregallagermodells mit dem Übergabeplatz

Station, von der für die Stückgutlenkung an zentraler Stelle das Wissen genügt, ob die Station frei ist oder mit was für einem Stück sie besetzt ist. (Nach /2.6/, S. 160, ist ein Hochregallager lediglich ein "zweidimensionales Kommissioniersystem".)

Selbstverständlich will man an zentraler Stelle i. a. auch wissen, welche Stücke sich gerade in der Station "Lager" befinden ; das Wissen hierüber kann selbstverständlich Ursache dafür sein, daß der Wunsch nach Ortsveränderung von Stücken auftritt. Der von der Zentrale an diese Station auszugebende Befehl ist dann beispielsweise : "Stelle Stück s_i zur Verfügung".

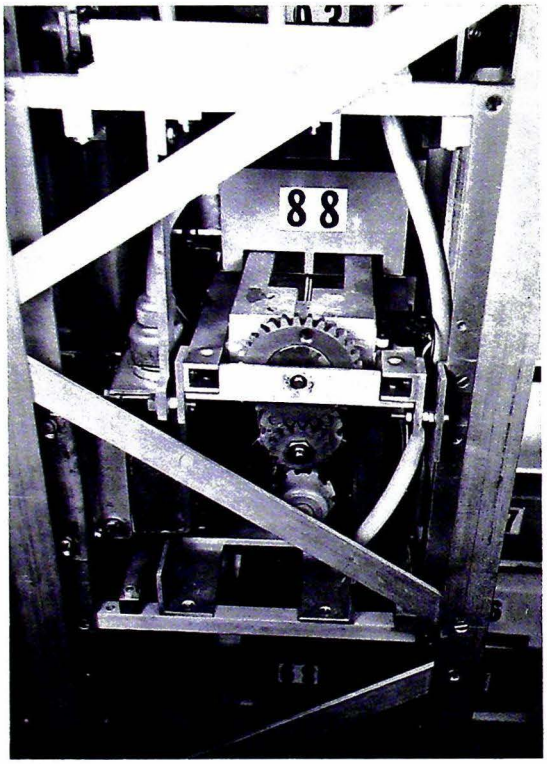


Bild 4. 9 : Ansicht der Regalförderzeuggabel

Die Station ist dann vollkommen autonom dazu in der Lage, den Befehl auszuführen, eine Zentrale benötigt man hierzu nicht unbedingt.

4.3.2 Anmerkungen zum Aufbau

Weil es zur Darstellung der Vorgänge genügt, wurde nur eine einzelne, halbseitige Hochregallagergasse aufgebaut.

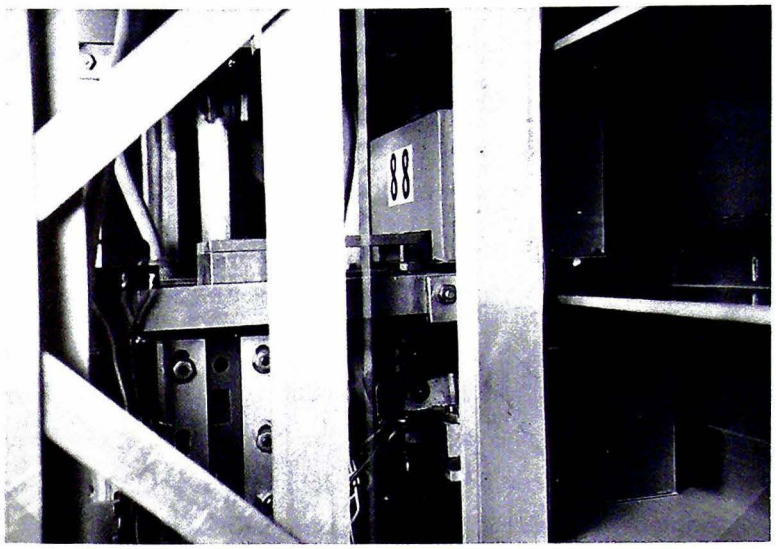


Bild 4.10 : Gabelspiel des Regalförderzeug

Der Lagerraum selbst umfaßt 9 x 11 Lagerplätze zur Aufnahme der Modellpaletten. (Die Zahl von 9 x 11 ergab sich aus den räumlichen Gegebenheiten im Labor. Daß dabei eine mehr quadratische als die in der Realität üblichen, sehr viel längeren als hohen Regalflächen entstanden ist, hat auf die Funktion des Modells keinen Einfluß.)

Die einzelnen Plätze sind durch die Einführung eines karthesischen Koordinatensystems in steigender Folge durchnummeriert. Der Übergabeplatz (EAP) als Schnittstelle zum Warenverteilsystem ist ebenfalls ein - zwar ausgezeichneter - Platz in diesem Koordinatensystem mit den Koordinaten x = 5, y = 0. Er befindet sich in der Mitte der Hochregallagerwand : Als Besonderheit fällt die Übergabestelle für Einlagerung und für Auslagerung zusammen.

Vor der Hochregallagerwand befindet sich das Regalförderzeug, das "Fördermittel für Ein- und Auslagerungsvorgänge in Regalanlagen" /2.1/ - nach der Klassifizierung dieser VDI-Richtlinie ein sog. regalabhängiges Regalförderzeug, weil es außerhalb des Regals nicht funktionsfähig ist. Es wird unten durch eine Bodenschiene (im Bild 4.8 durch die Förderanlage verdeckt) und oben auf dem Regal seitlich durch Rollen abgestützt.

Im rahmenförmigen Hubmast, der horizontal mittels Gleichstrommotor und Seilzug bewegt wird, ist die lastaufnehmende Regalförderzeuggabel angebracht, die mit Gleichstrommotor und Seilzug vertikal bewegt werden kann.

Die Regalförderzeuggabel - Bild 4.9 zeigt die Antriebe - kann außer in der Horizontalen auch in der Vertikalen vorn am Gabelende in gewissen Grenzen verfahren werden.

Dadurch ist es möglich, beim Aufnehmen oder beim Absetzen einer Palette die Position des Lastaufnahmemittels insgesamt unverändert beizubehalten. Im Gegensatz zum Verfahren der kurzen Hubbewegung /4.8/ erfolgt hier Aufnehmen und Absetzen allein durch lokale Gabelbewegungen.

Als Positionserfassungsverfahren wurde das digital absolute Meßverfahren gewählt, das bei der geringen Anzahl der Regalfächer sehr einfach zu realisieren war. Bei Überlaufen von Regalplatzmitten wird die in fünf Bit codierte Istposition parallel mit Gabellichtschranken abgelesen. Über Zwischenpositionen erhält man keine Kenntnis. Der praktische Betrieb (s.u.) hat gezeigt, daß diese (Un-) Genauigkeit genügt, wenn man nicht speziell auf die Optimierung der Regalförderzeugbewegungen abzielt.

4.3.3 Betrieb des Regalförderzeugs

Ein Handsteuerpult dient der Bedienung einer in konventioneller TTL aufgebauten, dezentralen Steuereinrichtung. Diese hat sich als sehr nützlich herausgestellt, weil der Mensch bei der Aufgabe der genauen ($\pm$ 1 mm) Positionierung leicht überfordert ist, insbesondere dann, wenn in der Horizontalen und Vertikalen gleichzeitig zu positionieren ist, weil das Ansteuern einer bestimmten Sollposition im Hochregallager in zwei gemeinsam initiierte, aber voneinander unabhängige Bewegungen zerfällt.

Das Handsteuerpult zeigt die letzte überlaufene Ist-Position in Dezimaldarstellung an und ermöglicht so die indirekte Verfolgung des Regalförderzeugs. Zwei Tastaturen dienen zur dezimalen Handeingabe der Sollpositionen durch den Bediener - eine für die waagerechte, die andere für die senkrechte.

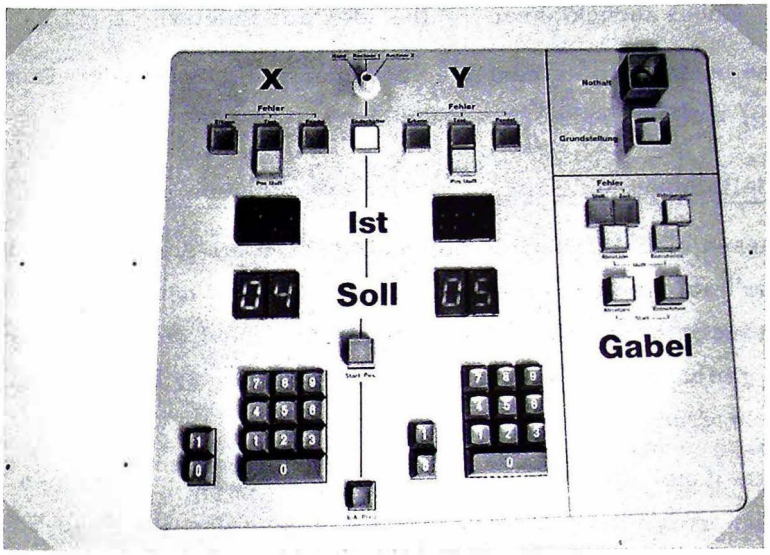


Bild 4.11 : Ansicht des Handsteuerpults für die Regalförderzeugsteuerung

Nach Eintasten der gewünschten Zielposition kann durch Betätigen der Start-Taste die Positionierung mittels der erwähnten Steuerungseinrichtung gestartet werden.

Ist die Sollposition erreicht, kann der Gabellauf gestartet werden (Bedienfeld rechts), und zwar entweder "Entnehmen" oder "Absetzen".

Ein Wählschalter am Steuerpult schaltet diese Steuereinrichtung so um, daß alle Signale zum Rechner geleitet werden bzw. alle Stellbefehle - die sonst aus der Steuereinrichtung kämen - vom Rechner gegeben werden müssen.

Die direkte, digitale Steuerung des Regalförderzeugs im geschlossen prozeßgekoppelten Betrieb wurde sowohl in Assembler /4.9/ als auch in PEARL programmiert.

Dabei stellen sich die Vorgänge als mit den typischen Merkmalen der Klasse der Folgeprozesse behaftet dar : So ist etwa das Entnehmen einer Plalette

- Gabel ausfahren (unter Palette einfahren)
- Gabel anheben (Palette anheben)
- Gabel zurückfahren (beladen mit Palette)

durch eine bestimmte Reihenfolge von Einzelereignissen oder Einzelzuständen /4.10/ zu überwachen bzw. zu steuern.

4.3.4 Eine Modellbildung am Beispiel des Hochregallagers

Global lassen sich Aufgaben in der Station "Kommissionierautomat Hochregallager" auf einen ähnlichen Nenner bringen wie bei den makroskopischen Aufgaben im Stückprozeß : "Bringe ein Stück -eine Palette- vom Ort A zum Ort B" - gleichgültig ob es sich um eine Einlagerung, eine Auslagerung oder einen Umlagerungsvorgang handelt.

Man könnte daher diese Station - wie jede andere auch - als einen abgeschlossenen Stückprozeß begreifen und zweckmäßigerweise das Wege-Stationenmodell bilden. Es bestünde aus 99 Ecken als Darstellung der Regalplätze. Dazu eine Ecke, die die Übergabestelle und noch eine zusätzliche Ecke, die die Regalförderzeuggabel repräsentiert.

Daß das Regalförderzeug seine räumliche Position verändert, ist unerheblich. Entscheidend ist nämlich, daß der Raum, den das Regalförderzeug überstreicht, als besetzt oder als frei beschrieben werden kann. Man ordnet so diesen Raum - als Kern des Wege/Stationentheorems - eine Ecke zu, in Bild 4.12 als "Gabelecke" bezeichnet.

Wegen der freien Beweglichkeit des Regalförderzeugs in der Regalebene ist die Gabelecke (bzw. der überstrichene Raum) mit jeder der Regalplatz-Ecken sternförmig und bidirektional verbunden, während nur zwei Kanten zwischen Gabelecke und Ein/Ausgabe-Platz existieren.

Man kommt aber trotzdem mit nur einem Beobachter für die Zentrale aus, weil sich in dieser Anordnung zur gleichen Zeit nie mehr als ein Stück bewegen kann. Daher ist dieses Modell von lokalem Nutzen, hat aber überregional so keine Bedeutung.

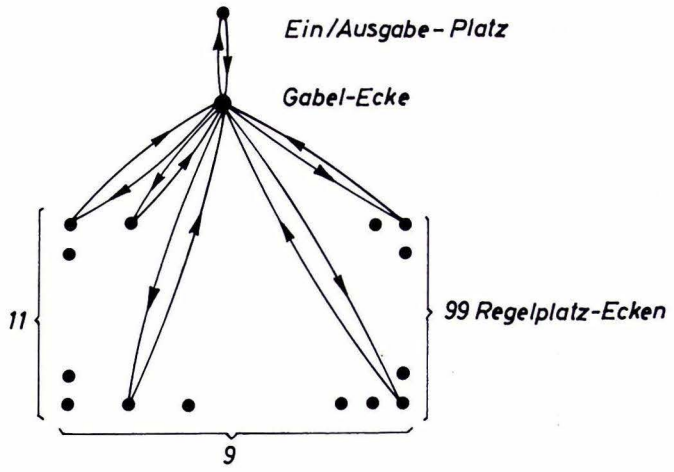


Bild 4.12 : Schematische Darstellung des Hochregallagermodells als Wege/Stationen-Graph

Überregional könnte man z.B. ein Prozeßmodell im Rechner ganz einfach in Form einer 9 x 11 Matrix zum Abspeichern der gerade aktuellen Belegung benutzen. Dieses Prozeßmodell dient insofern nur sekundär dem Ziel der Prozeßlenkung, weil die Erzeugung eines Transportauftrags an das Regalförderzeug zwar von dieser Belegungsliste mit abhängig ist; steht der Transportauftrag aber einmal fest, benötigt man zum direkten Prozeßeingriff die Belegung nicht mehr, weil sie sich während der Transportabwicklung nicht frei, sondern nur in bestimmten Folgen verändern kann.

5. Grundlegende Anforderungen an eine Prozeßprogrammiersprache bei der Automatisierung eines Stückprozesses

5.1 Ereignisse

Die charakteristische Eigenschaft des Stückprozesses mit einzelnen identifizierbaren Stücken, daß das Prozeßgeschehen sich nicht hinreichend durch einfache Ereignisse erfassen läßt, gibt Anlaß, sich kurz mit Ereignissen und ihre Verwendung in Rechnern zu beschäftigen.

In /5.1/ wird ein Ereignis allumfassend als "Eintritt des Erfülltseins einer Synchronisationsbedingung" definiert. Im LTPL-E-Sprachvergleich /5.2/, ähnlich umfassend beschrieben, wird es unterteilt in drei Arten : Verbunden ("connected"), frei ("free") und lose ("loose"). Die verbundenen und losen sind so unmittelbar von der Abwicklung von Rechenprozessen abhängig, daß ohne Rechenprozeß auch kein Ereignis dieser Kategorien eintreten kann. Die sog. freien, d.h. unabhängigen hingegen werden benötigt, um die Aktivitäten von Rechenprozessen überhaupt erst zu initiieren ; sie sind für die Stückgutverfolgung von ausschlaggebender Bedeutung.

Üblicherweise wandelt man in zu automatisierenden, gegenständlichen Prozessen relevante Vorkommnisse, d.h., freie Ereignisse - das Über- oder Unterschreiten von bestimmten, vorgegebenen physikalischen Schwellen - in Unterbrechungssignale und führt sie dem Prozeßrechner zu. Die Überschreitung der Schwelle "Zeitpunkt" stellt nur insofern eine Ausnahme dar, als die hardwaremäßige Zuführung des zugehörigen Unterbrechungssignals üblicherweise intern bereits vorhanden ist.

Über entsprechende Einrichtungen werden damit letztlich zugeordnete Rechenprozesse initiiert, so daß "Reaktionen" auf das Ereignis erfolgen können (siehe hierzu Kap. 6).

Sehr häufig ist eine Reaktion, d.h., die Abwicklung eines Rechenprozesses auf das Ereignis des Überschreitens einer bestimmten, vorgegebenen Zeitschwelle erwünscht. Benutzt man andere Größen als direkt die Zeit, so kann sowohl Über- als Unterschreiten von bestimmten, vorgegebenen Grenzwerten das Ereignis darstellen.

Um den Istzustand des nicht-anonymen Stückprozesses zu ermitteln, mußten Beobachter noch die Aufgabe der Identifizierung der Stücke übernehmen. D.h., das Ereignis "Stück hat Abschnitt betreten" ist durch die Identität des beobachteten Stücks zu ergänzen.

Dazu wird man zunächst jede Meldung des Beobachters B_m in ein Unterbrechungssignal umwandeln und mit diesem einen Rechenprozeß anstoßen, der dann die Identität des beobachteten Stücks von irgendwelchen externen Hardwareeinrichtungen abliest oder aus einem Prozeßmodell herleitet, d.h., die Beobachtung ergänzt.

5.2 Verarbeitung von M Beobachtungen

5.2.1 Die disjunktive Verknüpfung

Das Anstoßen eines Rechenprozesses läßt sich beispielsweise in der Prozeßprogrammiersprache PEARL wie in Beispiel 5.1 lösen.

Beispiel 5.1 :

```
WHEN ITR_6 ACTIVATE IDENT_6; /* Interrupt Nr. 6 mit Task
                                verknüpfen */
.....
.....
IDENT_6 : TASK;
TAKE FROM EX_IDENT_6 TO I_6; /* Ablesen Identität */
.....
END ;
```

Nur wenn ausschließlich ITR_6 und keine anderen Bedingungen mit der Task IDENT_6 verknüpft sind, dann ist nach Ablauf der Task die Beobachtung des Beobachters "6" mit dem Satz (T_6, b_6, I_6) vollständig ergänzt.

Das besondere Problem besteht jedoch darin, daß insgesamt M Beobachter[+)] gegeben sind.

Im Beispiel 5.1 wird gezeigt, wie die Identität i_k (=I_6) des Stückes und der Zeitpunkt t_k (=T_6) durch den einfachen Algorithmus:
"Eingabe-Operation für I_6" ermittelt werden kann. Gibt es M Beobachter,

---

+) Z.B. im realisierten Modellprozeß M = 91 Belegungsmeldungen.

entsteht für jeden dieselbe Aufgabenstellung, nämlich aus dem unqualifizierten Ereignis mittels eines Rechenprozesses den gewünschten Satz (t_k, b_m, i_k) zu erzeugen.

Daher wird man sich selbstverständlich bemühen, für alle M Beobachter denselben Algorithmus und damit letztlich auch dasselbe Rechenprozeßsegment zur Ermittlung der Identität eines Stückes zu verwenden. Daraus ergibt sich die erste Anforderung :

A1) Bedingungen müssen so mit einer Task verknüpfbar sein, daß jedes beliebige zugehörige Ereignis, das Erfülltsein jeder beliebigen aus M möglichen Bedingungen die Abwicklung der Task bewirken kann. Diese Art der Verknüpfung werde künftig als "disjunktiv" bezeichnet.

Beispiel 5.2 zeigt in PEARL-Notation, wie alle M elementaren Unterbrechungen - mit B1 bis BM bezeichnet - einer einzigen Task zugeordnet werden.

Beispiel 5.2 : Disjunktive Verknüpfung von M Bedingungen

```
WHEN B1, WHEN B2, ... WHEN BM ACTIVATE IDENT ;
IDENT : TASK ;
.....
.....
END ;
```

5.2.2 Unterbrechungsfelder

Das Anlisten einer jeden der M Unterbrechungen in WHEN-Statements ist nur als datenverarbeitungstechnischer Anachronismus zu bezeichnen. Wie für übliche Datentypen, die als Felder angeordnet und per Indizierung angesprochen werden können, muß auch Indizierung erlaubt sein ; daher die 2. Anforderung :

A2) Unterbrechungen müssen als Felder beschreibbar sein, so daß einzelne Unterbrechungen als Elemente dieser Felder indiziert angesprochen werden können.

Dann könnte man die Verknüpfung der einzelnen Bedingungen mit einer Task etwa in einer Schleife wie in Beispiel 5.3 vornehmen.

Beispiel 5.3 : Verknüpfung in einer Schleife

```
FOR J FROM 1 BY 1 TO M REPEAT ;
WHEN B (J) ACTIVATE IDENT; /* Element verknüpfen */
END ; /* Ende der Schleife */

IDENT : TASK ;
.....
.....
END ;
```

Daraus ergibt sich jedoch sofort ein neues Problem.

Im Beispiel 5.2 wurde per Notation unterstellt, daß z. B. die Teileinplanung "WHEN B2" nicht die Einplanung "WHEN B1" aufhebt, allgemein eben jede der Bedingungen parallel und unabhängig voneinander mit der Task IDENT verbunden ist.

Löst man dagegen die Schleife im Beispiel 5.3 gemäß ihrem Ablauf auf, so ergeben sich insgesamt die aufeinanderfolgenden Verknüpfungen :

```
WHEN B(1) ACTIVATE IDENT ;
WHEN B(2) ACTIVATE IDENT;
:::
:::
WHEN B(M) ACTIVATE IDENT ;
```

Jetzt muß grundsätzlich vorausgesetzt werden, daß nicht jede neue Verknüpfung Bedingung - Task jede alte Verknüpfung ersetzt. Sonst wäre nach Ablauf der Schleife nur noch die Verknüpfung der Bedingung WHEN B(M) mit der Task IDENT[+)] gültig.

Unabhängig davon, ob nun alle Verknüpfungen grundsätzlich - bis auf Widerruf, etwa mit der PEARL-Anweisung "PREVENT" - über alle Zeiten gehäuft werden oder ob nur die letzte, gerade überlaufene gelten soll, ist diese Art der Verknüpfung möglicherweise ungeeignet. Die Abarbeitung der Schleife könnte nämlich an beliebiger Stelle unterbrochen und evtl. längere Zeit blockiert sein.

+) Zum Problem des "schedule-Häufens" siehe /5.3/, /5.5 - 5.6/. In PEARL gilt, daß jede neue Einplanung für dasselbe Objekt die alte desselben Typs ersetzt /1.4, S 101 ff./.

Je nachdem an welcher "Stelle" die Unterbrechung eintritt, ist eine unbekannte Zahl von Bedingungen oder eine einzelne, nicht näher bezeichenbare Bedingung mit der Task verknüpft. Dieser Zustand ist natürlich unbefriedigend. Deswegen ist Unteilbarkeit der "WHEN... ACTIVATE"-Anweisungskombination Voraussetzung für ein klares und eindeutiges, vernünftiges Funktionieren der disjunktiven Verknüpfung. So sei im Beispiel 5.2 entweder keine oder es sind alle Bedingungen mit IDENT verknüpft.

Die sehr problematische Möglichkeit, auf Sprachebene Anweisungen zuzulassen, die die Unterbrechung von beliebigen Anweisungsblöcken verbieten, scheidet aus, weil solche Anweisungen eine ständige Gefahr von Verklemmungen darstellen. So benötigt man ein anderes Mittel zur disjunktiven Verknüpfung von Bedingungsfeldern mit einem Rechenprozeßsegment.

A3) Disjunktive Verknüpfung von Unterbrechungsfeldern muß in einer besonderen Art von "Wiederholung" möglich sein, die unteilbar ist.

Die erste Möglichkeit besteht in der Einführung einer besonderen Art von Schleife - eventuell mit Einschränkungen wegen der Fehleranfälligkeit - im WHEN-Statement an einer Position, an der ohnehin Unteilbarkeit, d.h. Ununterbrechbarkeit vorausgesetzt wird. Die zweite Möglichkeit[+)] stellt die Verknüpfung des gesamten Ereignisfeldes oder von zusammenhängenden Teilen ("Slices") zu einer Task im WHEN-Statement dar.

5.2.3 Das Auftraggeberproblem

Alle bis dato eruierten Anforderungen ergaben sich aus der praktischen Erwägung, unabhängige Ereignisse disjunktiv mit einer Task zu verknüpfen. Wenn man aber im Rechenprozeß auf Sprachebene - wie etwa in den Programmiersprachen nach /5.2/ - kein Mittel zur Verfügung hat, um zu erkennen, welches Ereignis für die gerade aktuelle Abwicklung verantwortlich zeichnet, d.h., welches der gerade aktuelle Auftraggeber ist, so waren alle Bemühungen bis hierher nutzlos.

Ist der Rechenprozeß erst einmal aktiv, so kann innerhalb der Task durch keinen wie auch immer gearteten Algorithmus mehr festgestellt werden, welcher Beobachter sich gemeldet hatte. Die Aufgabe, einzelne Stücke zu

---

+) Diese zweite Möglichkeit wurde auf Betreiben des Autors in die Sprache PEARL aufgenommen /5.4/.

lokalisieren, ist so nicht lösbar. Es ist dabei sogar völlig unerheblich, ob es sich bei den Beobachtern um identifizierende oder nicht-identifizierende Beobachter handelt, weil die Schwierigkeiten bei der Lokalisierung der Ereignisse, d.h. bei der Unterscheidung der Herkunft der Ereignisse selbst entstehen. Somit bliebe ernstlich nur die Alternative, jeder Unterbrechung B(1) bis B(M) einen eigenen Rechenprozeß exclusiv wie im Beispiel 5.4 zuzuordnen[+].

Beispiel 5.4 : M Rechenprozesse

```
WHEN B(1) ACTIVATE TASK1 ;
:
WHEN B(M) ACTIVATE TASKM ;
TASK1 : TASK
REQUEST SEMA ;
I = 1 ;
CALL ALGORITHMUS ;
RELEASE SEMA ;
END ;

:
TASKM ;
REQUEST SEMA ;
I = M
CALL ALGORITHMUS ;
RELEASE SEMA ;
END ;
:
ALGORITHMUS : PROCEDURE ;
:
END ;
```

Natürlich ist diese "Alternative" äußerst unbefriedigend : immerhin sind M Tasks zu codieren, die sich in nichts als in einer einzigen Wertzuweisung (I = ..) voneinander unterscheiden ; für größere Beobachterzahlen M ist sie sogar unbrauchbar, wenn eine gegebene Prozeßrechnerinstallation nicht in der Lage ist, M Tasks zu unterstützen[++].

[+] Im Beispiel 5.4 tritt das Problem der konkurrierenden Rechenprozesse offen zutage, während es im Beispiel 5.2 in der Aktivierung verborgen ist; Eine Behandlung dieses Problems erfolgt in 7.2

[++] Z.B. gestattet die PEARL-Implementation am Prozeßrechner AEG 60-50, an der der Autor dieser Arbeit mitgewirkt hat, lediglich 48 Tasks, während im Modellprozeß 91 benötigt würden.

Beide Alternativen,

- die disjunktive Verknüpfung vom M Bedingungen mit einer Task

oder

- M Tasks

scheiden also praktisch für die Lösung des Problems aus.

So zeigt sich, daß diese Aufgabenstellung mit Hilfe einer Prozeßprogrammiersprache dann nicht oder nur mit unvernünftigem Aufwand lösbar ist, wenn es mit Mitteln der verwendeten Sprache nicht möglich ist, den Auftraggeber eines Rechenprozesses zu erkennen. Der Einsatz einer solchen Prozeßprogrammiersprache erbringt im Vergleich zur Assemblerprogrammierung eindeutige Nachteile. Denn :

- Entweder ist man gezwungen, wie etwa im realisierten Modellprozeß[+)], zusätzliche, externe Hardwarehilfsmittel einzusetzen. So haben dann Spracheigenschaften zusätzliche und unerwartete Rückwirkungen auf die Hardware.
- Oder man muß M Tasks einsetzen, um die Rückwirkungen Sprache - externe Prozeßhardware zu vermeiden.

Die dritte und letzte Alternative bestünde nun darin, eine Mixtur von höherer Sprache und Assembler zu verwenden.

Mit Assembler-Anweisungen und entsprechenden Schnittstellen zum unterliegenden Betriebssystem ist es i.a. immer möglich, einen Auftraggeber von einem anderen zu unterscheiden. Ein in Assembler geschriebenes Modul könnte dann in höherer Sprache geschriebenen Tasks die Identität des Auftraggebers zur Verfügung stellen.

Natürlich ist auch diese Alternative äußerst unbefriedigend, wurde doch die höhere Sprache vor allem deshalb geschaffen, um der leidigen Assemblerprogrammierung mit all ihren Nachteilen zu entgehen. So läßt sich das Problem nur lösen, wenn die folgende Anforderung erfüllt ist :

A4) Ein Rechenprozeß muß in der Lage sein, seinen Auftraggeber zu erkennen.

+) Einsatz eines Sammelinterrupts durch eine Hardware, die Zustandsänderungen erkennt und selbstprogrammierte Identifizierung des Ereignisses durch Vergleich mit der Vorgeschichte.

5.3 Zusammenfassung

Das Problem der abschnittsweisen Stückverfolgung mittels M Beobachter - das M-Beobachter-Problem - führte zu folgenden elementaren funktionellen Anforderungen an die Eigenschaften einer Prozeßprogrammiersprache :

- Disjunktive Verknüpfung : Eine Menge von Bedingungen muß alternativ so einer Task zugeordnet werden können, daß jedes einzelne zugehörige Ereignis unabhängig von allen anderen den Rechenprozeß anstoßen kann.
- Unterbrechungsfelder : Wie andere Datentypen auch, müssen Unterbrechungen in Form von Feldern beschreibbar sein.
- Auftraggeberidentifizierung : Eine Task muß ihren gerade aktuellen Auftraggeber erkennen können.

Während man ohne die bis hierher genannten Eigenschaften die Stückgutverfolgung praktisch nicht oder nur mit unvernünftigem Aufwand lösen kann, ist die Anforderung der Unteilbarkeit von disjunktiven Verknüpfungen gemessen an der Lösbarkeit der Aufgabe nicht unbedingt erforderlich. Sie ist aber empfehlenswert, da man damit leicht einen unbestimmten Zustand beseitigen kann.

6. Mechanismen und Spracheigenschaften zur Behandlung von Aufträgen

Bei der Stückverfolgung entsteht neben dem im vorangegangenen Kapitel dargestellten Problem bei der Einplanung ein zeitliches Problem, wenn nämlich die Beobachter ihre Beobachtungen an eine Zentralstelle weiterleiten : Die Tätigkeiten der Zentralstelle - der Rechenprozeß - und die Beobachtungen, d.h., die freien Ereignisse überlappen sich zeitlich, so daß die eingetretenen Ereignisse bzw. deren Zielsetzungen aufbewahrt werden müssen, bis sie abgearbeitet werden können.

Dieses Kapitel befaßt sich daher mit den folgenden zwei Problemen :

1) Das Ursprungsproblem

   Wie kann der individuelle Verursacher eines Zustandsänderungswunsches bestimmt werden, wenn die Zustandsänderungwünsche durch mehrere freie, disjunktiv verknüpfte Ereignisse auslösbar sind.

2) Das Verbindungsproblem

   Wie ist der Bindemechanismus zwischen Ereignis (Auftraggeber) und dem Zielrechenprozeß (Auftragnehmer), dessen Zustandsänderung der Ereigniseintritt bewirken soll.

6.1 Mechanismen des Zustandswechsels am Beispiel von PEARL

6.1.1 Rechenprozeßzustände

Im VDI/VDE-Richtlinienentwurf 3354 /6.1/ können Rechenprozesse[+)] verschiedene Prozeßzustände annehmen, deren wichtigste Merkmale zusammengefaßt sind :

- ruhend : es liegt (zum Ablauf des Rechenprozesses) kein Auftrag vor. Der Rechenprozeß ist jedoch angemeldet.
- bereit : alle Voraussetzungen zum Ablauf des Rechenprozesses bis auf die Zuteilung des Prozessors sind erfüllt.
- laufend : der Rechenprozeß besitzt den Prozessor.
- blockiert (1 u. 2) : der Prozessor und/oder andere Betriebsmittel wurden entzogen.

In manchen Prozeßprogrammiersprachen sind mehr als diese 4 bzw. 5 Zustände /5.2/ definiert. Von ausschlaggebendem Interesse sind für den Benutzer eigentlich aber nur die Zustände, die von ihm per Steueranweisung veränderbar sind. (Zustände bzw. Zustandswechsel, die er nur indirekt oder gar nicht beeinflussen kann, muß er zwar - zumal in konkurrierender Umgebung - in Betracht ziehen, wird er aber gleichwohl eher als hinderlich denn als nützlich bezeichnen.)

Zur Behandlung der Zustandswechsel in PEARL kann man sich daher in Anlehnung an die erwähnte VDI-Richtlinie und an /5.1/ auf die vier Zustände ruhend, bereit, laufend und blockiert beschränken.

6.1.2 Taskzustandsänderungen

Zustandsdiagramm

Zweck der Einführung dieser Zustände ist es, mit ihnen Konzepte zur Steuerung von Rechenprozessen, d.h. die Wirkung von gewissen Steueran-

---

+) Der Abwicklung einer PEARL-Task äquivalent ist der dort definierte "(Rechen-) Prozeß 2. Art", der in der vorliegenden Arbeit abkürzend und vereinfacht Rechenprozeß genannt wird. Unter dem Begriff "Task" sei hier "das einem Rechenprozeß zugeordnete Programmstück" verstanden /6.13/.

weisungen, den (explizit) zustandsändernden Anweisungen in einer Prozeß-programmiersprache zu erklären.

Üblich ist hierzu die Darstellung in Form eines sog. Zustandsdiagrammes, eines gerichteten Pseudo-Graphen. Die Zustände werden dabei als Knoten oder Ecken, die möglichen Übergänge von einem Zustand zu einem anderen mittels gerichteter Kanten oder Bögen dargestellt.

Den meisten Kanten kann man wie in Bild 6.1a direkt eine bestimmte zustandsändernde Anweisung der Programmiersprache zuordnen, mit der Semantik, daß dieser Übergang mit dieser Anweisung vorgenommen wird[+].

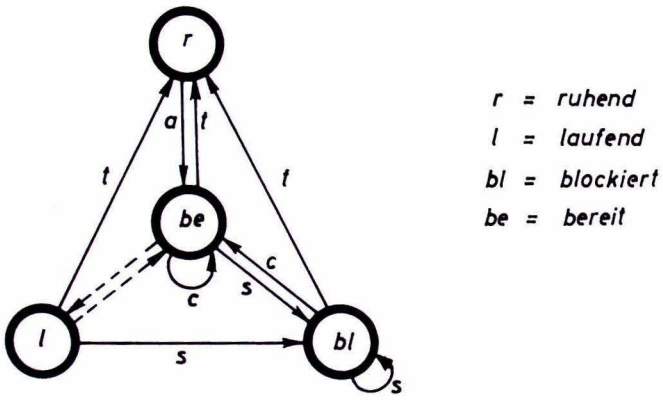


Bild 6.1a : Pseudo-Zustandsgraph für PEARL-Tasks nach /5.1/.

Daneben gibt es "besondere" Kanten, d.h. Übergänge, denen nicht direkt eine Steueranweisung gegenübersteht. Dies sind Übergänge, die von einem Organisator /5.1/, dem unterliegenden Betriebssystem vorgenommen werden. Sie sind auf implizit zustandsändernde Anweisungen des Rechenprozesses (z.B. Ein/Ausgabe-Anweisungen) oder auf bestimmten Konkurrenzsituationen mit anderen Rechenprozessen (z.B. Unterbrechung durch höherpriore Rechenprozesse) zurückzuführen und betreffen letzlich nur die Prozessorzuteilung (vgl. 6.1.1).

[+] a) Die in /6.2/ vorgeschlagene "Übergangstafel" zur Beschreibung der Zustandswechsel basiert auf einer Matrizendarstellung eines Graphen und ist so substantiell äquivalent.

b) Die Zuordnung Kante-Anweisung ist nicht notwendigerweise umkehrbar, vgl. Bild 6.1a.

von \ nach	r	l	bl	be
r	-	-	-	a_f
l	$t_{e,f}$	-	$s_{e,f}$	
bl	t_f	-	s_f	c_f
be	t_f		s_f	c_f

a ≙ ACTIVATE
c ≙ CONTINUE
s ≙ SUSPEND
t ≙ TERMINATE

Indizes:
e ≙ Zustandsänderung durch eigene Task
f ≙ Zustandsänderung durch fremde Task

Bild 6.1b : Anweisungen zur Zustandsänderung in PEARL, unterschieden nach Fremd- und Eigen-Aktionen.

Der Graph gibt weiterhin keine Auskunft darüber, ob der Zustandswechsel eines bestimmten Rechenprozesses auf dessen Ablauf, d.h. dessen zustandsändernde Anweisung selbst oder auf die Anweisung eines anderen, konkurrierenden Rechenprozesses erfolgen kann.

In /5.2/ werden deshalb die Anweisungen daraufhin unterschieden, wessen Zustand sie beeinflussen ("self-actions" und "actions on other tasks"). Diese Möglichkeiten für PEARL zeigt Bild 6.1b.

In einigen der Prozeßprogrammiersprachen nach /5.2/, darunter PEARL, kann der Zustandswechsel auch noch auf Eintreffen eines freien Ereignisses hin vorgenommen werden. Auch dies ist nicht aus der Graphendarstellung ersichtlich[+].

Explizite Anweisungen zur Zustandsänderung

Eine explizit zustandsändernde Anweisung[++] hat in PEARL die allgemeine Form (alle Syntax-Notationen nach /1.4/).

[+] Die Graphendarstellung gibt so letzlich nur einen bescheidenen Einblick in die Semantik der zustandsändernden Anweisungen. Die Darstellung in Form von Netzen (z.B. /6.4/) ergibt zwar einerseits eine grundsätzliche Verbesserung, aber andererseits einen gewissen Verlust an Übersichtlichkeit (z.B. /6.5/).

[++] Die explizit zustandsändernde Anweisung hat ausschließlich die Zustandsänderung zum Ziel. Andere Anweisungen, die als Nebeneffekt potentiell auch zu Zustandsänderungen führen, gelten in dieser Arbeit als implizit zustandsändernd.

(6.1) zustandsanweisung :: = [schedule] taskoperation [objekt] [sonstiges] ;

Sie zerfällt in 4 Teile :

1) Ereignisteil (schedule), optional :
Er beschreibt, bei welchen freien Ereignissen die Taskoperation ausgelöst werden soll.

2) taskoperation :
Die eigentliche zustandsändernde Operation Aktivierung, Suspendierung, Fortsetzung, Verhütung oder Terminierung[+] :
"ACTIVATE", "SUSPEND", "CONTINUE", "PREVENT", "TERMINATE".

3) objekt, optional :
Der Rechenprozeß, auf den die Taskoperation angewandt werden soll.

4) sonstiges : Sonstige Möglichkeiten nach /1.4/, optional, u.a..
Angabe der Priorität des Objekts.

Die Möglichkeiten, die diese umfassende Anweisung zur Steuerung von Zustandsänderungen eines Rechenprozesses

- aufgrund des Eintretens von freien Ereignissen,
- auf eigenes Verlangen ("self-action") und
- auf Verlangen fremder Rechenprozesse ("actions on other tasks")

bietet, zeigt Bild 6.2 (zu den Begriffen "hart" und "weich" siehe 6.1.3).

Zustandsanweisungen mit Ereignisteil werden im folgenden als "bedingt" bezeichnet, weil zur Exekution der Taskoperation die Erfüllung einer Bedingung, der Eintritt eines Ereignisses Voraussetzung ist ; die Zustandsanweisung ohne Ereignisteil wird entsprechend als "unbedingt" bezeichnet.

[+] Die RESUME-Anweisung wird als unteilbare Kombination aus Suspendierung und Fortsetzung hier nicht gesondert behandelt. Die PREVENT-Anweisung hebt Einplanungen auf und löscht vorliegende Aufträge, stellt also eigentlich eine Ausnahme dar.

	Anweisung		Zustandsänderung wird ausgelöst durch			Kopplung nach /6.3/
	Ereignisteil vorhanden	Objekt vorhanden	freies Ereignis	eigenen Rechenprozeß	fremden Rechenprozeß	
1	nein	nein	nein	ja	nein	hart
2	nein	ja		nein	ja	
3	ja	nein	ja			weich
4	ja	ja				

Bild 6.2 : Steuerung von Zustandsänderungen und "Kopplung"

Die syntaktische Konstruktion lautet :

(6.1a) zustandsanweisung :: = { bedingte_zustandsanweisung | unbedingte_zustandsanweisung }

(6.1b) bedingte_zustandsanweisung :: =
schedule unbedingte_zustandsanweisung

(6.1c) unbedingte_zustandsanweisung :: =
taskoperation [objekt] [sonstiges] ;

Die scharfe semantische Trennung nach bedingten und unbedingten Zustandsanweisungen geht auf /6.6/ zurück. Sie hat sich als zweckmäßig herausgestellt, wie das folgende Beispiel zeigt :

Beispiel 6.1 : Ein "trivialer" Fall

```
A : TASK ;
    SUSPEND;
    PUT DORA EDIT ('NANU') (A (4)) ;
END ;        /* ENDE DER TASK A */
```

Gibt man der unbedingten Zustandsanweisung (ohne Ereignisteil, 6.1c) die Bedeutung, daß diese nur als Sonderfall der bedingten Anweisung gelten soll, etwa mit dem Ereignisteil "AFTER 0 SEC", so ist folgende Situation vorstellbar : Die SUSPEND-Anweisung im Beispiel wird dem Organisator (Betriebssystem) "übergeben" mit der Maßgabe, die Blockierung einer bestimmten Task (hier eben A) nach 0 sec vorzunehmen. Nach dieser "Übergabe" oder Einplanung kann die Task A fortgesetzt werden.

Ob nun die Suspendierung (siehe Blockierung nach 6.1.1) von A vor der Ausführung der PUT-Anweisung durchgeführt wird, ist ungewiß. Wenn nämlich der Organisator geradeeben - nach Ablauf der Zeitspanne N sec, wobei zufällig eben N = 0 ist - mit der Bedienung anderer, vielleicht wichtigerer Rechenprozesse beschäftigt ist, kann es sein, daß die PUT-Anweisung und eventuell folgende vor der Suspendierung exekutiert wird. Dies lag aber sicher nicht in der Absicht des Programmierers.

Für PEARL wurde dieses Problem durch scharfe semantische Unterscheidung von bedingten und unbedingten Zustandsanweisungen wie folgt gelöst /6.3/ :

A) Zustandsändernde Anweisungen ohne Ereignisteil sind dann abgearbeitet, wenn der Zustandswechsel des referierten Rechenprozesses stattgefunden hat. Der Rechenprozeß, der den Wechsel verlangt, kann so erst nach erfolgreicher Exekution der Taskoperation weitergeführt werden. +)

B) Zustandsändernde Anweisungen mit Ereignisteil sind dann abgearbeitet, wenn die Taskoperationen eingeplant sind, d.h. dem Organisator übergeben wurden. Der Rechenprozeß, der den Wechsel verlangt, kann unmittelbar danach weitergeführt werden, ohne daß die Exekution der verlangten Taskoperationen abgewartet wird.

Die Anweisungen der Kategorie A werden als "hart", die der Kategorie B als "weich" an den Zustandswechsel des referierten Rechenprozesses "gekoppelt" bezeichnet.

6.1.3 Das Problem der weichen Kopplung bei der Stückverfolgung

Inhaltliche Zielsetzung von Taskoperationen

Weil sie für die folgende Untersuchung von Bedeutung ist, wird an dieser Stelle eine grundlegende Unterscheidung der Taskoperationen nach ihren Zielsetzungen eingeführt.

+) Die unbedingte Zustandsanweisung läuft so in Sequenz ab wie jede andere Anweisung auch, und nicht parallel zu anderen Anweisungen desselben Rechenprozesses. Der mit dieser Festlegung verbundenen Gefahr von Verklemmungen wird durch Auslösung von sog. Signalen begegnet /6.3/.

1) "Tätigkeitsauslösende" Taskoperationen sind Operationen, deren Ziel es ist, Aktionen oder Tätigkeiten[+)] von Rechenprozessen herbeizuführen.

2) "Tätigkeitshemmende" Taskoperationen sind Operationen deren Ziel es ist, Aktionen oder Tätigkeiten von Rechenprozesses zu unterbinden, zu verhüten oder zu beenden.

Tätigkeitsauslösend in diesem Sinne sind in PEARL die Aktivierungs- und Fortführungsanweisung "ACTIVATE" und "CONTINUE", tätigkeitshemmend sind Verhinderungs-, Suspendierungs- und Terminierungsanweisung "PREVENT", "SUSPEND" und "TERMINATE". (Die RESUME-Anweisung ist substituierbar aus beiden Gruppen.)

Tatsächlich ist nach Übergang in den Zustand "blockiert" oder "ruhend" das erstrebte Ziel "Tätigkeit unterbinden oder hemmen" vollständig erreicht.

Dagegen kann das Erreichen des Zustandes "bereit" nur als Voraussetzung zur Durchführung von Tätigkeiten bezeichnet werden. Das eigentliche, substantielle Ziel einer tätigkeitsauslösenden Taskoperation - Bearbeitung einer Aufgabe mittels Rechenprozeß - wird nur mit Hilfe des Organisators erreicht, der den Zustand "laufend" herbeiführt.

Die direkten Einwirkungsmöglichkeiten des Programmierers auf Rechenprozesse sind so auf die Zustandswechsel beschränkt, wobei in einem Falle Zustandswechsel und substantielles Ziel zusammenfällt, im anderen aber nicht. Den erfolgten Zustandswechsel kann man so lediglich dem formalen Erreichen des mit der Taskoperation assoziierten Zieles gleichsetzen.

Das Problem der Zustandsänderung aufgrund von Ereignissen

Wenn man bei der unbedingten Zustandsanweisung noch vorschreibt, welche Reaktionen vom Organisator beim erfolglosen Zustandswechselversuch zu erfolgen haben, so sind diejenigen Einwirkungsmöglichkeiten vollständig erklärt, die aus dem Fortschritt eines Rechenprozesses heraus auf einen anderen oder auf den eigenen Rechenprozeß gegeben sind.

+) Unter Tätigkeit oder Aktion ist hier die "tatsächliche Bearbeitung (eines Rechenprozesses) in einer Funktionseinheit eines Rechensystems" /6.1/ zu verstehen.

Unbedingte Zustandsanweisungen können nur beim Ablauf von Rechenprozessen zu Zustandsänderungen führen ; darin unterscheiden sie sich ausschlaggebend von den bedingten Anweisungen.

Das Ziel der bedingten Zustandsanweisung kann es nur sein, bei Eintritt eines freien Ereignisses einen bestimmten Rechenprozeß zu beeinflussen, d.h. es handelt sich um die Einplanung eines Zustandswechsels.

Die Exekution der Zustandsänderung eines beliebigen Rechenprozesses ist eventuell schon deshalb nicht unmittelbar bei Ereigniseintritt möglich, weil dieser sich - zufällig - nicht im adäquaten Zustand befinden könnte. Per definitionem ist das auslösende, freie Ereignis aber nicht von Zuständen irgendwelcher Rechenprozesse abhängig und so sein Eintritt nicht zu verhindern oder zu blockieren.

Selbst wenn man auf irgendeine Weise den Zustandswechsel doch erzwingen wollte, ist bei tätigkeitsauslösenden Taskoperationen das substantielle Ziel - die Tätigkeit - ja durch den Zustandswechsel alleine noch nicht erreicht.

Die Bearbeitungswünsche (Aufträge), ausgelöst durch den Eintritt eines freien Ereignisses, könnten so einfach verloren gehen, wenn man nicht Bindeglieder einsetzt, um diese aufzubewahren, d.h. zwischenzuspeichern oder zu puffern, und zwar so lange, bis das substantielle Ziel des Auftrages erreicht wird.

Wie erwähnt, entsteht diese Problemstellung genau bei der Stückverfolgung : Während ein Rechenprozeß als Reaktion auf die Beobachtung eines Stückes hin abläuft, ist es möglich, daß parallel hierzu, d.h., zeitlich überlappend durch die Bewegung anderer Stücke wiederum Beobachtungen anderer Beobachter anfallen, d.h., neue Bearbeitungswünsche entstehen.

6.2 Ermittlung des Auftraggebers

6.2.1 Begriffe

Nach /6.7/ ist ein Auftraggeber

> "eine Instanz, die eine andere Instanz zur Erbringung einer bestimmten Datenverarbeitungsleistung verpflichtet",

ein Auftragnehmer ist analog dazu

> "eine Instanz, die von einer anderen Instanz zur Erbringung einer bestimmten Datenverarbeitungsleistung verpflichtet wird".

Auf eine Prozeßprogrammiersprache wie PEARL übertragen, stellen die einzelnen Bedingungen des Ereignisteils in (6.1) die verpflichtenden "Instanzen", also die Auftraggeber dar. Die Instanz zur Erbringung einer bestimmten Leistung - der Auftragnehmer - ist das Codesegment bzw. Programm (Task), dessen "Abwicklung auf einem Rechensystem" nach /6.1/ einen Rechenprozeß darstellt. (Begreift man Instanzen als etwas lebendiges, tätiges, so könnte dem Auftraggeber das Ereignis, dem Auftragnehmer der Rechenprozeß entsprechen.)

Bei Erfülltsein einer Bedingung, dem Eintreten eines Ereignisses, entsteht ein Auftrag an einen Auftragnehmer. Der "Auftragsabwicklung", den "im Auftragnehmer ablaufenden Vorgängen zum Erbringen einer bestimmten Auftragsleistung" entspricht der Fortgang eines Rechenprozesses.

Das "Auftragskennzeichen", nach /6.7/ ein Kennzeichen zur Unterscheidung gleichzeitiger Auftragsbindungen, könnte das gewünschte Kennzeichen sein, das zur Identifizierung des auslösenden Ereignisses bei der Auftragsabwicklung benötigt wird (Ursprungsproblem).

Die "Auftragsbindung"- "die Verpflichtung eines Auftragnehmers gegenüber einem Auftraggeber zur Erbringung einer bestimmten Datenverarbeitungsleistung"-entsteht bei der Einplanung mittels bedingter Zustandsanweisungen.

Die inhaltliche Bedeutung eines Auftrages, d.h., seine Zielsetzung ergibt sich aus den Taskoperationen : tätigkeitshemmend oder tätigkeitsauslösend. Als Auftragsabwicklung eines hemmenden Auftrages könnte man sehr formal die Überführung eines Rechenprozesses in den Zustand des "Nichtstun" bezeich-

nen, sofern man in diesem Zusammenhang überhaupt noch den Begriff "Abwicklung" gebrauchen will. Es erscheint jedenfalls sehr viel sinnvoller, von Auftragsabwicklung nur im Zusammenhang mit tätigkeitsauslösenden Aufträgen zu sprechen.

6.2.2 Ein elementares Verfahren zur Ursprungsermittlung

Um ein Auftragskennzeichen weiterleiten zu können, muß zunächst eine Vorschrift erarbeitet werden, wie dies zu bilden ist.

Deshalb ist die bedingte Zustandsanweisung nach (6.1b) um die Fähigkeit zur Übermittlung der Identität des aktuellen Auftraggebers zu erweitern.

Das elementare Abzählverfahren

Der einfachste Vorschlag ist sicherlich die Methode des Abzählens. Man ordnet jeder der M Einzelbedingungen des Ereignisteils[+)] einfach durch Abzählen die Zahlen von 1 bis M zu. Dann wird vom Organisator verlangt, er möge bei Eintreten eines der Einzelereignisse die zugehörige Zahl ermitteln und "zur Verfügung stellen".

Da dies nicht bei allen bedingten Zustandsanweisungen erforderlich ist - wenn etwa kein Auftragskennzeichen benötigt wird - ist es zweckmäßig, dieses Verlangen auch durch entsprechende Notation, z.B. durch entsprechende Schlüsselworte, in der Zustandsanweisung durch den Programmierer steuerbar zu gestalten.

Zur Erläuterung der Funktionsweise dienen die folgenden Beispiele :

Beispiel 6.2 : Verschiedenartige Ereignisse

AT CLOCK1, AFTER DUR1, WHEN ITR1, WHEN ITR2

Ist z.B. die Bedingung "AFTER DUR1" (Ablauf einer Zeitdauer) erfüllt, so wird (vom Organisator) als Ursprung die Zahl 2, tritt z.B. das Ereignis (Unterbrechung) ITR1 ein, so wird die Zahl 3 ermittelt.

+) Disjunktive Verknüpfung, d.h. disjunktive Auftragsbindung vorausgesetzt.

Beispiel 6.3 : Eine Unterbrechungsscheibe ("interrupt-slice")

WHEN ITR (7 : 15) ACTIVATE ...

Wenn eine der Unterbrechungen aus der Scheibe von 7 bis 15 auftritt, wird als Ursprung die Zahl 1 bis 9 ermittelt.

Insbesondere bei Interruptscheiben böte sich die Ursprungszuordnung gemäß den Selektionsgrenzen an, im vorigen Beispiel also der Zahlenbereich von 7 bis 15.

Es ist aber zweckmäßig, die Abzählung rigoros bei 1 zu beginnen, weil man dann potentielle Mehrdeutigkeiten bei Verknüpfungen von mehreren Feldern wie im folgenden Beispiel verhindert.

Beispiel 6.4 : Mehrere Unterbrechungsscheiben

WHEN ITRA (3 : 6), WHEN ITRB (5 : 7) ...

Das Abzählverfahren ist sicher nicht die einzige Lösungsmöglichkeit. Lösungen wie etwa die Zuordnung von beliebigen Zahlen, Zeichen, Variablen, Zeigern oder Feldern erscheinen komfortabler und "eleganter". Sie sind aber alle auf den geschilderten Primitivmechanismus reduzierbar oder beinhalten ihn im Kern ; sie werden deshalb im Rahmen dieser Arbeit nicht weiter behandelt.

Manipulation des Ursprungs mit Sprachmitteln

Es genügt nicht, daß der Organisator die Ursprungsinformation ermittelt, sie muß auch so hinterlegt werden, daß der Auftragnehmer auch über sie verfügen kann.

Als einfachstes Verfahren erscheint die Einführung einer Hilfsvariablen vom Typ Ganzzahl, in die die Ursprungsinformation vom Organisator abzulegen ist. Zur syntaktischen und semantischen Kennzeichnung könnte man die Hilfsvariable mit entsprechendem Präfix in die Zustandsanweisung aufnehmen, z. B. "ORIGIN identifier". +)

+) Der Gültigkeitsbereich der Hilfsvariablen "identifier" muß dann mindestens dem der gesamten Zustandsanweisung entsprechen, sie dürfte dann also nicht in beliebigen Blöcken, sondern nur auf übergeordneter Ebene (z. B. Modul-Ebene) vereinbart werden, was vom Compiler aber überprüfbar ist.

Von der formalen Gültigkeit unabhängig ist die Frage, was denn zu einem bestimmten Zeitpunkt der Inhalt dieser Hilfsvariablen sein soll ; Es sei immer die Zahl, die der Nummer des letzten, aufgetretenen Ereignisses entspricht.

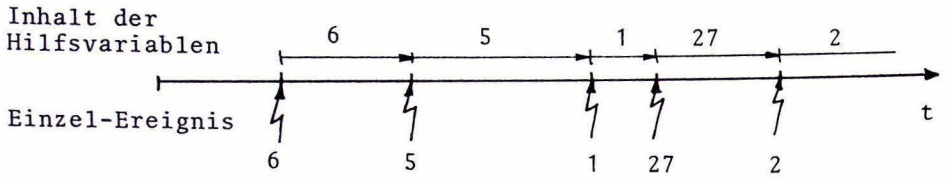


Bild 6.3 : Ein Beispiel zum Inhalt der Hilfsvariablen bei Eintritt mehrerer Ereignisse

Wie das Beispiel in Bild 6.3 - verschiedene Stücke betreten nacheinander zu verschiedenen Zeitpunkten die Abschnitte 6, 5, 1, 27, 2 etc. - zeigt, wechselt eine solche Hilfsvariable ihren Inhalt mehr oder weniger rasch, je nach Ereignisanfall. Daß die Zeitspanne, während deren der Inhalt unverändert bleibt, zur Verwendung im auftragnehmenden Rechenprozeß - z.B. zur Ausgabe eines Stellbefehls - nicht immer ausreichend sein muß, zeigt die folgende "worst-case"-Betrachtung.

Es können nämlich "gleichzeitig", d.h. innerhalb einer bestimmten Zeitspanne Δt, der durch Hardware- und Software-Grenzen gegebenen Minimalauflösbarkeit des Rechners (und des Organisators) mehrere Ereignisse auftreten, also z.B. mehrere Stücke "gleichzeitig" beobachtet werden. Vom Organisator kann verlangt werden, daß keines der Ereignisse übergangen bzw. vergessen wird.[+)] Dann würde, wie das Beispiel in Bild 6.4 vereinfacht zeigt, nur während einer bestimmten, i.a. sehr kleinen Zeitspanne τ - der Bearbeitungszeit des Organisators für ein einzelnes Ereignis - das Auftragskennzeichen in der Hilfsvariablen zur Verfügung gestellt.

+) Diese Aussage über das Nicht-Übergehen von Einzelereignissen bezieht sich nur auf Ereignisse unterschiedlichen Ursprungs. Treten innerhalb eines bestimmten Intervalls mehrmals Ereignisse desselben Ursprungs ein, so könnten sie verloren gehen, da es nicht möglich ist, diese unendlich oft zu speichern (vgl. hierzu Kap. 7).

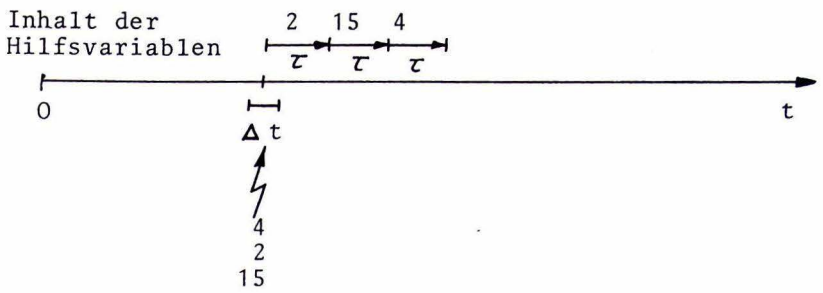


Bild 6.4 : Überlappung von Ereignissen : 3 Stücke betreten innerhalb Δt die Abschnitte 4, 2 und 15

Damit nun kein Auftragskennzeichen verlorengeht, muß der Auftragnehmer dies stets rechtzeitig entnehmen. Organisator bzw. Teile davon und Rechenprozeß müßten also streng synchronisiert, d.h. sequentialisiert werden. Weil aber der Auftragnehmer ein beliebiger, also auch blockierbarer Rechenprozeß sein kann, könnte auch der Organisator bzw. dessen Teil, der freie Ereignisse abhandelt, blockiert werden. Genau dies war aber nicht beabsichtigt, weil eine Blockierung des Organisators in dieser Art auf jeden Fall zum Verlust des gesamten Auftrages inklusive Kennzeichen (wenn nicht zu noch schlimmeren) führen muß.

Eine Verwendung einer Hilfsvariablen in der geschilderten Form als statisches Kommunikationshilfsmittel zur Weiterleitung und Manipulation der Auftraggeberidentität ist also so nur in Ausnahmefällen, nicht aber allgemein möglich.

Dasselbe gilt für die Alternativ-Lösung mittels einer Standard-Funktionalprozedur, die bei Aufruf im auftragnehmenden Rechenprozeß das letzte gültige Auftragskennzeichen beim Organisator erfragt und so im Rechenprozeß zur Verfügung stellt : Es entsteht letztlich wiederum die Gefahr des Auftragverlustes.

So führt die Analyse zu dem Schluß, daß der vom Organisator ermittelte Auftraggeber bzw. dessen Ursprungsziffer nicht statisch abgelegt werden darf, sondern daß diese Information den Auftrag an einen Rechenprozeß als eine Art Parameter oder Botschaft begleiten muß. Es werden Hilfsmittel und Einrichtungen neuer, bis jetzt in PEARL noch nicht bekannter Art benötigt, die die Organisation bzw. Steuerung der Informationsweiterleitung mit Sprachmitteln ermöglichen.

6.3 Parallelität von Auftragsanfall und Auftragsabwicklung

Wie an den Beispielen der Bilder 6.3 und 6.4 erläutert, können die Ereignisse nun schnell hintereinander oder gar überlappend eintreten. Wenn viele Bedingungen disjunktiv mit einem Rechenprozeß verknüpft sind, so können parallel zur Abarbeitung eines einzelnen Auftrages durch einen zugeordneten Rechenprozeß - der Auftragsabwicklung - neue Ereignisse eintreten, d.h. neue Aufträge entstehen. Gerade mit diesem Fall muß beim M-Beobachter-Problem gerechnet werden: Durch die unregelmäßigen Bewegungen vieler Stücke ist der Anfall an Aufträgen nur schwer oder gar nicht direkt beeinflußbar.

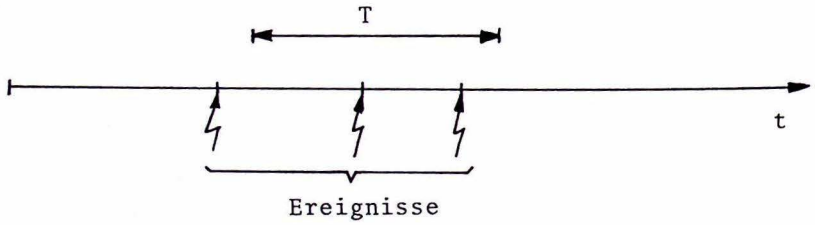


Bild 6.5 : Paralleler Auftragsanfall (Stückbewegungen) während der Auftragsabwicklung (T = Mindest-Abwicklungszeit)

Operationen, die Tätigkeiten unterbinden, tragen nur auf Umwegen zur Lösung des M-Beobachter-Problems bei. Deshalb werden künftig nur tätigkeitsauslösende Aufträge und deren Abwicklung behandelt.

Zwei Arten der Abarbeitung von parallel zur Abwicklung anfallender ("paralleler") Aufträge sind denkbar, je nach dem, ob eine Task reentrantfähig ist oder nicht.

6.3.1 Auftagsabwicklung mit nicht-reentrantfähigen Tasks

Eine nicht-reentrantfähige Task kann vor ihrer ordnungsgemäßen Abwicklung nicht von neuem begonnen werden. Wenn also mehrere Aufträge für eine solche Task gleichzeitig bestehen, kann im Fall von Nicht-Reentrance nur immer einer nach dem anderen bearbeitet werden.

So ergibt sich zwangsläufig eine Sequentialisierung. Die Anweisungen in einer Task laufen ständig - in Sequenz - nacheinander ab, innerhalb der Task gibt es so keine Probleme bezüglich Interaktionen.

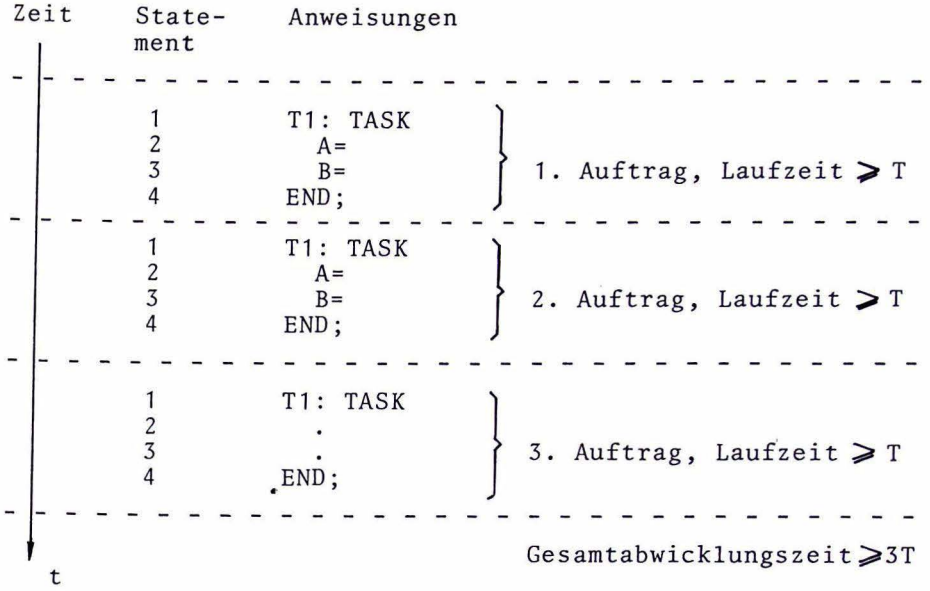


Bild 6.6 : Abwicklung dreier parallel angefallener Aufträge mit nicht-reentrantfähiger Task

Bild 6.6 zeigt ein Beispiel mit dem Auftragsanfall an eine Task T1 nach Bild 6.5. Die Gesamtabwicklungszeit beträgt dann mindestens 3 * T, bei N Aufträgen allgemein N * T.

Aufträge müssen also aufbewahrt, d.h. gepuffert werden, und zwar so lange, bis alle vorangegangenen Aufträge vom Auftragnehmer abgewickelt sind.

6.3.2 Auftragsabwicklung mit reentrantfähigen Tasks

Eine reentrantfähige Task könnte unabhängig von der ordnungsgemäßen Beendigung beliebig oft abgewickelt werden. Die Wirkung ist so, als wäre die Task (beliebig) oft vorhanden, d.h., als gäbe es (beliebig) viele "Kopien".

An den Ablauf dieser Mehrfachbenutzung gibt es keine Einschränkungen ; es ist sogar möglich, daß ein später begonnener Rechenprozeß einen früher begonnenen - wie etwa im Bild 6.7 - überholt.

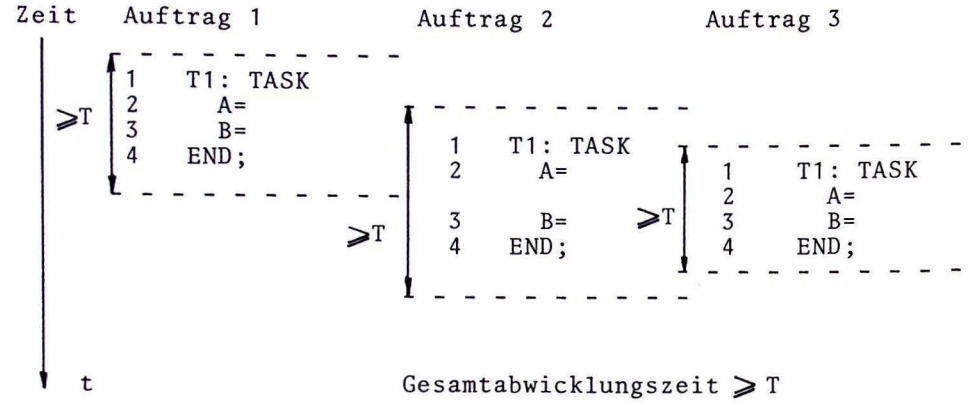


Bild 6.7 : Abwicklung dreier, parallel anfallender Aufträge mit reentrantfähiger Task

Eine schematische Gegenüberstellung der Auftragsabwicklung mit reentrantfähigen bzw. nicht-reentrantfähigen Tasks zeigt Bild 6.8.

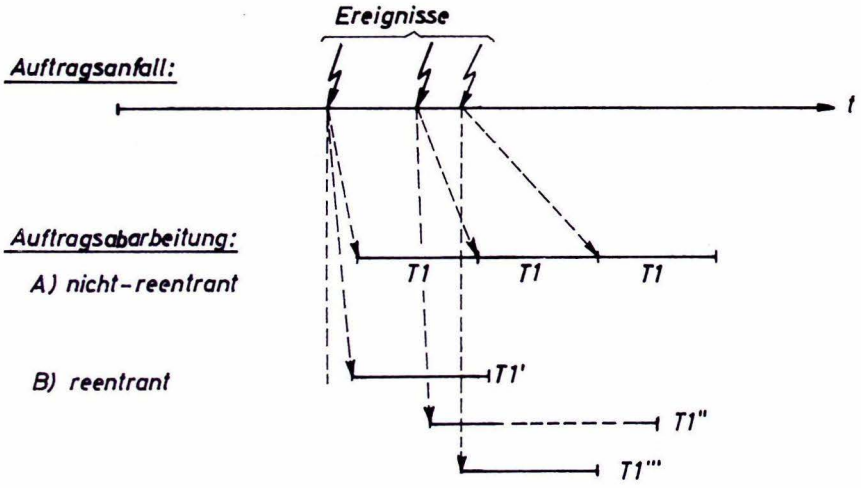


Bild 6.8 : Schematische Gegenüberstellung der Abarbeitung mehrerer Aufträge mit reentrant- und nicht-reentrantfähigen Tasks

Als verlockend muß erscheinen, daß jedem Auftrag eine solche "Kopie" zuteilbar ist : die Abwicklung aller Aufträge wäre unter diesem Gesichtspunkt schneller möglich, die Gesamtabwicklungszeit für N Aufträge könnte kleiner als N * T sein.

Es ist klar, daß durch die Parallelität der Rechenprozesse - durch die gleichen Segmente, d.h. Codestücke, quasi zu sich selbst - auch eine Konkurrenzsituation entsteht, die gemäß den Synchronisationsregeln /6.9, 6.10/ zu steuern ist.

Die einzige, nicht explizit zu synchronisierende Verbindung zur Umgebung des Rechenprozesses könnte bei Reentrance-Fähigkeit nur über die - implizit per Organisator synchronisierte - Parameterübergabe installiert werden. Existiert - wie bis dato bei PEARL - solch ein Parameterübergabemechanismus für Rechenprozesse nicht, dann löst Reentrancefähigkeit das Problem nicht, denn :

- Entweder der Rechenprozeß benötigt und hat auch Verbindungen zu seiner Umwelt, d.h. er benutzt - neben dem Prozessor - irgendwelche rechenprozeßexterne, nicht ausschließlich ihm zugehörige Betriebsmittel (hier das Zustandsabbild des Stückprozesses). Dann muß der Zugriff auf dieses Betriebsmittel synchronisiert werden, weil jeder Rechenprozeß auf dasselbe, externe Betriebsmittel zugreift.

  Damit ist Reentrance-Fähigkeit - von Sonderfällen wie z.B. in /6.11/ abgesehen, wenn die Zugriffsrechte durch sehr einfache Absprachen zu regeln sind - kein Mittel zur Lösung des Problems, weil die Forderung nach wechselseitigem Ausschluß die Sequentialisierung erzwingt.

- Oder der Rechenprozeß benötigt und hat keine Verbindung zu seiner Umwelt. Dann wird er auch keine bleibende Wirkung bezüglich seiner Umwelt erzielen (außer etwa im Verbrauch von Prozessorzeit); was im Rechenprozeß auch immer getan wird - niemand würde es je erfahren.

Damit scheidet der Einsatz von reentrantfähigen Tasks zur Abwicklung parallel anfallender Aufträge im Stückprozeß i.a. aus.

6.4 Mechanismen zur Bindung von Auftraggeber und Auftragnehmer

Wenn man nur nicht-reentrantfähige Tasks zur Abarbeitung der anfallenden Aufträge einsetzen kann, so benötigt man eine Möglichkeit, diejenigen zwischenzuspeichern, die nicht unmittelbar abgewickelt werden können.

Dies erfordert also die Zwischenschaltung eines Bindegliedes, das in der Lage ist

- Aufträge von einem Auftraggeber entgegenzunehmen,
- Aufträge zu speichern und
- Aufträge an den Auftragnehmer abzugeben.

Im folgenden werden Arten, Eigenschaften und Strukturen solcher Bindeglieder untersucht. Mit diesen Bindegliedern entstehen so Mechanismen zur Auftragsbindung, d.h. Modelle zur Auftragsbearbeitung.

Weil tätigkeitshemmende Aufträge nur auf Umwegen zur Lösung einer Aufgabe beitragen, werden die tätigkeitsauslösenden Aufträge bevorzugt behandelt. (Die entwickelten Bindeglieder wären gleichwohl für alle Auftragsarten geeignet.)

6.4.1 Der Lebenslauf eines (tätigkeitsauslösenden) Auftrages

Bevor man Bindeglieder zwischen Auftraggeber und Auftragnehmer näher untersuchen kann, muß erst bestimmt werden, wie die Abfolge der einzelnen Aktionen vom Anfall eines Auftrages, dem Auftragsbeginn, bis hin zu seiner Erledigung - der Auftragsabwicklung - sinnvollerweise sein soll. Diese Abfolge sei als "Lebenslauf eines Auftrages" in vier Abschnitten oder Phasen wie folgt definiert :

Phase 1 : Ereignisphase (Geburt)
Das freie Ereignis tritt ein. Das Eintreten stellt den Auftragsbeginn, die Geburt eines Auftrages dar.

Phase 2 : Notierungsphase (Registrierung)
Nachdem das Ereignis eingetreten ist, wird der zugehörige Auftrag notiert. Diese Registrierung ist nötig, weil die Abarbeitung des Auftrages nicht unmittelbar nach Ereigniseintritt, sondern erst mit - unbestimmter - Verzögerung begonnen werden kann.

Phase 3 : Initiierungsphase (Zustandswechsel)
Nach Ereigniseintritt und Notierung müssen - irgendwann - Maßnahmen getroffen werden, um den Zielrechenprozeß (den Auftragnehmer) in den gewünschten Zustand überzuführen (wenn Auftragsinhalt und Rechenprozeßzustand adäquat sind). Von besonderem Interesse für den Lebenslauf des Auftrages ist hier die Maßnahme des Löschens oder Streichens einer Notierung. Sie werde in der Initiierungsphase vorgenommen.

Phase 4 : Die Bearbeitungsphase (Abwicklung)
Das letzte Glied in der Auftragsbearbeitungskette ist die eigentliche, durch den Eintritt des Ereignisses auszulösende Auftragsabwicklung durch einen Rechenprozeß. Erst dann, wenn alle vorgesehenen Aktionen des Rechenprozesses beendet sind, kann die Auftragsleistung als ordnungsgemäß und vollständig erbracht gelten, die durch einen tätigkeitsauslösenden Auftrag beabsichtigt war.

Es ist sinnvoll, die Phase 4 zum Lebenslauf eines Auftrages zu zählen, obwohl das vordergründige Auftragsziel "Zustandswechsel" immer bereits in Phase 3 erreicht wird.

Bei tätigkeitshemmenden Aufträgen gibt es tatsächlich nur dieses Ziel, die Phase 4 existiert nicht. Bei tätigkeitsauslösenden Aufträgen jedoch besteht die Quintessenz der gesamten Auftragserteilung natürlich in der Abwicklung, im Erbringen einer bestimmten (Datenverarbeitungs-) Leistung, nicht im bloßen Zustandswechsel.

Die hier definierten Phasen sind Abschnitte, die Ereignisse oder Tätigkeiten voraussehen. Die zugehörigen, zwischen den Aktivitäten eingebetteten Zustände sind im Bild 6.9 dargestellt.

6.4.2 Bindeglieder

Im folgenden werden die elementaren Modelle zur Bindung von Auftraggeber und Auftragnehmer entwickelt. Ihre Beurteilung richtet sich dabei danach, welche Möglichkeiten sie der Lösung des M-Beobachter-Problems in Stückprozessen bieten.

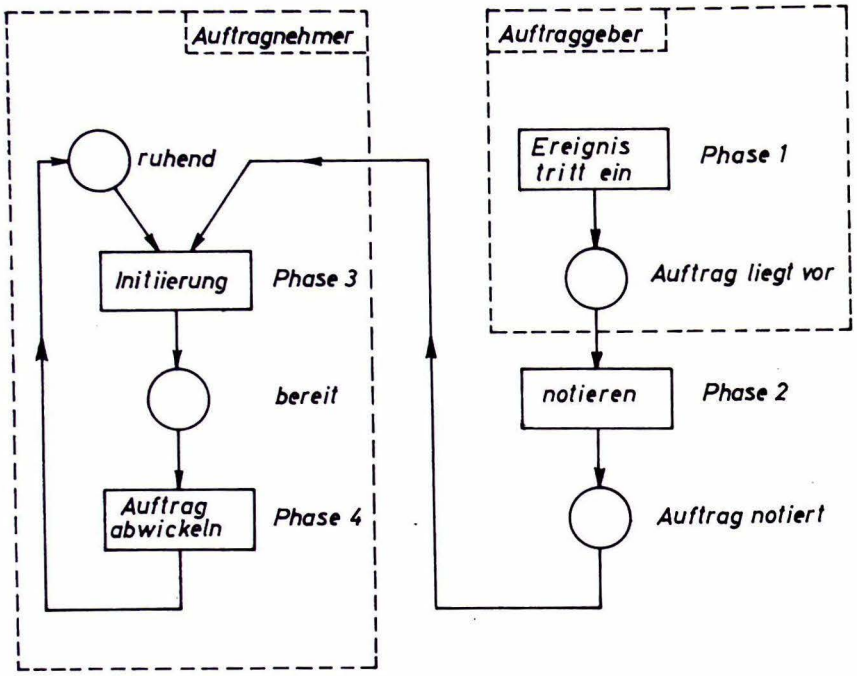


Bild 6.9 : Darstellung des Lebenslaufs eines Aktivierungs-Auftrages anhand eines Petri-Netzes

Das Flaggenmodell

Das sicherlich einfachste und elementarste Bindeglied ist eine binäre Variable, eine Flagge. Mit ihrer Hilfe kann notiert werden, ob ein Auftrag für einen Rechenprozeß vorliegt. Der Lebenslauf eines einzelnen Auftrages an einen bestimmten Rechenprozeß stellt sich mit diesem Bindeglied wie folgt dar :

- Phase 1 : Ereignisphase
  Das Ereignis tritt ein.
- Phase 2 : Notierungsphase
  Notieren des Auftrages, d.h. Setzen der Flagge
- Phase 3 : Initiierungsphase
  U.a. : Steichen des Auftrages, d.h. Rücksetzen der Flagge
- Phase 4 : Bearbeitungsphase
  Auftragsabwicklung

Der Vorteil des Flaggenmodells liegt auf der Hand : Die Notierung von verschiedenen Aufträgen für verschiedene Rechenprozesse ist mit einfachsten algorithmischen Hilfsmitteln möglich - mit Listen von Binärvariablen. Die Implementierung ist so sehr einfach.

Der erste Nachteil dieses Verfahrens ist ebenso evident : Von den während der Auftragsabwicklung entstehenden, neuen Aufträgen für denselben Rechenprozeß kann nur ein einziger gespeichert werden, alle anderen gehen verloren. Dasselbe gilt, wenn mehrere Aufträge entstehen, bevor eine Auftragsabwicklung zustande kommt.

In manchen Anwendungsfällen mag dies unerheblich und sogar sinnvoll sein. Entweder man arbeitet dann bewußt mit dem Verlust von Ereignissen bzw. Aufträgen. Oder man ist anhand des technischen Prozesses, dem das Ereignis entspringt, in der Lage zu garantieren, daß weder Mehrfach-Beauftragung noch Überlappung auftreten.

Für die Stückgutverfolgung mittels M-Beobachter jedenfalls ist das Flaggenmodell nicht brauchbar. Weder kann i. a. der Auftragsverlust toleriert, noch kann sichergestellt werden, daß Überlappung von Ereignis und Auftragsabarbeitung ausgeschlossen sind.

Der zweite, gravierende Nachteil: Woher der Auftrag stammt, ist nicht mehr feststellbar. Dies aber war eine der wesentlichsten Anforderungen bei der Stückverfolgung. Somit ist dieses Bindeglied "Flagge" zur Lösung dieses Problems nicht geeignet.

Anzumerken ist, daß das Streichendes Auftrages, d. h. Rücksetzen der Flagge nicht unbedingt in der Phase 3, sondern erst nach der Phase 4 erfolgen könnte. Dies hätte folgende Auswirkungen auf die Eigenschaften des Bindemechanismus :

a) Ein Auftrag kann gespeichert oder gepuffert werden, wenn er nicht parallel zur Auftragsabwicklung entsteht.

b) Jeder Auftrag geht verloren, der parallel zur Auftragsabwicklung entsteht.

Das Zählermodell

Zur nächsten Möglichkeit der Speicherung von Aufträgen gelangt man durch Übergang von der Zweiwertigkeit zur Mehrwertigkeit. So läßt sich mit Hilfe eines einfachen Zählers die Anzahl der angefallenen Aufträge aufsummieren. Die Existenz eines Auftrages für einen Auftragnehmer ergibt sich dann daraus, daß dieser (Auftrags-) Zähler z.B. eine Zahl>0 zum Inhalt hat.

Der Lebenslauf eines Auftrages stellt sich dann wie folgt dar :

- Phase 1 : Ereignisphase
- Phase 2 : Notierung
  Notieren des Auftrages durch Erhöhen des Auftragszählers um 1.
- Phase 3 : Initiierung
  U.a. Streichen des Auftrages durch Erniedrigen des Auftragszählers um 1.
- Phase 4 : Abwicklung

Ähnlich wie beim Flaggenmodell ist die Implementation sehr einfach ; je Auftragnehmer, d.h. je Rechenprozeß muß eine Zählvariable eingerichtet werden. Im Rahmen der physikalischen Möglichkeiten der Ganzzahldarstellung gehen nun keine Aufträge mehr verloren.

Wenn die Aufträge nicht nach ihrer Herkunft unterschieden werden müssen, ist dieses Modell zur Zwischenpufferung von M Aufträgen - ob vor oder parallel zur Abwicklung entstehend - völlig hinreichend.

Im beauftragten Rechenprozeß kann anhand des Zählerstandes eine gewisse Information über den "Stand der Arbeiten" gewonnen werden, wenn der Zählerinhalt in der Phase 3, der Initiierungsphase kopiert wird, und so dem Rechenprozeß während der Auftragsabwicklung zur Verfügung steht.[+)] Anwendungsfälle sind denkbar, in denen die Kenntnis des Zählerstandes von Nutzen ist, besonders, wenn die Aufträge immer vom selben Auftraggeber stammen.

[+)] Die Übergabe wird zweckmässigerweise - zwangsweise synchronisiert - in der Zustandswechselphase erfolgen, um mögliche Verklemmungssituationen beim Zugriff auf das Betriebsmittel "Auftragszähler" auszuschließen (siehe 6.4.3).

Natürlich kann im beauftragten Rechenprozeß immer noch nicht erkannt werden, welcher der Auftraggeber des gerade abzuwickelnden Auftrages ist.

Das Nachrichten-Puffermodell

Die wohl allgemeinste Verbindungsmöglichkeit ergibt sich durch die Einführung eines Nachrichten-Puffers, der in der Lage ist, Informationen oder Nachrichten aufzunehmen, aufzubewahren und wieder abzugeben. Die Nachricht kann dabei beliebig strukturiert sein.

Jeder Auftrag an einen Rechenprozeß ist - in Form einer Nachricht - in diesen Puffer einzubringen, wo er bis zur Entnahme aufbewahrt wird. Die Bindung des Auftragnehmers an den Puffer besteht in der Verpflichtung des Auftragnehmers, alle Nachrichten so rasch wie möglich dem Puffer zu entnehmen und je Nachricht einen Auftrag abzuwickeln.

Neben der Primärinformation, dem Auftragsinhalt, kann jede beliebige Sekundär-Information übermittelt werden, der Auftragnehmer erhält so (zusätzliche) Nachrichten vom Auftraggeber.

Der Lebenslauf eines Auftrages stellt sich dann folgendermaßen dar :

- Phase 1 : Ereignisphase
- Phase 2 : Notierungsphase
  Einbringen des Nachrichten-Elements E_i in den Puffer.
- Phase 3 : Initiierungsphase
  U. a. : Herausnehmen des Elements E_i aus dem Puffer und Übergabe des Elements an den beauftragten Rechenprozeß.
- Phase 4 : Abwicklung des Rechenprozesses

Mit diesem Nachrichtenpuffer als Bindeglied ist das M-Beobachter-Problem erstmals lösbar :

- Ein Auftrag wird weitergeleitet, indem die durch Abzählverfahren nach 6.2.2 zugeordnete Auftraggeberidentität in den Puffer eingebracht wird.
- Die Auftraggeberidentität wird dem Puffer vor der Auftragsabwicklung entnommen, so daß sie bei der Auftragsabwicklung verfügbar ist.

War beim einfachen Flaggenmodell und beim Zählermodell noch strittig, ob Rücksetzen der Flagge bzw. Zurückzählen vor oder nach Phase 4 zu geschehen habe, so ist dies beim Puffermodell eindeutig : Der Auftragnehmer benötigt die Information "Herkunft" während der Abwicklung und nicht erst danach, d.h., die Entnahme des Nachrichtenelements muß vor der Abwicklung erfolgen.

Zusammenfassung

Im Bild 6.10 sind alle Bindeglieder symbolisch zusammengefaßt, wobei unter der Phase 4 die Informationen dargestellt sind, die einem Rechenprozeß bei der Auftragsabwicklung über einen Auftrag zur Verfügung stehen.

Bindeglied	Phase 1 Ereignis	Phase 2 Notierung	Phase 3 Initiierung	Phase 4 Bearbeitung
		- Aktionen des Organisators -		Informationen bei Abwicklung
Flagge	↯	FLAG : = '1' B1	FLAG : = 'ϕ' B1	——
Zähler	↯	ZAEHLER : = ZAEHLER+1	ZAEHLER : = ZAEHLER-1	Zählerstand
Puffer	↯	E + Puffer	Puffer − E	jede beliebige

Bild 6.10 : Lebenslauf eines Auftrages und Auftragsbindeglieder

Anzumerken ist, daß es eine Art Hierarchie der Modelle gibt : Wenn alle Nachrichten im Puffer gleichen Inhalts sind, so erzielt man die Wirkung des Zählers ; wenn man zusätzlich die Pufferkapazität auf ein Element beschränkt, erzielt man die Wirkung der Flagge. (Zum Problem der Pufferkapazität siehe 7.5.1)

Das Puffermodell ist das universellste, so daß man sich damit begnügen könnte. Weil es unökonomisch ist, eine einfache Flagge oder einen Zähler mit Hilfe eines kompletten Puffermechanismus' zu substituieren, haben alle Bindeglieder ihre Existenzberechtigung.

6.4.3 Synchronisation und Auftragserteilung

Der Austausch von Nachrichten oder Botschaften über Puffer ist als eines der Synchronisationskonzepte zur Lösung gewisser Probleme in konkurrierenden Rechenprozessen bekannt. Es eignet sich generell zum Nachrichtenverkehr von Rechenprozessen untereinander. Auf Sprachebene setzt es u.a. sich zeitlich wechselseitig ausschließende Anweisungen zum Füllen und Leeren des Puffers voraus. Diese wirken blockierend, wenn sie nicht erfolgreich durchgeführt werden können.

Voraussetzung für ein verklemmungsfreies Funktionieren einer aufgebauten Kommunikation über den Nachrichtenpuffer ist dann natürlich, daß im nachrichtenverbrauchenden Rechenprozeß stets an die Stelle der Leerung zurückgekehrt wird, d.h., daß stets wie in einer Art Schleife ein Versuch zur Nachrichtenentnahme unternommen wird. Je nachdem, ob der Versuch erfolgreich war - es war eine Nachricht vorhanden - oder nicht - es war keine Nachricht vorhanden - wird im Rechenprozeß fortgefahren oder gewartet. (Die Wechselwirkung von Fortfahren und Warten ist das Grundprinzip aller Synchronisationsmechanismen.) Der Verbraucher muß so dauernd in den Zustand blockiert überführt werden.

Alle im Abschnitt 6.4.2 vorgestellten Mechanismen zur Weiterleitung von Aufträgen sind letztlich implizite Botschaftsmechanismen. Wenn man, wie vorgeschlagen, das Füllen des Puffers bei Ereigniseintritt durch den Organisator und das Leeren ebenfalls durch den Organisator an definierter Stelle vorschreibt, ist die Gefahr von Verklemmungen von vornherein geringer.

Läßt man die Entnahme von auftragsbegleitenden Nachrichten mit expliziten Anweisungen zu, so hätte der Programmierer dafür zu sorgen, daß im auftragnehmenden Rechenprozeß stets eine Entnahme explizit versucht wird. Da dies statisch nicht überprüfbar ist und somit die richtige Vorsorge nicht gewährleistet werden kann, ist die vorgeschlagene Lösung des impliziten Botschaftskonzepts unter dem Aspekt der erhöhten Zuverlässigkeit vorzuziehen.

6.5 Ein Lösungsvorschlag

Die vorangegangenen Ausführungen haben gezeigt, daß man einen Auftragsverkehr über einen Puffer auf jeden Fall benötigt, wenn der Auftraggeber nicht unmittelbar ein frei programmierbarer Anwender-Rechenprozeß ist, sondern implizit im Organisator verborgen ist.

Wenn man schon einen Puffermechanismus einführt, kann man diesen auch sehr vorteilhaft zum Nachrichtenverkehr von Rechenprozessen untereinander einsetzen, obwohl er dort nicht unbedingt benötigt wird. +)

Der folgende Lösungsvorschlag trägt so den Gesichtspunkten Rechnung.

a) Lösung des M-Beobachter-Problems mittels Puffer
b) Nachrichtenverkehr von Rechenprozessen untereinander
c) Einbettbarkeit in PEARL

Selbstverständlich ist eine begleitende Nachricht bei tätigkeitshemmenden Aufträgen nicht sinnvoll, weil der Rechenprozeß, dessen Tätigkeit auf Zeit oder auf Dauer unterbunden werden soll, diese ja nicht konsumieren kann.

Für die begleitende Nachricht beim Fortführungsauftrag gelten die Einwände des vorigen Abschnitts.

Aus diesen Gründen wird der tätigkeitsauslösende Auftrag der Aktivierung als am geeignetsten betrachtet, eine begleitende Nachricht zu übermitteln.

6.5.1 Einführung der Aktivierung mit Parameter-Übergabe

Die Aktivierungsanweisung als Teil der Zustandsanweisung (6.1) wird um die Parameterübergabe mit Ursprungsanweisung erweitert.

(6.2) aktivierungsanweisung :: =
[schedule]
ACTIVATE taskbezeichner [(taskparameterliste)]
[sonstiges] ;

+) Der Nachrichtenpuffer läßt sich konstruieren mittels einfacher Synchronisationshilfsmittel (z. B. Semaphore), die den wechselseitigen Ausschluß beim Pufferzugriff ermöglichen. Die Substitution ist jedoch sehr mühselig. /6.14/

Dabei sei

(6.2a) taskparameterliste :: = $\{, . \{ \text{expression\_seven} \mid \text{ORIGIN}\} \ldots\}$

Die Semantik dieser Anweisung sei :

- Ist eine "taskparameterliste" angegeben, so sind deren eventuell darin enthaltene Ausdrücke ("expression_seven") zum Zeitpunkt des Überlaufens der "aktivierungsanweisung" zu ermitteln. +)

  (Als "taskparameter" sind einschränkend gegenüber den aktuellen Parametern des Prozeduraufrufs nur Ausdrücke (bzw. die Hilfsgröße ORIGIN) zugelassen, weil diese ein Ergebnis liefern, das per Wert übergeben werden kann.)

- Bei bedingter Aktivierungsanweisung wird beim Ereigniseintritt - falls vorhanden - ein Satz von Werten kopiert und in den Nachrichtenpuffer eingebracht, der zum Zielrechenprozeß gehört. Dieser Parametersatz wird so lange im Puffer aufbewahrt, bis der mit dem Auftrag beabsichtigte Zustandswechsel des Zielrechenprozesses stattfindet. Bei Zustandswechsel wird der Parametersatz entnommen (siehe 6.5.2).

- Bei unbedingter Aktivierungsanweisung wird der Parametersatz nicht zwischengepuffert, sondern unmittelbar mit dem Zustandswechsel an den Zielrechenprozeß übergeben. Dies entspricht der Festlegung in /6.3/.

- Die bedingte Aktivierungsanweisung kann als möglichen Taskparameter die Hilfsvariable mit der Bezeichnung "ORIGIN" enthalten. Sie ist vom Typ Ganzzahl und dient als eine Art von formalem Parameter. An dieser Position wird bei Eintreten eines der Ereignisse im "schedule" die Auftraggeber-Identität vom Organisator eingesetzt, die bei der Einplanung nach dem Abzählverfahren nach 6.2.2 zugeordnet wurde, und mit allen anderen Parametern in den Puffer eingebracht.

+) Um Gültigkeits- und Laufzeitprobleme (Ausdrucksauswertung bei Ereigniseintritt) aus dem Wege zu gehen, ist diese Lösung vorzuziehen.

Hierzu einige Beispiele :

Beispiel 6.5 : Bedingte Aktivierung mit Ursprungsanweisung

```
A : TASK;
:
X = 5;
WHEN ITR (7 : 15) ACTIVATE UVW (X, ORIGIN) ;
:
END; /*  ENDE TASK A  */
```

Tritt nach Einplanung der Aktivierung in Task A z.B. das Ereignis ITR(8) ein, so wird der Parametersatz (5, 2) zum Konsum durch UVW gepuffert. Anhand des 2. Parameters kann dann in UVW der Auftraggeber identifiziert werden.

Beispiel 6.6 : Bedingte Aktivierung benutzt zur Kommunikation von Rechenprozessen mittels Nachrichten-Puffer

```
B : TASK;
:
FOR I FROM 1 BY 2 TO X + 5 REPEAT
  AFTER 0 SEC ACTIVATE RST (I, X+I);
END;
:
END; /*  ENDE TASK B  */
```

Nach Ablauf von 0 sec, d.h. so bald als möglich, soll die Task RST aktiviert werden. Jede Aktivierung enthält einen Parametersatz (I, X + I), entwickelt zum Zeitpunkt des Überlaufens der Einplanung. Das Beispiel enthält keinen Ursprungsaufruf, die Parametrisierung dient also ausschließlich zur Kommmunikation der Task B mit der Task RST über Puffer.

6.5.2 Einführung der formalen Parameterspezifikation, der Pufferart und der Pufferanzahl in der Task-Deklaration

Natürlich ist es ausschließlich vom zu lösenden Problem abhängig, ob man Aufträge mittels Nachrichtenpuffer oder nur etwa mit einer Flagge weiterreichen kann. Auf Sprachebene muß also der Auftragsmechanismus für jeden Rechenprozeß individuell steuerbar gestaltet werden, man darf ihn sicher nicht der Implementation überlassen, wie dies in /6.3/ vorgeschlagen wurde.

Zweckmäßigerweise stellt man sich den Puffer als eine Art von Wartespeicher für Aufträge vor, die vor der Bedieneinheit "Task" zur Abfertigung aufgebaut ist und die nur zu diesem gehört.

Gründe der effektiven Auslastung sprechen zwar gegen einen individuellen Auftragsstauraum für jedes Programm /6.12/. Wie aber in Kapitel 7 gezeigt werden wird, ist dieser in Form des Nachrichtenpuffers fester Länge zur Lösung des M-Beobachter-Problems bei der Stückverfolgung notwendig und hinreichend.

Wenn man den zu übermittelnden, primären Auftragsinhalt wie in 6.5.1 auf die Aktivierung beschränkt, bietet sich folgende Lösung an :

(6.3) task_deklaration :: =

taskbezeichner : TASK

[{ ({, . bezeichner ...}) BUFFER (ganzzahlkonstante) }]
[{ COUNT (ganzzahlkonstante) }]

[sonstiges] ;

Die Bedeutung dieser Anweisung sei :

- Die Regeln für die Beschreibung formaler Parameter ("bezeichner") in einer Prozedur gelten auch für die Beschreibung der Taskparameter. Damit kann, ähnlich wie beim Prozedurmechanismus, vom Compiler überprüft werden, ob Parameterzahl und Typ der in der Aktivierungsanweisung nach (6.2) angegebenen Parameter mit der Taskparameterspezifikation übereinstimmen.

- Bei Zustandswechsel "ruhend - bereit", der Exekution der Aktivierung, wird

  -- ein Satz von Parametern aus dem Puffer entnommen, falls spezifiziert,

  -- der Zähler um 1 erniedrigt, falls COUNT spezifiziert ist,

  -- die Flagge niedergeholt, falls weder Parameter mit Pufferzahl noch Zähler angegeben ist.

- Die BUFFER-Option bestimmt mittels Ganzzahlkonstante die Zahl der zur Verfügung zu stellenden Nachrichtenpufferplätze ; die COUNT-Option bestimmt, wieviele Aufträge gezählt werden sollen (Zählermodell). Fehlt die Option völlig, sei das Flaggenmodell als Default impliziert.

Auch hierzu einige Beispiele :

Beispiel 6.7 : Task mit Parameterspezifikation

```
T1 : TASK (U, V, X) BUFFER 20 PRIORITY 10 ;
     SPC U FLOAT, V BIT(5), X FIXED ;
     :
END ; /*  ENDE T1  */
```

Bei jeder Aktivierung wird vom Organisator ein Parametersatz (U, V, X) zur Verfügung gestellt. Zur Überprüfbarkeit ist der Typ der Variablen zu spezifizieren ("SPC"). Insgesamt 20 Parametersätze, d.h. 20 Aktivierungsaufträge mit Begleitnachrichten können gepuffert werden.

Beispiel 6.8 : Eine mögliche Lösung des M-Beobachter-Problems anhand des Modellprozesses

```
T2 : TASK ;
     WHEN ITR (1 : 91) ACTIVATE T3 (ORIGIN) ;
END ; /*  ENDE T2  */

T3 : TASK(Z) BUFFER 91 ;
     SPC Z FIXED ;
     CALL ALGORITHMUS (Z) ;
     :
END ; /*  ENDE T3  */
```

In der Task T2 wird in der Einplanungsanweisung die Interruptscheibe ITR(1 : 91) zur Aktivierung der Task T3 verwendet. Der Ursprung wird als auftragsbegleitende Nachricht bei Eintritt eines der 91 Ereignisse an T3 weitergereicht.

Wie im Beispiel 6.8 gezeigt, ist durch Kombination der vorgeschlagenen Sprachanweisungen erstmals die Lösung des M-Beobachter-Problems möglich : Bei der Einplanung wird das Weiterleiten des Auftraggebers verlangt, das dann im Zielrechenprozeß als Taskparameter zur Verfügung steht. (Über die Zahl der benötigten Pufferplätze siehe Kapitel 7.)

6.6 Zusammenfassung und Kritik

Der vorgeschlagene Mechanismus erfüllt folgende funktionellen Anforderungen im Zusammenhang mit dem Problem der Stückgutverfolgung :

1) Die Zahl der zwischenspeicherbaren Aufträge muß auf Sprachebene steuerbar sein. Sie ist einzig und allein vom Problem abhängig, sie darf höchstens in der aktuellen Implementation begrenzt, nicht aber generell implementationsabhängig und schon gar nicht auf Sprachebene begrenzt sein.

2) Sind mehrere Ereignisse disjunktiv auftragsauslösend für einen Rechenprozeß, so genügt in manchen Fällen - etwa bei der Stückgutverfolgung - die Speicherung von Aufträgen anhand eines einfachen Zählers nicht, weil im beauftragten Rechenprozeß der Auftraggeber nicht erkannt werden kann. Vielmehr ist eine Nachricht, ein Auftragskennzeichen zu übermitteln.

3) Sind mehrere Ereignisse disjunktiv auftragsauslösend, so muß auf Sprachebene Struktur und Zahl der Nachrichtenpufferplätze für die auftragsbegleitenden Nachrichten oder Parameter festlegbar sein.

4) Ob zur Zwischenspeicherung von momentan nicht sofort abzuwickelnden Aufträgen per Flagge, Zähler oder Nachrichtenpuffer zu erfolgen hat, muß auf Sprachebene steuerbar sein.

Die Einführung eines expliziten Nachrichtenkonzepts stellt den vorgegebenen Tasking-Mechanismus von PEARL letztlich von Grund auf in Frage. Zur Lösung des M-Beobachter-Problems kann man sich jedoch auf den vorgeschlagenen impliziten Nachrichtenmechanismus des Auftragsbegleitens beschränken. Der Programmierer ist nicht gezwungen, aus dem Puffer etwas zu entnehmen ; er kann sich aber auch nicht dagegen wehren, daß etwas bei Aktivierung entnommen wird. Die Verklemmungsgefahr ist damit verringert.

Der Datenverkehr von Rechenprozessen untereinander ist jedoch genau wie beim Nachrichtenkonzept sehr erleichtert, weil die Synchronisation, die sonst explizit beim Zugriff auf den selbst zu organisierenden Puffer vorzunehmen wäre, in der Aktivierungsanweisung implizit verborgen ist.

Weil es nicht sinnvoll erscheint, tätigkeitshemmenden Aufträgen begleitende Nachrichten mitzugeben, wurde die Nachrichtenübermittlung auf die Aktivierung beschränkt, d.h. die Aktivierung gegenüber den anderen Zustandsanweisungen priviliegiert.

7. Laufzeitbetrachtungen und Konkurrenzsituationen im Stückprozeß

Es ist charakteristisch für den Stückprozeß, daß viele gleichartige Vorgänge parallel anfallen. Dieses Kapitel stellt die hieraus resultierenden Konkurrenzsituationen anhand des M-Beobachter-Modells des Kapitels 3 und deren Auswirkungen dar.

7.1 Einzelbeobachtung und die elementaren Laufzeitgesetze

7.1.1 Definition des "worst-case"

Zur Stückgutverfolgung benötigte man Beobachter, deren Aufgabe darin bestand, den Belegungszustand gewisser Abschnitte zu erfassen.

Nun muß angenommen werden, daß für jedes einzelne Stück, das sich gerade im Abschnitt i bewegt, noch vor Verlassen des Abschnittes ein oder mehrere, dieses individuelle Stück betreffende Stellbefehle zu geben sind (z. B. ein "Halt"-Befehl, weil der vorausliegende Abschnitt j besetzt ist). +)

Für die im Folgenden vorgenommene "worst-case"-Betrachtung wird weiterhin angenommen, daß jedes beliebige Stück auf einem Abschnitt b_i immer eine bestimmte Mindest-Stückdurchlaufzeit t_{li} benötigt, um den Abschnitt b_i von der Beobachtungsstelle bis zum Ende des Abschnitts zu durchfahren ++).

Die Task, die die Steuerung eines Stückes s_j im Abschnitt b_i vornehmen soll, benötigt von ihrer Initiierung (Zustand "laufend") bis hin zur Ausgabe des Stellbefehls eine bestimmte Abwicklungszeit. Diese Abwicklungszeit wird jedenfalls von der Stückidentität abhängig sein, weil die Abwicklung u. a. Suchalgorithmen zu der Führung des Prozeßmodells enthalten wird, deren Abwicklungszeit nicht konstant ist, sondern vom Suchargument abhängt. Um diesen Einfluß zu eliminieren, wird deshalb an dieser Stelle die Abwicklungszeitdauer t'_{ai} angesetzt, die bei ungünstigster Konfiguration entsteht.

+) Dürfte ein identifizierbares Stück einen Abschnitt betreten, ohne daß ein von diesem individuellen, gerade beobachteten Stück abhängender Stellbefehl zu geben ist, dann dürfte dies auch jedes beliebige andere Stück. D. h., ein Beobachter in diesem Abschnitt ist funktionell für die Lenkung der einzelnen Stücke überflüssig.

++) Für die Durchlaufzeiten eines Abschnittes können eventuell bestimmte Wahrscheinlichkeitsverteilungen angegeben werden. Ausschlaggebend ist hier jedoch, daß es eine Schwelle t_{li} für jeden Abschnitt gibt, die aus physikalischen Gründen nicht unterschritten werden kann.

In konkurrierender Umgebung könnte natürlich die Abwicklung jedes Auftrages beliebig verzögert werden, weil möglicherweise nicht alle Betriebsmittel zur Verfügung stehen. Diese Verzögerungszeit kann natürlich nicht in t'_{ai} enthalten sein, weil sie unbekannt ist ; t'_{ai} ist so die Abwicklungszeit, die in nicht-konkurrierender Umgebung für die in Bezug auf die Abwicklungszeit ungünstigste Beobachtung aus physikalischen Gründen nicht unterschritten werden kann.

Genauso ist die Zeitspanne, die von Auftragsgeburt bis zur Initiierung vergeht, nicht bestimmbar. Sie kann jedoch auch unter günstigsten Umständen aus physikalischen Gründen auch nicht unter eine bestimmte Mindestzeitdauer t_{gi} verbracht werden.

Die Gesamt-Abwicklungszeitspanne - von Geburt des Auftrages bis zu seiner erfolgreichen, planmäßigen Beendigung, mit der unter den geschilderten Umständen zu rechnen ist, beträgt so für die Beobachtung im Abschnitt i :

(7.1) $$t_{ai} = t'_{ai} + t_{gi}$$

7.1.2 Die elementaren Laufzeitgesetze

Damit ein Stellbefehl für ein bestimmtes Stück im entsprechenden Abschnitt noch rechtzeitig eintrifft, darf die Gesamt-Abwicklungszeit einer jeden Beobachtung höchstens gleich der Stückdurchlaufzeit sein. Über alle M Beobachtungsabschnitte muß deshalb gelten :

(7.2a)
$$\begin{aligned} t_{a1} &= t_a(B_1) \leqslant t_{l1} \\ t_{a2} &= t_a(B_2) \leqslant t_{l2} \\ &\vdots \\ t_{aM} &= t_a(B_M) \leqslant t_{lM} \end{aligned}$$

1. Laufzeitgesetz : Die nicht unterschreitbare Gesamtzeitspanne t_{ai} zur Abwicklung aller Aktionen bei der Lenkung eines Stückes in einem bestimmten Abschnitt b_i - von der Beobachtung bis zur Ausgabe des Stellbefehls - muß kleiner oder gleich der Stückdurchlaufzeit t_{li} im Beobachtungsabschnitt b_i sein :

(7.2b) $$t_{ai} = t_a(B_i) \leqslant t_{li} \qquad i = 1, 2, \ldots M$$

Wird das Laufzeitgesetz verletzt, so ist mit fehlerhaftem Materialfluß zu rechnen, weil das individuell zu lenkende Stück seinen zugehörigen, individuellen Stellbefehl zeitlich überholt. Die daraus resultierende Fehllenkung von Stücken ist zumindest unerwünscht, in extremen Fällen eventuell sogar gefährlich.

Die einzig wirksame Abhilfe besteht im Aufbau von Hardware-Anordnungen, die das Verlassen des Abschnittes nur auf ausdrücklichen Befehl gestatten, sofern dies möglich ist. Mit solch einer Art von Sicherungs-Anordnung verhindert man Fehllenkungen, nur würde durch den daraus resultierenden, ständigen Start-Stop-Betrieb der Durchsatz möglicherweise erheblich vermindert bzw. die Realisierung zu aufwendig.

Wie ausführlich dargestellt, wird man natürlich die Bearbeitung aller M Beobachtungen von derselben Instanz durchführen lassen. Dann muß im "worst-case" mit der größten Abwicklungszeit aus allen Abschnitten gerechnet werden. Dann gilt :

$$t_a = \max_{i,\,j = 1,\,2,\,\ldots\,M} \left[t_{ai} \right] = t_a(B_i) = t_a(B_j) = \text{const.}$$

Daraus resultiert eine Verschärfung des 1. Laufzeitgesetzes. Es gilt nämlich nach (7.2a)

$$\begin{aligned} t_{a1} &= t_a = \text{const} \leqslant t_{11} \\ t_{a2} &= t_a = \text{const} \leqslant t_{12} \\ &\vdots \\ t_{aM} &= t_a = \text{const} \leqslant t_{1M}, \end{aligned}$$

d.h., insgesamt muß die Abwicklungszeit t_a kleiner als die minimalste Stückdurchlaufzeit aus allen Abschnitten sein :

2. Laufzeitgesetz: Benutzt man bei der Abwicklung aller Aktionen zur Lenkung eines Stückes in allen Abschnitten dieselbe Instanz, so muß die nicht-unterschreitbare Abwicklungszeit t_a jeder beliebigen Beobachtung kleiner oder gleich der minimalsten Stückdurchlaufzeit t_{1min} - von Beobachtungsort bis Abschnittende - aus allen M Abschnitten sein :

(7.3) $t_a \leqslant t_{1\,min}$

mit $t_{1\,min} = \min \left[t_{li} \right]$ $\quad i = 1, 2, \ldots M$

(Streckenabhängige Abwicklungszeit ist ebenfalls denkbar ; dann müssen die Restriktionen des 1. Laufzeitgesetzes Anwendung finden.)

Im 2. Laufzeitgesetz stellt die kürzeste Durchlaufzeit - vereinfachend bei konstanter Bewegungsgeschwindigkeit die kürzeste Strecke - den Engpass bezüglich der Funktionsweise bzw. Funktionstüchtigkeit des automatisierten Stückprozesses dar.

Wenn die schärfste Restriktion des Gesetzes 1 bzw. das Gesetz 2 nicht eingehalten werden kann, müssen in den Stückprozeß Hardware-Sicherungen eingebaut werden, weil der technische Prozeß im Vergleich zum Rechenprozeß zu schnell ist.

7.2 Konkurrenz der Beobachter

Zur Herleitung der Konkurrenzsituationen dient das im Kapitel 3 eingeführte M-Beobachter-Modell.

Solange ein Rechenprozeß nur seine eigenen, ihm zugehörigen Hilfsquellen benutzt, entstehen bezüglich der Interferenz mit anderen, parallel hierzu ablaufenden Rechenprozessen keine grundsätzlichen Probleme.

Es handelt sich dann um nichtinteraktive, parallele Rechenprozesse, die vollständig voneinander getrennt sind. So sind die Tätigkeiten eines jeden der identifizierenden Beobachter "Lese Identität ab" echt parallele, voneinander völlig getrennte, lokale Aufgaben, die beispielsweise mit Hilfe je eines Rechners bewältigt werden könnten. Jeder der Beobachter kann aber nur so lange unabhängig vom anderen operieren, solange es gilt, das Ereignis der Ankunft eines Stückes lokal festzuhalten.

Um eine Übersicht über die gesamte Stückgutanordnung zu erhalten, mußten die lokalen Beobachtungen zentral gesammelt werden. Die hierfür benötigten Zeitspannen seien :

Tätigkeit	Zeit-spanne
1) Erkennen eines Stücks und notieren	T_B
2) Beobachtung an Zentrale abgeben	$T_Ü$

Da es M Beobachter gibt, und sich die Stücke im schlechtesten Fall völlig ungeordnet bewegen, könnten sich schlimmstenfalls alle M Beobachtertätigkeiten zeitlich wie im Bild 7.1 überlappen.

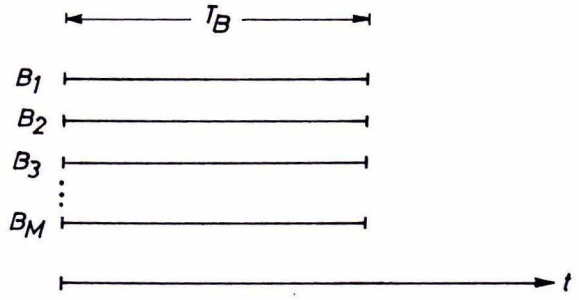


Bild 7.1 : Parallele Beobachtertätigkeiten

Bezüglich der Tätigkeit 1) gibt es trotz M parallel anfallender Beobachtungen keine grundsätzlichen Probleme, notfalls sind M Rechner oder Prozessoren einzusetzen, um wirkliche Parallelität zu erreichen.

Dagegen führt die Tätigkeit 2) "Gebe Beobachtung ab" zu einem Konkurrenzproblem. Es entsteht ein interaktiver Prozeß, weil sich alle Beobachter das Betriebs- oder Hilfsmittel "Oberbeobachter" teilen ; sie treten auf diesem Wege indirekt in Konkurrenz zueinander.

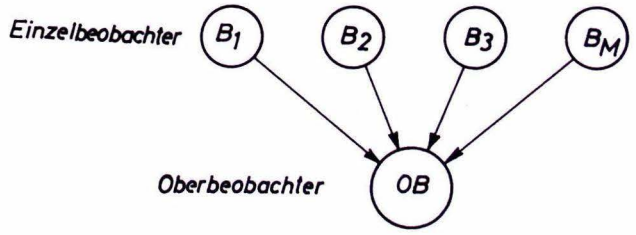


Bild 7.2 : Der Informationsfluß im Stückprozeß anhand des M-Beobachtermodells

Auf die Verhältnisse in Rechnern übertragen, entsprechen die Tätigkeiten der einzelnen Beobachter und die des Oberbeobachters Rechenprozessen, die folglich miteinander konkurrieren.

Teilen sich aber konkurrierende Rechenprozesse Betriebsmittel, so muß gewährleistet sein, daß

- die Betriebsmittel gegen ungewollte Interferenz durch andere Rechenprozesse geschützt sind, und
- die Ergebnisse jedes einzelnen Rechenprozesses unabhängig von der Geschwindigkeit des Rechenprozesses sind, da dieser keinen Einfluß auf seinen Fortschritt hat.

Andernfalls können zeitabhängige Fehler auftreten, obwohl jeder Rechenprozeß für sich alleine reproduzierbare und richtige Ergebnisse liefert, wenn er nicht in konkurrierender Umgebung ablaufen muß, d.h., daß interaktive Rechenprozesse bzw. deren interaktive Teile nicht parallel zueinander ablaufen dürfen. Sie müssen in ihren interaktiven Teilen synchronisiert und somit sequentialisiert werden /6.9/.

Durch die Verkettung der Beobachter mit einer Zentrale zwecks Kommunikation ist aber eine interaktive Anordnung entstanden, in der, da sich Parallelität und Interaktion ausschließen, der Zwang zur Synchronisation

- zwischen jedem Beobachter B_i und dem Oberbeobachter

oder indirekt

- zwischen jedem B_i und jedem B_j $\quad (i \neq j)$

besteht.

Fallen, wie im Bild 7.1, alle Beobachtungen zur selben Zeit an, so muß jede Übertragung der Meldung eines Beobachters gegen jede andere Übertragung verriegelt werden. Den zeitlichen Ablauf der Nachrichtensammlung zeigt schematisch Bild 7.3. (Als Reihenfolge für die Übernahme der Meldungen sei willkürlich 1, 2, ... M angenommen.)

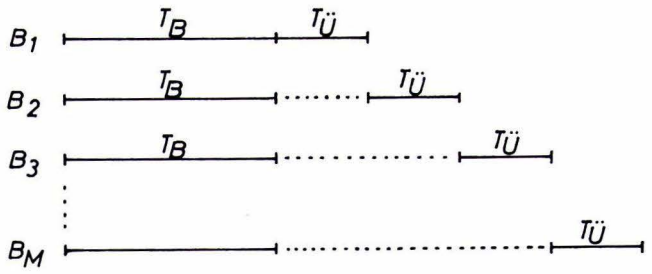


Bild 7.3 : Zentrale Sammlung aller parallelen Beobachtungen

Der Beobachter B_1 kann dann während der Zeit $T_ü$ keine weiteren Beobachtungen registrieren, genauso, wie der Oberbeobachter nicht in der Lage ist, eine andere Meldung entgegenzunehmen. Alle anderen Beobachter sind daher bezüglich der Übertragung blockiert. So ist im Bild 7.3 B_2 um die Zeit $T_ü$, B_3 um 2 $T_ü$ blockiert, bis B_2 bzw. B_3 ihre Meldungen abgeben können.

Bis alle Beobachter ihre Meldungen abgegeben haben, vergeht bei dieser "worst-case"-Betrachtung also die Zeit

$$M \cdot T_ü .$$

7.3 Laufzeitbetrachtungen über alle Abschnitte

7.3.1 Das 3. Laufzeitgesetz

Die bloße Übertragung der Beobachtungen an einen zentralen Oberbeobachter ist erst Voraussetzung für die zentrale Lenkung eines Stückprozesses. Diese Zentrale erhält eine Folge von Informationen, die zu bearbeiten sind : jede Beobachtung (eines Stückes) ist in die Prozeßmodelle, die Abbildungen des Stückprozesses einzutragen, so daß anhand der Abbilder und der zugehörigen Steueralgorithmen jedes Stück individuell gelenkt werden kann.

Weiterhin von der "worst-case"-Betrachtung ausgehend, sollen sich gleichzeitig auf allen M Strecken die Beobachtertätigkeiten überlappen, M verschiedene Stücke laufen also gleichzeitig an M Beobachtern vorbei.

Von allen M Abschnitten sei

- b_{min} der Abschnitt mit der minimalsten Stückdurchlaufzeit t_{1min} von Beobachter bis Abschnittsende,
- b_{max} der Abschnitt mit der maximalsten Stückdurchlaufzeit t_{1max} von Beobachter bis Abschnittsende.

Bezeichnet man die Zeitspanne, die im "worst-case" von Übernahme einer einzelnen Beobachtung von den dezentralen Beobachtern an die Zentrale bis zur Ausgabe des zugehörigen Stellbefehls nicht unterschritten werden kann, als die Verarbeitungszeit des zentralen Oberbeobachters $T_v = t'_{ai} + T_{ü}$, so kann es $M \cdot T_v$ dauern, bis eine erneute Beobachtung desselben Beobachters bearbeitet werden kann. D.h., jede Beobachtung ist in diesem schlechtesten Fall um $M \cdot T_v$ blockiert.

Erfolgt die Bearbeitung der Meldungen in willkürlicher Reihenfolge, so besteht zuerst für das Stück auf dem Abschnitt mit der kürzesten Stückdurchlaufzeit die Gefahr, daß der Stellbefehl für das gerade zu manipulierende Stück zu spät eintrifft. Zufällig könnte nämlich ausgerechnet die Bearbeitung der Beobachtung an der laufzeitmäßig kürzesten Strecke um $M \cdot T_v$ verzögert werden ; ist $M \cdot T_v$ aber größer als t_{1min} - der minimalsten Stückdurchlaufzeit von Beobachterposition bis zum Ende des Beobachtungsabschnittes - dann kommt der zugehörige Stellbefehl mit Sicherheit zu spät ; ohne besondere Vorkehrungen hat das zu lenkende Stück den Abschnitt vorher verlassen. Daraus resultiert das

3. Laufzeitgesetz

Im Stückprozeß mit M Beobachtern muß die Durchlaufzeit durch den laufzeitmäßig kürzesten Abschnitt b_{min} größer oder gleich sein als M mal die nicht-unterschreitbare Verarbeitungszeit T_v, die für eine Einzelbeobachtung von dem Beobachtungszeitpunkt bis zum Abschluß aller relevanten Aktionen benötigt wird :

$$(7.4) \qquad t_{l\,min} \geqslant M * T_v$$
$$t_{l\,min} = \min\left[t_{li}\right] = t_l\,(b_{min})$$

Ein einfaches Beispiel mit drei unterschiedlichen Stückdurchlaufzeiten in den Abschnitten 1 bis 3 und Einhaltung des 3. Laufzeitgesetzes zeigt Bild 7.4.

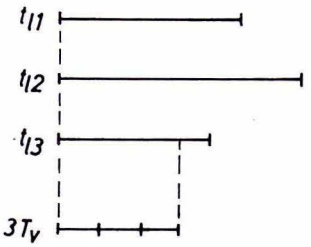


Bild 7.4 : Beispiel für die Einhaltung des 3. Laufzeitgesetzes bei drei Abschnitten (t_{11}, t_{12}, $t_{13} \geqslant 3\,T_v$)

Das 3. Laufzeitgesetz stellt eine erhebliche Verschärfung aller bisherigen Laufzeit-Randbedingungen dar, besonders für große M. +) Waren die bisherigen Laufzeitgesetze letztlich auf einzelne Abschnitte bezogen, so ergibt sich aus dem 3. Laufzeitgesetz eine Einschränkung für jeden Einzelabschnitt, die von allen Abschnitten und deren Zahl mit abhängig ist. Den Engpaß für die gesamte Stückgutanordnung bzw. dessen Funktionsfähigkeit stellt dabei die laufzeitmäßig kürzeste Strecke dar.

Dieser Engpaß bei der zentralen Erfassung und der zentralen Auswertung der Beobachtungen besteht völlig unabhängig davon, ob man einen oder mehrere Prozessoren oder Rechner zur Verfügung hat, und welche Strategie man zur Prozessorvergabe, Unterbrechungsbearbeitung, Ereignisverwaltung etc. in Rechnern anwendet. (Zur dezentralen Bearbeitung der Beobachtungen siehe 7.3.3)

+) Im realisierten Modellprozeß beträgt die kürzeste Stückdurchlaufzeit $t_{lmin} \approx 0,3$ sec. Bei 91 Beobachtern und im schlechtesten Fall bei 91 gleichzeitig bewegten Transporteinheiten bliebe zur Bearbeitung einer Beobachtung grob gerechnet nur 0,3 sec/91 $\approx$ 3 msec oder umgerechnet ca. 600 Instruktionen in der AEG 60-50, also bei weitem zu wenig /4.2/.

7.3.2 Die Position des Beobachters

In allen Laufzeitbetrachtungen spielt die Stückdurchlaufzeit t_{li} vom Beobachtungsort bis zum Abschnittsende eine entscheidende Rolle. Unter diesem Aspekt würde man versuchen, einen Beobachter, wenn irgend möglich, am Anfang eines Abschnittes zu postieren, damit möglichst viel Zeit zwischen den Zeitpunkten der Beobachtung, d. h. Erfassung und der Stellbefehlausgabe zur Verfügung bleibt.

Je nach physikalischen Gegebenheiten riskiert man damit jedoch, daß mehrere Stücke in den gleichen Beobachtungsabschnitt einfahren. In der seitherigen Betrachtung war stillschweigend vorausgesetzt worden, daß während der Zeit, in der sich ein Stück in einem bestimmten Beobachtungsabschnitt aufhält, nicht inzwischen ein anderes Stück denselben Bereich betritt. Dann war es auch gleichgültig, wann genau der Stellbefehl gegeben wurde : Die hergeleiteten Laufzeitgesetze sagen nur aus, bis wann er spätestens zu erfolgen habe. Und nur dann steht die Stückdurchlaufzeit in voller Länge zur Verfügung, eine einzelne Beobachtung darf maximal bis zu dieser Zeit blockiert werden.

Läßt man jedoch das Eintreten mehrerer Stücke in einem Beobachtungsbereich zu, so gewinnt man zwar mehr Zeit, um den Stellbefehl auszugeben, handelt sich aber zusätzliche Schwierigkeiten ein. Man benötigt jetzt nicht nur eine Aussage über den spätesten Zeitpunkt der Stellbefehlsausgabe, sondern eine ebenfalls aus "worst-case"-Betrachtungen entstandene Aussage über den frühesten Zeitpunkt der Stellbefehlsausgabe.

Jedenfalls gingen mit dieser Maßnahme verschärfte Betriebsbedingungen einher, obgleich eigentlich eine Entschärfung beabsichtigt war.

Aus den Laufzeitgesetzen hätte man den Schluß ableiten können, daß die Aufstellung eines Beobachters möglichst am Anfang eines Abschnitts eine Maßnahme zur Entschärfung darstellt. Die letztere Überlegung zeigt jedoch, daß gerade das Gegenteil der Fall ist : Nur, wenn der Beobachter so nahe wie möglich am Ende seines Abschnittes postiert wird, daß sich nicht zwei oder mehr Stücke gleichzeitig im Beobachtungsbereich befinden können, entfällt die Einschränkung bezüglich der zeitlichen Untergrenze für den Stellbefehl.

Ohne Hardware-Hilfsmittel zur Einfahrtsblockierung ist dies genau dann der Fall, wenn zwischen Beobachterposition und Abschnittsende nur noch die Distanz a liegt, der physikalisch mögliche Minimalabstand von Stück zu Stück in Folge.

Eine Beobachtung darf so nur noch um die Zeitspanne blockiert werden, die zwischen dem Eintreffen zweier Stücke im ungünstigsten Fall minimal vergeht. (Zur Abschätzung kann als Grenzwert a/v_{max} herangezogen werden, wenn a die minimalste Distanz von Stück zu Stück und v_{max} die größte Fahrtgeschwindigkeit über alle Abschnitte darstellt.)

7.3.3 Anmerkungen

Sicherungs -Anordnungen

Bei Verletzung der Laufzeitgesetze wird der automatisierte Stückprozeß möglicherweise unerklärliches,

- nicht reproduzierbares und
- zeitabhängiges

Fehlverhalten zeigen, weil vielleicht der aufgetretene "worst-case" nicht so ohne weiteres reproduziert werden kann. So könnte es - abhängig von der Zahl der bewegten Stücke und damit zeitabhängig - zu möglicherweise sogar gefährlichen, auf jeden Fall aber höchst unerwünschten Fehllenkungen von Stücken kommen.

Allein schon die Gefahr, daß Stücke sozusagen "übersehen" werden könnten, widerspricht einer der Hauptmotivationen zur geschlossen prozeßgekoppelten Automatisierung von Stückprozessen : Parallelität von Material- und Informationsfluß.

Meist aus wirtschaftlichen Gründen wird ein bestimmter Durchsatz und eine entsprechende Hardwarekonfiguration des Stückprozesses festgelegt. Man kann dann aber nicht einfach verlangen, ein entsprechendes Rechnersystem möge - wie auch immer - die (natürlich fehlerfreie) Prozeßerfassung, Prozeßführung und Lenkung in dieser Anordnung übernehmen. Wenn das Laufzeitgesetz verletzt wird, kann dies mit Softwaremitteln allein nie gelingen.

Wenn also Software-Mittel nicht aus diesem Dilemma helfen, müssen bei Gefahr der Verletzung des Laufzeitgesetzes Hardware-Hilfsmittel eingesetzt werden. So treten hier unerwartete Rückwirkungen der Software auf die Hardware auf.

Die einzige Abhilfe besteht darin, daß man ein Dialog- bzw. Quittierungssystem aufbaut, mit deren Hilfe ein Stück immer erst dann und nur dann eine bestimmte Stelle überfahren darf, wenn ein positives Stellsignal des Rechners dies ausdrücklich erlaubt.

Ein solches Quittierungssystem ist übrigens in vergleichbaren konkurrierenden Anordnungen durchaus üblich, ja sogar selbstverständlich. So würde es niemand einfallen, etwa in einem on-line-Fakturiersystem eines Supermarktes die "Beobachtung" eines Kassierers "Preis, Warenkennung" zu ignorieren, nur weil gerade zufällig der Zentralrechner zuviele Fakturiervorgänge gleichzeitig zu bearbeiten hat. So kann dort natürlich der Preis der nächsten Ware erst dann in die Tastatur eingetippt werden, wenn dies nach Quittierung der letzten Meldung per Zentrale wieder gestattet wird.

Ist ein Quittierungsverfahren nicht möglich oder nicht erwünscht, weil es Start-Stop-Betrieb voraussetzt[+)], muß man nach anderen Möglichkeiten suchen.

Dezentralisierung

Sicherungs-Vorrichtungen in der Hardware verhindern zwar, daß unerwünschte Prozeßzustände eintreten können. Als Negativ-Maßnahme tragen sie natürlich in keiner Weise zum Erreichen eines bestimmten Durchsatzes bei, ganz im Gegenteil. Möchte man einen bestimmten Materialfluß also trotz Verletzung des 3. Laufzeitgesetzes herbeiführen, so muß man versuchen, die Zahlenwerte in der Beziehung (7.3) zu verändern. (Sind die niedereren Laufzeitgesetze verletzt, braucht man dies natürlich erst gar nicht zu versuchen.)

Nach (7.3) könnte eine Vergrößerung der minimalsten Stückdurchlaufzeit t_{lmin} zur Einhaltung des 3. Laufzeitgesetzes führen. Es ist jedoch evident, daß damit der Durchsatz an dieser Strecke sinken muß und so der Gesamtdurchsatz leiden könnte, weil ja, wie gezeigt, letzlich nur die Fördergeschwindigkeit ein - in gewissen Grenzen - frei wählbarer Parameter ist.

+) Z.B. beim sog. Ablaufberg der Bundesbahn, wo man die ablaufenden Güterwagons nicht anhalten möchte bzw. nicht anhalten kann.

Als nächste Maßnahme böte sich die Verminderung der Beobachterzahl M an. Dabei ginge i.a. sogar die nicht-unterschreitbare Verarbeitungszeit T_v zurück, die ja wegen des Führens des Prozeßmodells möglicherweise von M überproportional abhängig sein kann ; der Gewinn würde so verstärkt. Natürlich geht damit eine Verschlechterung der Möglichkeiten bezüglich der Verfolgung und Lenkung von Stücken einher. Der größere Durchsatz muß so mit geringeren Einwirkungsmöglichkeiten erkauft werden, sofern die Verminderung der Beobachterzahl unter diesem Gesichtspunkt überhaupt möglich ist.

Die Minimal-Abwicklungszeit T_v läßt sich ebenfalls nicht ohne Probleme ändern bzw. verringern, weil die Rechengeschwindigkeit von Rechnern nach oben begrenzt ist.

Die Maßnahme, die einen gewissen Erfolg verspricht, stellt die Verteilung der benötigten Rechenleistung auf mehrere autonome Rechner dar. Setzte man z.B. N Rechner oder Prozessoren ein, so könnte die benötigte Rechenzeitspanne $M \cdot T_v$ eben auf die N Rechner verteilt werden.

Dazu ist die gesamte Stückgutanordnung zu untergliedern, d.h., in N Moduln zu zerlegen, so daß jedem Modul k eine bestimmte Abschnitts- und Beobachterzahl $M_k = c_k \cdot M$ angehört (mit $\sum_{k=1}^{N} c_k = 1$).

Um einen Überblick über die Gesamt-Anordnung zu erhalten, wird ein (N +1). Rechner als Zentrale eingesetzt, mit dem nicht mehr die Beobachter, sondern die "subzentralen" Rechner der Moduln verkehren. Es entsteht so eine hierarchische Rechneranordnung.

In jedem Modul k konkurrieren nur noch die zugehörigen M_k Beobachter um den bezüglich des Moduls lokalen "Oberbeobachter" Nr. k. So muß für jedes Modul nach dem 3. Laufzeitgesetz gelten : $t_{l\,min}(k) \geqslant M_k \cdot T_{vk}$, d.h., die minimalste Stückdurchlaufzeit im Modul k muß größer als das Produkt aus Beobachterzahl M_k und minimalster Abwicklungszeit T_{vk} sein.

So können parallel zum Modul k alle Beobachtungen in allen anderen Moduln von den anderen Rechnern verarbeitet werden. Eine Absprache der Subzentralen untereinander ist nur dann nötig, wenn ein Stück von einem Modul in ein anderes wechselt : Der interne Stückgutverkehr im Modul kann unabhängig von allen anderen und damit echt parallel abgewickelt werden, der grenzüber-

schreitende Verkehr muß dageben überregional gelenkt werden - z.B. über den Zentralrechner bzw. Zentralprozessor.

Die Abwicklungszeitspanne, die für die Modul-lokalen Beobachtungsverarbeitung zur Verfügung steht, wird durch die Zeit, die die Absprache mit einer Zentrale benötigt, entsprechend vermindert[+]. Daraus ergibt sich eine Optimierungsaufgabe, die hier nur umrissen werden soll. Zur Erhöhung der Rechenleistung möchte man einerseits einen möglichst großen Modularisierungsgrad erreichen, um entsprechend viele Rechner einzusetzen. Andererseits erhöht eine große Rechnerzahl die Zahl der nötigen Absprachen, so daß - gemessen am 3. Laufzeitgesetz - die Leistungsfähigkeit der Stückgutanordnung wieder sinkt.

Die ganze Betrachtung setzt jedoch grundsätzlich voraus, daß die "Zentrale", d.h., der übergeordnete Zentralrechner des so entstehenden hierarchischen Rechnersystems nicht über alle Stückbewegungen respektive Abschnittsbelegungen informiert sein muß (und auch nicht sein kann). Diese Verteilung der Rechenleistung per Dezentralisierung bewirkt so, daß die Zentralstelle nur Kenntnis über Inhalte von Stückprozeßmoduln haben kann. (Benötigt man an zentraler Stelle Informationen über jede Stückbewegung und jede Abschnittsbewegung im Detail, so hätte sich durch den Einsatz von N = M Rechnern nichts geändert.)

Der Erfolg der Dezentralisierung, der Verteilung der benötigten Rechenleistung auf N (+1) Rechner muß so von verschiedenen Seiten betrachtet werden[++]:

- Unter dem Gesichtspunkt des - planerisch bedingten - Durchsatzes ist der Einsatz von N (+1) Rechnern erstrebenswert.
- Unter dem Gesichtspunkt der detaillierten Auskunftsbereitschaft ist der Einsatz von N (+1) Rechnern nicht erstrebenswert. Die detaillierte Kenntnis an zentraler Stelle über die Position einzelner Stücke nimmt ab und damit auch die Fähigkeit zu individueller, detaillierter Lenkung der Stücke.

[+] Beträgt die maximalste Absprachezeit mit einem Zentralrechner t_a und sind N Subzentralrechner gegeben, so muß $t_{l\,min}(k) \geq M \cdot T_{vk} + N \cdot t_a$ sein, wie sich mit derselben "worst-case"-Strategie zeigen läßt, die zum 3. Laufzeitgesetz führte.

[++] Aspekte wie Sicherheit, Zuverlässigkeit etc. hierbei ausgeklammert.

7.4 Eine potentielle Verbesserung des Verhaltens des technischen Prozesses

7.4.1 Ein Beispiel

Wie gezeigt, lassen sich gegen die logische Schwelle aller abgeleiteten Laufzeitgesetze grundsätzlich nur im Prinzip negative Hardwareeinrichtungen einsetzen, so daß letztlich eine Aktion vorsichtshalber überhaupt verhindert wird, ehe eine zweifelhafte Aktion - zweifelhaft weil potentiell "richtig" oder "falsch" - unternommen wird. (Dabei ist zu bedenken, daß in bestimmten Situationen jedoch eine nicht unternommene Aktion - juristisch eine "Unterlassung" - genau so falsch und sträflich sein könnte, wie eine unternommene.)

Die Randbedingungen beruhen konsequenterweise auf der Annahme, daß ständige Häufung des "worst-case", d.h., daß ständig die maximale Spitzenlast gegeben ist.

Die maximale Spitzenlast - ständiger paralleler Anfall von M Beobachtungen - ist aus physikalischen Gründen nicht überschreitbar. Es muß jedoch häufig nicht mit deren ständigem, sondern nur mit deren zufälligem Auftreten gerechnet werden. Neben - möglicherweise zufälligen - Zeiten der Spitzenlast gibt es in einer Stückgutanordnung - möglicherweise - auch Zeiten von "normaler" oder geringerer Belastung.

Für diesen Fall lassen sich gewisse, letzlich nur statistisch bestimmbare Verbesserungen erzielen, wenn man die Abarbeitung der parallel anfallenden Beobachtungen ordnet, d.h. organisiert.

Bisher wurde stets angenommen, daß zufällig parallel anfallende Beobachtungen, wie z.B. im Bild 7.1, in beliebiger, willkürlicher Reihenfolge bearbeitet werden können.

Der Auftragsanfall, auf freie Ereignisse zurückgehend, kann gar nicht (oder nur in gewissen Grenzen) gelenkt werden, dagegen aber die Auftragsabwicklung. Wenn mehrere Aufträge (Beobachtungen) gleichzeitig für ein- und denselben Auftragnehmer vorliegen, führt folgende Abfertigungsdisziplin zu einer potentiellen Verbesserung des Prozeßverhaltens :

> Je kürzer die Stückdurchlaufzeit in einem Abschnitt b_i ist, desto höher wähle man die Priorität für die Abfertigung der Beobachtung aus diesem Abschnitt relativ zu anderen Abschnitten.

Der mögliche Vorteil, der durch diese Organisation der Auftragsabwicklung gleichzeitig vorliegender Aufträge erzielbar ist, sei anhand des Bildes 7.4 mit drei Beobachtern erläutert. Die Annahme hierbei ist, daß 3 Stücke von 3 Beobachtern "1", "2" und "3" gleichzeitig gemeldet werden.

- Die Stückdurchlaufzeit auf den 3 Abschnitten sei aus bestimmten Gründen (z. B., weil die zulässigen Maximalgeschwindigkeiten unterschiedlich sind) von Position des Beobachters bis Ende des Abschnittes verschieden ; es gelte

  $$t_{12} < t_{11} < t_{13},$$

  obwohl alle Beobachter so nahe wie möglich am Abschnittsende postiert sind.

- Die Beobachtungen der Beobachter "1", "2" und "3" fallen im Rahmen gewisser Grenzen gleichzeitig an. Die zugehörigen Bedingungen "1", "2" und "3" seien disjunktiv mit einer Task verknüpft.

- Erneute Beobachtungen der selben Beobachter sollen daraufhin irgendwann später anfallen, jedenfalls nicht unmittelbar nach dem Stückdurchlauf. D.h., die aufgetretene Spitzenbelastung - alle 3 Beobachtungen gleichzeitig - sei zufällig ; danach folge eine Zeit relativer Ruhe.

- Das 3. Laufzeitgesetz sei verletzt, nachdem nicht gilt $3\,T_v \leqslant t_{1\,min} = t_{12}$. Die Sicherungs-Vorrichtung wird so auf jeden Fall benötigt, da nur sie die Fehllenkung von Stücken absolut verhindert.

- Läßt man willkürliche Abarbeitung der gleichzeitig vorliegenden Aufträge durch Rechenprozesse zu, so ergeben sich beispielsweise folgende Möglichkeiten :

  - Die Abwicklung "A" nach Bild 7.4 mit der Reihenfolge der Beobachtungen "1", "3", "2". Dann wird, im Bild gezeigt, die Beobachtung "2" zu spät verarbeitet, der Stellbefehl für den Abschnitt 2 kommt zu spät.

  - Die Abwicklung "B" mit der Reihenfolge "1", "2", "3". Dann kommen alle Stellbefehle rechtzeitig, wenn auch für Abschnitt 2 gerade noch.

o Die Abwicklung "C" in umgekehrter Reihenfolge bezüglich der Streckendurchlaufzeit. Sie muß als die günstigste erscheinen, denn gegenüber allen anderen Abfertigungsreihenfolgen kommen die Stellbefehle in allen Abschnitten so frühzeitig an, daß sogar noch Freizeit bleibt zur Erledigung anderer Aufgaben.

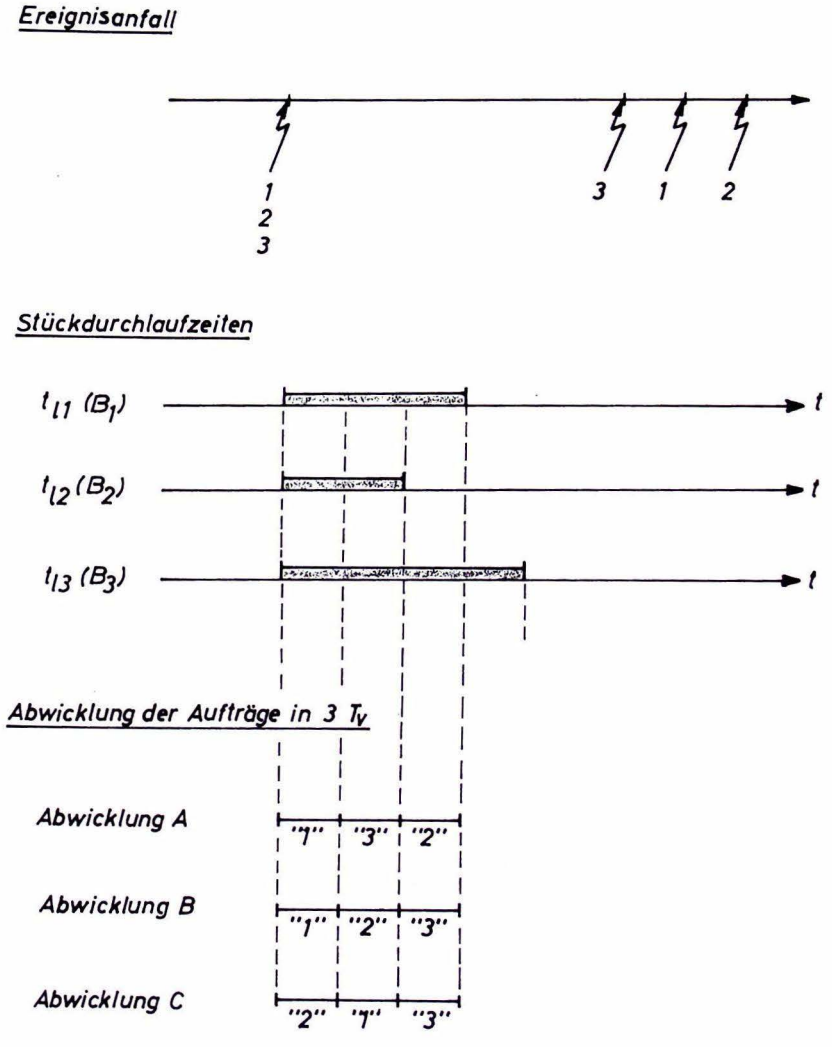


Bild 7.4 : Verbesserung des Prozeßverhaltens bei Organisation der Auftragsabwicklung.

7.4.2 Eine bedingte funktionelle Anforderung an die Eigenschaften einer Prozeß-programmiersprache

Mit der Abfertigungsreihenfolge "A" in Bild 7.4 müßte das beobachtete Stück auf dem Abschnitt B_2 mittels Sicherungs-Einrichtung angehalten werden, bis der quittierende Stellbefehl für dieses Stück eintrifft (nach $3 * T_v$). Damit einher geht eine Verschlechterung des Prozeßverhaltens, die - ob drastisch oder nicht - mittels Organisation der Abfertigung sofort unterbunden werden könnte.

Der Erfolg der Abfertigung "C" liegt darin begründet, daß man durch Organisation den momentanen Engpaß auf den in Bezug auf die Durchlaufzeit längsten Abschnitt verlagert hat. +)

Die Verwendung der Abwicklungsstrategie nach der kürzesten Stückdurchlaufzeit setzt voraus, daß die

> Abfertigung gleichzeitig vorliegender Aufträge

gesteuert werden kann. Damit kann das Verhalten des technischen Prozesses unter bestimmten Voraussetzungen beeinflußt werden.

Die Forderung, die sich hieraus auf die Eigenschaften einer Prozeßprogrammiersprache ableiten läßt, sei wie folgt formuliert :

> Die Abfertigungsdisziplin von gleichzeitig vorliegenden Aufträgen an einen Auftragnehmer sollte steuerbar sein.

Es genügt also nicht allein, in einer Prozeßprogrammiersprache die Konkurrenz mehrerer paralleler Rechenprozesse untereinander um diverse Betriebsmittel zu steuern. Die angestellten Überlegungen zeigen, daß quasi auch die Konkurrenz gleichzeitig vorliegender Aufträge um die Abwicklungsinstanz steuerbar sein sollte.

Diese Anforderung an die Eigenschaften einer Prozeßprogrammiersprache ist nicht unabdingbar, weil sie, wie erwähnt, nur auf sozusagen "statistischer Verbesserung" des Prozeßverhaltens beruht.

+) Eine Überwindung der Schwelle des 3. Laufzeitgesetzes erbringt diese Maßnahme jedoch nicht, weil bei ständigem Auftreten des "worst-case" der Engpaß der laufzeitmäßig kürzesten Strecke bestehen bleibt.

Hat man diese Möglichkeit der Steuerung der Auftragsabwicklung nicht, dann ist für die Automatisierung des Stückprozesses damit kein prinzipielles, unüberwindliches Hindernis entstanden. Man begibt sich aber gewisser Vorteile, weil, wie das Beispiel zeigt, eine im Grunde unnötige Verschlechterung des Verhaltens des technischen Prozesses (Anhalten des Stückes im Abschnitt "2" bei Abwicklung "A") in bestimmten Situationen nicht zu verhindern ist.

So hätten fehlende Spracheigenschaften indirekt unerwünschte Rückwirkungen auf das technische Prozeßgeschehen.

7.4.3 Ein einfacher Lösungsvorschlag

Wie im vorangehenden Kapitel dargelegt, wird als Sprachmittel zur Auftragserteilung auch hier die Aktivierungsanweisung als bevorzugt angesehen. Um die Konkurrenz paralleler Aufträge zu regeln, wird die bedingte Aktivierungsanweisung erneut erweitert um die PRECEDENCE-Option wie folgt :

(7.5) bedingte_aktivierungsanweisung :: =

```
schedule    ACTIVATE
            taskbezeichner  [(taskparameterliste)]
            [PRECEDENCE ( {,.ganzzahl ...} )]
            [PRIORITY ausdruck]
```

Die Semantik der bedingten Aktivierungsanweisung mit PRECEDENCE-Option soll sein :

- Bei jedem Ereignis-Eintritt, der im "schedule" beschrieben ist, wird ein Aktivierungs-Auftrag an den Auftragnehmer "taskbezeichner" abgesetzt.
- Neben den möglichen aktuellen Parametern begleitet den Auftrag ein Vorrang-Kennzeichen, das sich aus der "ganzzahl"-Liste hinter der PRECEDENCE-Anweisung ergibt.
- Das Vorrang-Kennzeichen für den Aktivierungs-Auftrag wird ermittelt durch Abzählen in der "ganzzahl"-Liste entsprechend dem aufgetretenen Einzelereignis aus dem Ereignisteil "schedule".

- Die Abfertigung gleichzeitig vorliegender Aufträge für den Auftragnehmer "taskbezeichner" richtet sich nach dem auftragsbegleitenden Vorrangkennzeichen. Je kleiner diese Ganzzahl ist, desto eher wird der Auftrag abgewickelt und umgekehrt.
- Die Priorität - ebenfalls ein Auftragsbegleiter - steuert im Gegensatz hierzu die Konkurrenz des Rechenprozesses "taskbezeichner" zu anderen Rechenprozessen.

Hierzu ein Beispiel zur Lösung des Problems nach Bild 7.4 :

Beispiel 7.1 : 3 Beobachter gemäß Bild 7.4

```
WHEN ITR (1 : 3) ACTIVATE T (ORIGIN) PRECEDENCE (2, 1, 3) ;
.
.
T : TASK (URSPRUNG) ;
SPECIFY URSPRUNG FIXED ;
DECLARE IN_IDENT BIT (10) ;
TAKE FROM EX_IDENT (URSPRUNG) TO IN_IDENT;/* Identität ablesen */
CALL ALGORITHMUS (URSPR, IN_IDENT) ;
END ; /* ENDE T */
```

Die 3 Unterbrechungen ITR(1), ITR(2) und ITR(3) sollten bei Eintreten die Task T aktivieren. Der mitgegebene Parameter ist der Ursprung nach Kapitel 6, und der Vorrang der Aufträge ist durch die PRECEDENCE-Option durch Abzählen wie folgt festgelegt : Bei Eintreten des Ereignisses ITR(1) erhält der abgesetzte Aktivierungsauftrag das Vorrangkennzeichen 2, bei ITR(2) wird 1 und bei ITR(3) wird 3 als Vorrangkennzeichen vergeben. Wenn, wie im Bild 7.4, zufällig alle 3 Beobachtungen "gleichzeitig" anfallen, so entstehen per bedingter Aktivierungsanweisung 3 Aufträge. Der Auftragnehmer ist aber durch die begleitenden Vorrangkennzeichen gezwungen, zuerst die Beobachtung des Beobachters Nr. 2, dann die des Beobachters 1 und schließlich die des Beobachters 3 zu erledigen.

Auf diese Weise wird die gewünschte Abfertigungsreihenfolge erreicht.

Sicherlich handelt es sich bei diesem Vorschlag nicht um die einzig denkbare Möglichkeit. Andere Lösungen, etwa das Zulassen von Variablen, Ausdrücken, Feldern, Zeigern, Zeiten etc. an der PRECEDENCE-Position erscheinen möglich. Wesentlich ist nur, daß ein aufgrund vieler potentieller Ereignisse entstandener Auftrag entspechend nach den Wünschen des Programmierers abgearbeitet werden kann.

Wenn - wie beim M-Beobachter-Problem - mit möglicherweise großen Ereignisfeldern in der Aktivierungsanweisung gerechnet werden muß, erscheint das Anlisten von M Vorrang-Nummern in der PRECEDENCE-Ganzzahlliste als nicht sehr befriedigend. Für den Fall, daß es gleiche Vorrangzahlen für verschiedene Auftragsquellen gibt, könnte man sich Methoden der Abkürzung o.ä. ausdenken (z.B. Replikatoren). Der eigentliche Arbeitsaufwand, der hinter dieser Liste steht, ist die Steuerung der Auftragskonkurrenz. Das Ergebnis dieser Überlegungen sind diese Vorrangzahlen. Diese müssen dem Organisator ja irgendwann mitgeteilt werden, so daß sich i.a. das Anlisten von Zahlenkolonnen wohl nicht vermeiden lassen wird, an welcher Stelle im Programm auch immer.

7.5 Schlußbemerkungen

7.5.1 Puffergröße, Auftragsverlust und Auftragsabfertigung

Im Kapitel 6 wurde dargestellt, warum nicht nur die Pufferung von Aufträgen als solche, sondern auch die (Nachrichten-) Puffergröße mit Mitteln einer Prozeßprogrammiersprache steuerbar sein muß. Die Frage nach der im aktuellen Anwendungsfall zu verwendenden Puffergröße blieb dabei jedoch unbeantwortet, genauso wie die Frage nach der Abfertigungsdisziplin bezüglich der gepufferten Aufträge.

Die Überlegungen zur Konkurrenzsituation im Stückprozeß dieses Kapitels führten zu folgenden Ergebnissen :

1) Die Pufferzahl M ist zur Lösung der Aufgabe "Stückgutlenkung und -Verfolgung mittels M Beobachter "notwendig und hinreichend.

   Notwendig, weil alle M zufällig gleichzeitig anfallenden Beobachtungen in Form von Aufträgen gespeichert werden müssen. Hat man weniger Pufferplätze zur Verfügung, riskiert man im "worst-case" den ersatzlosen Verlust von Aufträgen.

   Hinreichend ist die Pufferzahl M deswegen, weil aus physikalischen Gründen Ereignisse bei ein- und demselben Beobachter nicht beliebig dicht aufeinander, sondern höchstens in bestimmten, zeitlich nicht unterschreitbaren Abständen eintreten können. Bei Einhaltung des 3. Laufzeitgesetzes werden selbst im ständigen "worst-case" alle M Beobachtungen so rechtzeitig bearbeitet, daß genau M Pufferplätze ausreichend sind. Damit geht kein Auftrag verloren. Umgekehrt bewirkt eine Erhöhung der Pufferplätze bei Verletzung des 3. Laufzeitgesetzes zwar eine geringere Wahrscheinlichkeit des Auftragsverlustes ; sie bewirkt damit aber nicht, daß Fehllenkungen vermieden werden.

   (Das bei endlicher Pufferlänge und rein stochastischem Ereignisfall zu erwartende "Warte-Verlust-System" tritt bei Einhaltung des 3. Laufzeitgesetzes und der Pufferzahl M nicht auf.)

2) Die Organisation der Abfertigung gleichzeitig vorliegender Aufträge durch einen Auftragnehmer kann nachteilige Auswirkungen im Verhalten des technischen Prozesses verhindern.

7.5.2 Zyklische Zustandserfassung

In dieser Arbeit wurde bisher fast ausschließlich die Erfassung des Zustandes des technischen Prozesses anhand von freien Ereignissen, der sog. spontanen Zustandserfassung untersucht. Daneben gibt es aber auch die Möglichkeit, den Prozeßzustand nicht auf freie Ereignisse, sondern sozusagen "auf Verdacht" hin zu untersuchen.

In regelmäßigen Abständen - zyklisch - werden dabei der Reihe nach alle Beobachter - ob identifizierend oder nicht - nach ihren letzten gültigen Beobachtungen befragt. Danach müssen alle Beobachtungen des letzten Zyklus mit denen des jetzigen verglichen werden, so daß auf diese Weise die Veränderung des Prozeßzustandes, d.h., des Belegungszustandes nachvollziehbar ist.

Der Einwand, daß durch das zyklische gegenüber dem spontanen Verfahren die sehr häufige Unterbrechung von anderen Rechenprozessen vermieden werden könnte, ist in dieser Form nicht stichhaltig. Zyklische Beauftragung von Rechenprozessen - sieht man von dem pathologischen Fall der zeitkonsumierenden Schleife ab - wird wohl immer über hardwaremäßige Unterbrechungseinrichtungen erreicht, indem man nach Ablauf von bestimmten Zeitspannen entsprechende, von einer Uhr stammende Unterbrechungssignale verwendet. Daß diese Unterbrechungssignale dem Rechner i.a. nicht gesondert extern zugeführt werden müssen, sondern eine Verbindung meist intern bereits existiert, ändert an dieser Tatsache ja nichts.

Nur die Tatsache, daß die zyklische Auftragserteilung vom Organisator im Rahmen der normalen Abwicklung der periodischen, organisator-internen Zeitzählung mit abgewickelt werden kann, verhindert die Erhöhung der Unterbrechungsrate. (Ansonsten wäre die Belastung ganz im Gegenteil eher höher.)

Dem hieraus eventuell resultierenden Vorteil der zyklischen Zustandserfassung steht der Nachteil gegenüber, daß quasi "auf Verdacht" immer etwas unternommen werden muß.

Zu Zeiten der "worst-case" Spitzenlast besteht zwischen beiden Methoden sicherlich nur ein geringer Unterschied. Wenn dagegen das Geschehen im technischen Prozeß etwas ruhiger ist, bietet die Methode der ereignisgesteuerten Zustandserfassung den Vorteil, daß mehr Rechner-Freizeit zur Erledigung anderer Nebenaufgaben zur Verfügung steht.

Dieser Effekt läßt so die spontane Zustandserfassung als ungleich attraktiver erscheinen. Welche Schwierigkeiten dabei auftreten, zeigt die vorliegende Arbeit.

8. Zusammenfassung

Die vorliegende Arbeit ist unter der Zielsetzung entstanden, diejenigen Eigenschaften einer Prozeßprogrammiersprache zu ermitteln, die bei der Automatisierung eines konkreten, typischen technischen Prozesses benötigt werden. Dabei erfolgte gezielt die Beschränkung auf die Klasse der Stückprozesse.

In den Kapiteln 2 und 3 wurden daher zunächst die Grundlagen für die Arbeit zusammengestellt. Anhand des Ansatzes der Massen- und Stückzahlbilanz über abgegrenzte, abgeschlossene Materialflußanordnungen wurden logische Prozeßelemente ermittelt, die allgemein zur Erklärung des Prozeßgeschehens herangezogen werden können. Verfahren zur Bildung von Prozeßmodellen schließen sich an. Sie sind auf die Abstraktion in Form von Graphen ausgerichtet, wobei neue, zweckmäßige Zuordnungen von Prozeßelementen zu den Elementen eines Graphen vorgeschlagen werden.

Eine Untersuchung zur Stückverfolgung und -Lenkung diente als Ausgangspunkt zur Unterteilung der globalen Aufgabenstellungen in makroskopische und mikroskopische Aufgaben. Die Institution des "Beobachters" stellt als Denkmodell eine wichtige Ausgangsposition bei der Erfassung des Prozeßzustandes dar und führt auf das "M-Beobachter-Problem".

Das im Labor aufgebaute gegenständliche Modell eines Stückprozesses wird in den wesentlichsten funktionellen Teilen vorgestellt. Am Beispiel des Modellprozesses - der auch der praktischen Demonstration des vom Autor mitimplementierten PEARL-Compilers dient - wurden die vorgestellten Modelle und Methoden konkret veranschaulicht.

Nach diesen grundlegenden Vorarbeiten sind im Kapitel 5 die grundsätzlichen Anforderungen an eine Prozeßprogrammiersprache wiedergegeben, die sich bei der Stückgutverfolgung ergeben. Es sind Sprachmittel, die zur Bewältigung vieler parallel anfallender Ereignisse bzw. deren Einplanung gebraucht werden. Hieraus resultierte das Auftraggeber oder Ursprungsproblem.

Vor dem Hintergrund der Verfolgung vieler gleichzeitig anfallender Stückbewegungen findet sich in Kapitel 6 als eines der zentralen Themen dieser Arbeit eine grundlegende Diskussion der bisher ungelösten Probleme bei der Auftragserteilung an Tasks, d.h., bei der Bewältigung von Bearbeitungswünschen in Rechnern.

Nach Zusammenstellung der Grundlagen, einleitender Definition der in diesem Zusammenhang eingeführten Begriffe, nach der grundlegenden Unterteilung in tätigkeitsauslösende und tätigkeitshemmende Aufträge anhand der PEARL-Tasking-Operationen und der allgemeinen Definition eines Lebenslaufs von Aufträgen wurden drei modellhafte, elementare Mechanismen zur Bindung von Auftraggeber und Auftragnehmer vorgeschlagen. Sie unterscheiden sich durch ihre Bindeglieder, die zur Weiterleitung eines Bearbeitungswunsches von seiner Geburt bis hin zu seiner Abwicklung mittels Rechenprozeß eingesetzt werden. Es ergab sich, daß mit dem Mechanismus der auftragsbegleitenden Nachricht die Probleme bei der Beherrschung vieler paralleler, unabhängig voneinander eintretender Ereignisse mittels Rechner befriedigend lösbar sind, die bei der Ortsveränderung vieler voneinander unabhängiger Stücke auftreten.

Anhand des M-Beobachter-Modellansatzes beschäftigt sich das Kapitel 7 abschließend mit einigen Konkurrenzsituationen in Stückprozessen, die bei der zentralen Sammlung von Beobachtungen entstehen. Anhand einer "worst-case"-Betrachtung wurden sog. Laufzeitgesetze abgeleitet. Anhand des wichtigsten 3. Laufzeitgesetzes, das auf der impliziten Konkurrenz der Beobachter untereinander beruht,wurden die Voraussetzungen erbracht, unter denen aufgrund gewisser physikalischer Randbedingungen im Stückprozeß eine endliche Zahl von Auftragspufferplätzen hinreichend und notwendig ist. Es wurde gezeigt, daß vorranggesteuerte Auftragsabarbeitungsstrategien in bestimmten Konkurrenzsituationen Vorteile erbringen können.

Anhang 1

A. Verteilungen in Inzidenz- und Adjazenzmatrizen von Graphen

A.1 Inzidenzmatrix

Ein Graph mit p Ecken und q Kanten muss $2 * q$ Inzidenzen aufweisen, weil per definitionem jede Kante mit 2 Ecken inzidiert. Die Inzidenzmatrix besteht aus $p * q$ Elementen, so daß

(A.1) $$z_{oi} = p * q - 2 * q = q\,(p-2)$$

Plätze mit Nullen besetzt sind. Bezogen auf die Gesamt-Elementzahl sind relativ

(A.2) $$z_{ri} = \frac{q\,(p-2)}{p * q} = 1 - 2/p$$

Plätze mit Nullen besetzt.

Bemerkenswert ist, daß diese Relation unabhängig von der Kantenzahl q ist.

A.2 Adjazenzmatrix

Die Nachbarschaft zweier Ecken ergibt sich durch die verbindende Kante, d.h., es gibt q Nachbarschaften. Im ungerichteten Graphen, in der die Adjazenzmatrix symmetrisch sein muß, sind daher $2 * q$ Elemente mit 1, oder umgekehrt

(A.3) $$z_{oa} = p^2 - 2q$$

der p^2 Matrizenplätze mit 0 besetzt. Der relative Anteil beträgt dann

(A.4) $$z_{ra} = \frac{p^2 - 2q}{p^2} = 1 - 2q/p^2$$

Im Digraphen sind bei q gerichteten Kanten nicht mehr 2 q, sondern nur noch q gerichtete Verbindungen möglich, so daß sich der relative Anteil der Nullen ergibt zu

(A.5) $$z_{ra'} = 1 - q/p^2$$

Ein Unterschied zwischen (A.4) und (A.5) besteht im Grunde nicht, weil die Adjazenmatrix des ungerichteten Graphen symmetrisch ist, und so die Hälfte aller Einsen den Informationsgehalt 0 besitzen.

Anhang 2

Verzeichnis der Kurzzeichen

A	Adjazenzmatrix eines Graphen
a_{ij}	Elemente der Adjazenzmatrix A
B	Inzidenzmatrix eines Graphen
B_m	Beobachter Nr. m
b_m	Beobachtungsbereich m
b_{ij}	Elemente der Inzidenzmatrix B
I	Menge der Stückidentitäten in einem Stückprozeß
i_k	Identität des Stückes k $i_k \epsilon I$
M	Beobachteranzahl im Stückprozeß
S	Menge aller Stücke in einer Stückgutanordnung
s_i	Stück i $s_i \epsilon S$
t_{ai}	$= t_a(b_i)$: Gesamt-Abwicklungsspanne von Auftragsgeburt bis Auftragsbeendigung im Beobachtungsabschnitt b_i
t_{li}	Stückdurchlaufzeit im Beobachtungsabschnitt b_i
t_{lmin}	minimalste Stückdurchlaufzeit aller Abschnitte
T_v	maximale Verarbeitungszeit des zentralen Oberbeobachters für jede beliebige Beobachtung
p	Zahl der Ecken eines Graphen
q	Zahl der Kanten eines Graphen
V	Menge der Ecken eines Graphen
v_i	Ecke i eines Graphen $v_i \epsilon V$
X	Menge der Kanten eines Graphen
x_i	Kante i eines Graphen $x_i \epsilon X$

Literaturverzeichnis

/1.1/ Schmitz, P. : Die Wirksamkeit von Programmiersprachen, Betriebswirtschaftlicher Verlag Dr. Th. Gabler, Wiesbaden (1972)

/1.2/ DIN 44300 : Informationsverarbeitung, Begriffe. Beuth-Verlag (1972), Nr. 44

/1.3/ Lauber, R. : Prozeßautomatisierung I, Springer Verlag (1976)

/1.4/ Timmesfeld, K.H. et al.: PEARL - Vorschlag für eine Prozeß- und Experimentautomatisierungssprache, PDV-Bericht KFK-PDV 1 (April 1973), Gesellschaft für Kernforschung mbH Karlsruhe

/1.5/ Gruber et al. : ASME-PEARL-SUBSET/1, PDV-Bericht Nr. KFK-PDV 76 (Mai 1976), Gesellschaft für Kernforschung, Karlsruhe

/1.6/ Eichenauer, B. : Ein portabler Übersetzer für einen Subset der Prozeßprogrammiersprache PEARL, in Lecture Notes in Computer Science (Editor Goos, Hartmanis), Nr. 12, GFK-GI-GMR Fachtagung Prozeßrechner 1974, S. 576 ff, Springer-Verlag Berlin, Heidelberg, New York

/1.7/ Helfert, M.E. : Implementation des PEARL-Compiler-Oberteils auf der AEG 60-50 des IRP, PDV-Entwicklungsnotiz Nr. PDV-E 64 (März 1976), Gesellschaft für Kernforschung, Karlsruhe

/1.8/ Helfert M.E. et al. : E/A- und Segmentierungsschnittstelle des oberen Compilerteils zu den Zielmaschinen, ASME-Interne-Notiz Nr. S-013 (12.8.74) (unveröffentlicht)

/1.9/ Helfert, M.E. : Steueranweisungen an den PEARL-Compiler der AEG 60-50, ASME-Interne Notiz Nr. S-030 (26.8.75)(unveröffentlicht)

/1.10/ Leutzbach, W. : Einführung in die Theorie des Verkehrsflusses, Springer-Verlag, Berlin (1972)

/1.11/ Becher, F. : Optimierung der Lagerhaltung durch Simulation, dhf 12/72, S. 724

/1.12/ Dreger, W. : Kapazität komplexer Fördersysteme, fördern und heben 23 (1973) Nr. 14, S. 749

/1.13/ Gudehus, T. : Grundgesetze fördertechnischer Systeme, Materialfluß- und Staugesetze, fördern und heben 23 (1973) Nr. 11, S. 601

- " - : Engpässe in Fördersystemen, fördern und heben (1975) Nr. 3/4, S. 206

- " - : Grenzleistungsgesetze für Verteiler und Sammelelemente, fördern und heben 25 (1975) Nr. 16, S. 1532

- " - : Ermittlung der Planungsgrundlagen für Warenverteil- und Lagersysteme, dhf 3/72, S. 100

Gudehus, T. : Warteschlangen und Wartezeiten in Warenverteil- und Lagersystemen, fördern und heben 23 (1973), Nr.5, S. 225

- " - : Leistungsberechnung für Kommissioniersysteme, in Referate des 1. Europäischen Materialflußkongreß, Verlag moderne industrie (1974), S. 147 ff

- " - : Grundlagen der Spielzeitberechnung für automatisierte Hochregallager, dhf Sonderheft 1972, S. 210

/1.14/ Heym, M. : Spezielle Probleme der Transportoptimierung, dhf Sonderheft 1972, S. 216

/1.15/ Jansen, R. : Optimale Transportsysteme- richtig geplant mit der Nutzwertanalyse, fördern und heben 24 (1974) Nr. 11, S. 1063

/1.16/ Jünemann, R. : Stückgut lagern und verteilen, fördern und heben, (1975) Nr. 6, S. 604

/1.17/ Jünemann R. : Systemplanung für Stückgutlager, Krauskopf-Verlag, Mainz (1971)

/1.18/ Krieg, W. : Integraler Warenfluß, Industrielle Organisation 44 (1975) Nr. 7, S. 341

/1.19/ Rupp, R.H. : Planung und Realisierung von Lager- und Verteilsystemen, dhf Sonderheft (1973), S. 300

/1.20/ Schaab, W. : Automatisierte Hochregalanlagen, Bemessung und Wirtschaftlichkeit, VDI-Verlag Düsseldorf (1969)

/1.21/ Schippkühler, J. : Zur Optimierung der Fördervorgänge vor und in einem Hochregallager. Dissertation TU Berlin (1972)

/1.22/ Stemmer, G., Fischer, W. : Darstellung diskontinuierlicher Förderabläufe in Simulationsmodellen, fördern und heben 25 (1975), Nr. 11, S. 1069 und S. 1465

/1.23/ Bendeich, E., Kölle, J. : Projektierungshilfsmittel beim Einsatz von Prozeßrechnern für Stückgutprozesse in der Fertigungstechnik, PDV-Bericht KFK-PDV 97, Gesellschaft f. Kernforschung, Karlsruhe (Okt. 1976)

/1.24/ Kadlec, V., Vodacek, L. : Lineare Optimierung im Transportwesen, Westdeutscher Verlag Köln/Opladen (1968)

/1.25/ Ford, L.R., Fulkerson, E. : Flow in Networks, Princeton University Press (1962)

/2.1/ VDI-Richtlinie 2411 : Begriffe und Erläuterungen im Förderwesen, VDI-Verlag Düsseldorf (1970)

/2.2/ VDI-Richtlinie 3565 : Klassifizierung von Stückgut für Stetigförderer, VDI-Verlag Düsseldorf (März 1971)

/2.3/ Saeltzer, G. : Theorie und Simulation von Stückgutprozessen, Messen Steuern Regeln Nr. 17 (1974), H. 8, S 269 ff

/2.4/ Todt, H.J. : Systemtheoretische Ansätze für die Materialflußgestaltung, Forschungsberichte VDI, Reihe 13, Nr. 11, VDI-Verlag Düsseldorf (1971)

/2.5/ Strobel, H. : Anwendung der modernen Regelungstheorie im Verkehrswesen, Messen Steuern Regeln Nr. 17 (1974), H. 11, S 392 ff.

/2.6/ Herbst, D., Kremser, J. : Studie über innerbetriebliche Fördersysteme als Vorbereitung zur Erweiterung des Einsatzes der problemspezifischen Fördersprache PSF, PDV-Bericht KFK-PDV 71 (Mai 1976), Gesellschaft für Kernforschung, Karlsruhe

/2.7/ DIN 19226 : Regelungstechnik und Steuerungstechnik, Begriffe und Benennungen, Beuth-Verlag Berlin

/2.8/ Steinbuch, K., Rupprecht, W. : Nachrichtentechnik, Springer-Verlag (1967), S. 327 ff.

/2.9/ VDI-Richtlinie 3300 : Materialfluß-Untersuchungen, VDI-Verlag, Düsseldorf (Aug. 1973)
VDI-Richtlinie 2689 : Leitfaden für Materialflußuntersuchungen, VDI-Verlag Düsseldorf (Jan. 1974)

/2.10/ DIN 66201: Prozeßrechensysteme, Begriffe, Beuth-Verlag Berlin (Aug. 1971)

/2.11/ Harary, F. : Graphentheorie, Oldenburg Verlag München, Wien (1974)

/2.12/ Mayeda, W. : Graph Theory, Wiley-Interscience Verlag, New York (1972)

/2.13/ Herbst, D., Kremser, J. : Spezielle Probleme der allgemeinen Programmierung von Förderprozessen, in "Wege zur Automatisierung im Förder- und Lagerwesen", Seminar an der Universität Dortmund (7.11.75)

/2.14/ Stemmer, G. : Sensitivity Analysis of an internal Transportation System by Means of Simulation, IFAC-Workshop on Optimization Applied to Transportation Systems, Wien (1976)

/2.15/ Gudehus, T. : Grundlagen der Kommissioniertechnik, Verlag Giradet, Essen (1973), S. 11 ff

/2.16/ Hopgood, F.R.A. : Compiling Techniques, Macdonald Verlag, London (1970)

/2.17/ Gries, D. : Compiler Construction for Digital Computers, (1971), John Wiley & Sons, Inc, New York

/2.18/ Knuth, D.E. : The Art of Computer Programming - Sorting and Searching, Addision-Wesley Publishing Company, London (1974)

/3.1/ Leutzbach, W. : Einführung in die Theorie des Verkehrsflusses, Springer Verlag, Berlin (1972)

/3.2/ Dörrscheidt, F., Klittich, M. : Untersuchungen zum Einschleusvorgang, Technische Mitteilungen AEG-Telefunken 65 (1975) Nr. 8, S. 323

/3.3/ Schaab, W. : Automatisierte Hochregalanlagen, VDI-Verlag, Düsseldorf (1969)

/3.4/ Wiedenmann, R. : Untersuchung der Eignung der Prozeßrechnersprache PEARL zur zyklischen Prozeßdatenerfassung, Dissertation Universität Stuttgart (1978)

/4.1/ Lauber, R. : Experimentelle Untersuchung von Spracheigenschaften der Prozeßrechnersprache PEARL anhand von Modellprozessen in : Praxis von Sprachen, Programmiersystemen und Programmgeneratoren (Herausg. D. Krönig), C. Hauser Verlag München Wien (1976)

/4.2/ Knoll, H-G. : Programmierung der Stückverfolgung und -Lenkung in einem Stückprozeß mit PEARL, Diplomarbeit am Institut f. Regelungstechnik und Prozeßautomatisierung, Universität Stuttgart (März 1977)

/4.3/ Ferschl, F. : Zufallsabhängige Wirtschaftsprozesse, Physica-Verlag Wien und Würzburg (1964)

/4.4/ Gudehus, T. : Grundlagen der Kommissioniertechnik, Verlag Girardet, Essen (1973), S. 101 ff

/4.5/ Steimer, F. : Projektierung des Transport und Verteilsystems für den Modellprozeß "Stückprozeß", Diplomarbeit am Institut für Regelungstechnik und Prozeßautomatisierung, Universität Stuttgart (Jan. 1974)

/4.6/ Franzius H. : Stand und weitere Entwicklung der Förder- und Lagertechnik in "Grundsätze und Methoden der Materialflußgestaltung", Lehrgang Nr. 2076/53.02/4, Technische Akademie Esslingen (Okt. 1973)

/4.7/ Albert, F. : Warenverteilsystem zum Modellprozeß, Diplomarbeit am Institut für Regelungstechnik und Prozeßautomatisierung, Universität Stuttgart (1974)

/4.8/ Kramp, E. : Positioniermöglichkeiten für Regalförderzeuge, BBC-Nachrichten (1973), Heft 5, S. 121 ff.

/4.9/ Unkel, H.-J. : Rechnersteuerung eines Regalförderzeugs mit Hilfe des Prozeßrechners AEG 60-50, Diplomarbeit am Institut für Regelungstechnik und Prozeßautomatisierung, Universität Stuttgart (1975)

/4.10/ Ghassemi, A. : Untersuchung der Eignung der Prozeßprogrammiersprache PEARL zur Automatisierung von Folgeprozessen, Dissertation Universität Stuttgart (1978)

/5.1/ Eichenauer, B.F. : Dynamische Prioritätsvergabe an Tasks in Prozeßrechensystemen, Dissertation Universität Stuttgart (1975)

/5.2/ Rößler, R., Schenk, K. (Editors) : LTPL-European Group : Language Comparision (Juli 1975), EG Brüssel

/5.3/ Helfert, M.E. : Zur Problematik von Einplanungen, der Auftraggeber-Identifikation und der Pufferung in PEARL, SAK-Bericht 31-76 (2.4.76) Gesellschaft für Kernforschung mbH, Karlsruhe

/5.4/ Helfert, M.E. : Einige fundamentale Anforderungen an eine Prozeßprogrammiersprache anhand der Automatisierung von Stückprozessen, SAK-Bericht 27-75 (4.11.75) Gesellschaft für Kernforschung mbH, Karlsruhe (unveröffentlicht).

/5.5/ Lucas, K. : Änderungsvorschläge zu PEARL, SAK-Ä Nr. 11-75 (24.11.75), SAK-Ä 12-75 (24.11.75), SAK-Ä 12-76 (10.2.76), alle Gesellschaft für Kernforschung mbH, Karlsruhe (unveröffentlicht).

/5.6/ Koch, Lexa : Bewertung der Beschlüsse des PEARL-SAK bezüglich des Scheduling und Bereinigungsvorschläge, Studie Nr. 41/76 (19.7.76) BBC Mannheim (unveröffentlicht)

/5.7/ Helfert, M.E. : Änderungsvorschläge zu PEARL, SAK-Ä Nr. 5-76 (21.1.76) Gesellschaft für Kernforschung mbH, Karlsruhe (unveröffentlicht)

/6.1/ VDI/VDE-Richtlinie 3554 (Entwurf) : Funktionelle Beschreibung von Prozeßrechner-Betriebssystemen (März 1976), VDI-Verlag, Düsseldorf

/6.2/ Rieder, P. : Prozeßzustände bei Echtzeitprogrammiersprachen, GFK-GI-GMR Fachtagung Prozeßrechner 1974, in Lecture Notes in Computer Science, Nr. 12, Springer-Verlag, Berlin (1974)

/6.3/ Spitzingsee-Klausur des PEARL-Subset-Arbeitskreises : Tasking, SAK-Bericht Nr. 8-76 (15.1.76), Gesellschaft für Kernforschung mbH, Karlsruhe, Projekt Prozeßlenkung mit Datenverarbeitung (unveröffentlicht)

/6.4/ Wegner, E., Hopmann, C. : Towards a Description of PEARL apt for Standardization, Institut für Software-Technologie, Gesellschaft für Mathematik und Datenverarbeitung, Bonn (3.2.1975)

/6.5/ Wegner, E., Hopmann, C. : Towards a Description of PEARL apt for Standardization, Growing Darft, Institut für Software-Technologie, Gesellschaft für Mathematik und Datenverarbeitung, Bonn (12.12.1975)

/6.6/ Wegner, E. : Fragen zum Resume, SAK-Bericht Nr. 33-75 (12.12.75), Gesellschaft für Kernforschung mbH, Karlsruhe (unveröffentlicht)

/6.7/ DIN 66200 (2. Entwurf) : Betrieb von Rechensystemen, Begriffe, Auftragsabwicklung (April 1976), Beuth-Verlag, Berlin und Köln

/6.8/ Helfert, M.E. : Vorschlag zur Lösung des Auftraggeber-Problems in PEARL, SAK-Änderungsvorschlag 5-76 (21.1.76), Gesellschaft für Kernforschung mbH, Karlsruhe (unveröffentlicht)

/6.9/ Dijkstra, E.W. : Co-Operating Sequential Processes. In : Programming Languages (Editor Genuys), Academic-Press, London (1968)

/6.10/ Brinch Hansen, P. : Operating System Principles (1973), Prentice-Hall Inc., Englewood Cliffs, New Jersey

/6.11/ Wegner, E. : Baumstrukturierte Programme, Dissertation Technische Universität Berlin (1974)

/6.12/ Bautz, J. : Auftragsverkehr über einen generellen Auftragspuffer bei Prozeßrechnersystemen, GI-Tagung Karlsruhe (1972), Lecture Notes in Computer Science, Springer-Verlag, vol 1

/6.13/ Schwald, A., Baumann, R. : PEARL im Vergleich mit anderen Echtzeitsprachen, Vortragsmanuskript zum PEARL-Aussprachetag (9.3.1977)

/6.14/ Zeh, A. : Untersuchung der Zuverlässigkeit der Prozeßrechnersprache PEARL bei der Steuerung und Synchronisation von Tasks, unveröffentlicht

Lebenslauf

Am 15. März 1945 wurde ich in Saaz im Sudetenland als Sohn des Bankkaufmanns Ernst Helfert und seiner Ehefrau Friedericke, geb. Sacher, geboren. Als deutsche Staatsangehörige wurden wir noch im Dezember 1945 ausgewiesen. Nach kurzem Aufenthalt in Thüringen siedelten wir 1946 nach Adelschlag Kreis Eichstätt in Bayern um, wo ich die Grundschule von 1951 bis 1954 besuchte.

In Böblingen besuchte ich die Grundschule ein weiteres Jahr, und in Sindelfingen von 1955 bis zum Abitur 1964 das Goldberggymnasium.

Nach einem halbjährigen Praktikum begann ich im Wintersemester 1964/65 das Studium der Elektrotechnik an der damaligen Technischen Hochschule Stuttgart, das im Wintersemester 1970/71 mit einer Diplomarbeit bei Prof. Dr.-Ing. W. Kaiser abgeschlossen wurde.

Von 1971 bis 1972 war ich im Bereich der Groß-EDV tätig und wickelte eigenverantwortlich konventionell-kommerzielle Automatisierungsprojekte ab.

Seit Juni 1972 bin ich als wissenschaftlicher Mitarbeiter am Institut für Regelungstechnik und Prozeßautomatisierung der Universität Stuttgart, Direktor Prof. Dr.-Ing. R. Lauber bei der Abwicklung des Forschungsvorhabens "Prozeßlenkung mit Datenverarbeitungsanlagen (PDV), Prozeßprogrammiersystem auf der Grundlage von PEARL" beteiligt. Im Rahmen dieses Vorhabens entstand vorliegende Arbeit.

M.E. Helfert

Eine Untersuchung der Anforderungen an eine Prozeßprogrammiersprache bei der Automatisierung von Stückprozessen

Kernforschungszentrum Karlsruhe GmbH
PDV-Bericht KfK-PDV 135, Oktober 1978
152 Seiten, 39 Abbildungen, 1 Tabelle, 74 Literaturstellen

Die operativen Aufgabenstellungen bei der prozeßgekoppelten Automatisierung von Stückprozessen werden zusammengestellt. Aufgrund einer Untersuchung von Elementen und Struktur von Stückprozessen werden abstrakte Prozeßmodelle entwickelt. Der gegenständliche Modellaufbau eines Stückprozesses im Labor und dessen Betrieb an einem Prozeßrechner AEG 60-50 mittels PEARL-Programmen dient u.a. der Veranschaulichung und praktischen Demonstration der Verfahren. Die Ortsveränderung von Stücken ist Anlaß, die Anforderungen an eine Prozeßprogrammiersprache bei der Automatisierung von Stückprozessen zu diskutieren und zusammenzustellen.

M.E. Helfert

Eine Untersuchung der Anforderungen an eine Prozeßprogrammiersprache bei der Automatisierung von Stückprozessen

Kernforschungszentrum Karlsruhe GmbH
PDV-Bericht KfK-PDV 135, Oktober 1978
152 Seiten, 39 Abbildungen, 1 Tabelle, 74 Literaturstellen

Die operativen Aufgabenstellungen bei der prozeßgekoppelten Automatisierung von Stückprozessen werden zusammengestellt. Aufgrund einer Untersuchung von Elementen und Struktur von Stückprozessen werden abstrakte Prozeßmodelle entwickelt. Der gegenständliche Modellaufbau eines Stückprozesses im Labor und dessen Betrieb an einem Prozeßrechner AEG 60-50 mittels PEARL-Programmen dient u.a. der Veranschaulichung und praktischen Demonstration der Verfahren. Die Ortsveränderung von Stücken ist Anlaß, die Anforderungen an eine Prozeßprogrammiersprache bei der Automatisierung von Stückprozessen zu diskutieren und zusammenzustellen.

M.E. Helfert

An Investigation of the Functional Requirements of a Process Control Computer Language Used in the Automation of Material Flow Processes.

Kernforschungszentrum Karlsruhe GmbH
PDV-Report KfK-PDV 135, October 1978
152 pages, 39 figures, 1 table, 74 references

The operative tasks at the automation of material flow processes are compiled. Abstract mathematical process models are developed by investigating the elements and structures of material flow processes. A hardware model of a material flow system, built up in the laboratory, operated by a process computer AEG 60-50 with PEARL-Programs, serves for demonstration of the methods. Investigating the change of location of material, the functional requirements of a process control computer language to be used for the automation of material flow processes are discussed and collected.

M.E. Helfert

An Investigation of the Functional Requirements of a Process Control Computer Language Used in the Automation of Material Flow Processes.

Kernforschungszentrum Karlsruhe GmbH
PDV-Report KfK-PDV 135, October 1978
152 pages, 39 figures, 1 table, 74 references

The operative tasks at the automation of material flow processes are compiled. Abstract mathematical process models are developed by investigating the elements and structures of material flow processes. A hardware model of a material flow system, built up in the laboratory, operated by a process computer AEG 60-50 with PEARL-Programs, serves for demonstration of the methods. Investigating the change of location of material, the functional requirements of a process control computer language to be used for the automation of material flow processes are discussed and collected.